Hindawi Publishing Corporation
EURASIP Journal on Applied Signal Processing
Volume 2006, Article ID 12093, Pages 1–10
DOI 10.1155/ASP/2006/12093

Fast and Accurate Ground Truth Generation for Skew-Tolerance Evaluation of Page Segmentation Algorithms

Oleg Okun and Matti Pietikäinen

Infotech Oulu and Department of Electrical and Information Engineering, Machine Vision Group, University of Oulu, P.O.Box 4500, FI-90014, Finland

Received 15 February 2005; Revised 30 May 2005; Accepted 12 July 2005

Many image segmentation algorithms are known, but often there is an inherent obstacle in the unbiased evaluation of segmentation quality: the absence or lack of a common objective representation for segmentation results. Such a representation, known as the ground truth, is a description of what one should obtain as the result of ideal segmentation, independently of the segmentation algorithm used. The creation of ground truth is a laborious process and therefore any degree of automation is always welcome. Document image analysis is one of the areas where ground truths are employed. In this paper, we describe an automated tool called GROTTO intended to generate ground truths for skewed document images, which can be used for the performance evaluation of page segmentation algorithms. Some of these algorithms are claimed to be insensitive to skew (tilt of text lines). However, this fact is usually supported only by a visual comparison of what one obtains and what one should obtain since ground truths are mostly available for upright images, that is, those without skew. As a result, the evaluation is both subjective; that is, prone to errors, and tedious. Our tool allows users to quickly and easily produce many sufficiently accurate ground truths that can be employed in practice and therefore it facilitates automatic performance evaluation. The main idea is to utilize the ground truths available for upright images and the concept of the representative square [9] in order to produce the ground truths for skewed images. The usefulness of our tool is demonstrated through a number of experiments with real-document images of complex layout.

1. INTRODUCTION

Segmentation is an important step in image analysis since it detects homogeneous regions whose characteristics can then be computed and analyzed, for example, for discriminating between different classes of objects such as faces and nonfaces. However, the unbiased evaluation of segmentation results is difficult because it requires an ideal description of what one should obtain as the result of segmentation of a certain image regardless of the segmentation algorithm. This ideal description, known as the ground truth, can be utilized for judging whether segmentation is correct or not, and how well a given image is segmented. The generation of ground truths is laborious and often prone to errors, especially if done manually. Thus, any degree of automation brought to this procedure is typically welcome.

Document page segmentation is one of the areas of image analysis where ground truths are to be employed. Page segmentation divides an image of a certain document, such as a newspaper article or advertisement, into homogeneous regions, each containing data of a particular type such as text, graphics, or picture. The accuracy of page segmentation is of great importance since segmentation results will be an input for higher-level operations, and errors in segmentation can lead to those in character recognition. Performance evaluation of page segmentation algorithms is therefore necessary because it can help to identify errors inherent to a given algorithm and to understand their reasons. The performance is typically assessed either visually by a human or by using ground truths.

The first approach consists of visually checking the obtained results and making conclusions about their correctness. This approach is subjective (thus unreliable) and tedious. That is why researchers have normally used a small or relatively moderate number of images in testing page segmentation algorithms.

The second alternative is automated and therefore more attractive when it is necessary to process a large number of images. It uses a special (usually text) file, called a ground truth (GT), for each image, containing a description of different regions that should be detected during correct segmentation. This description typically includes (for each

region) a polygon or rectangle surrounding the region together with the type of data constituting the region (types are text, graphics, picture), which is compared to the results obtained after applying the page segmentation algorithm under consideration. It is worthy to mention that several different correct segmentations of the same image are sometimes possible, and it is often difficult (or may be even hardly possible) to define the term "best or ideal segmentation", which all other segmentations can be compared to.

In this paper, we describe an experimental tool called GROTTO (ground truthing tool) to generate many ground truths in an automatic and computationally cheap manner when page skew (tilt of text lines) is present.[1] To our best knowledge, no existing ground truthing method [2, 5, 7, 8, 12] *explicitly* aims at this task. Its importance is, however, motivated by the following reasons. A majority of the existing GTs were created for upright images, that is, for those without skew, because their creation is usually quite time consuming. To be able to use them when skew is present, one typically needs to scan the skewed page and then to compensate for the skew. In this case, it will be difficult to judge the skew tolerance of a given page segmentation algorithm because segmentation will be done when there is no skew. Errors in skew estimation are also possible, which can deteriorate the whole process.

Keeping this in mind, GROTTO attempts to reduce the cost of GT creation by making some of the most laborious operations more automatic. The main idea is to use the GTs for upright images in order to generate those for skewed images. We consider that the rotation transformation applied to the upright image is sufficient to produce the skewed image for any angle. Such an assumption gives us two advantages. Firstly, we know the skew angle and secondly, we exclude the scaling and translation, that is, we avoid a sophisticated, time consuming and not always accurate document registration procedure.[2] In fact, neither printing nor scanning of the original document is necessary in our approach. All that we need is the GT for the corresponding upright image. In the next section, modern approaches to ground truth generation are briefly reviewed.

2. BRIEF OVERVIEW OF GROUND TRUTHING STRATEGIES

Because our interest in this paper is the GT generation for skew-tolerance evaluation, we will not consider methods aiming at pure text images, where the task is mostly to correctly place bounding rectangles around each character (for details, see [5, 6, 8, 10]).

When using GTs, there are two different approaches to the performance evaluation of page segmentation algorithms. In the first approach, the judgment about segmentation quality is based on OCR results [4, 7, 10]. In this case, the GT is often a simple sequence of correct characters. It is sometimes assumed that the OCR errors do not result from segmentation errors [7], so that the number of operations (manipulations with characters and text blocks) needed to convert the OCR output into the correct result can be used to evaluate the segmentation performance. This approach may hide the reason for error when both types of error appear in the same place and it does not make the segmentation evaluation independent of the whole document analysis system.

In the second approach, the segmentation results are directly compared with the corresponding GTs without using OCR [2, 3, 11, 12]. In this case, the GTs contain region descriptions as sets of pixels [11, 12], rectangular zones [3], or isothetic polygons whose edges are only horizontal or vertical [2].

When ground truthing is done at the pixel level [11, 12], it provides the finest possible details and can represent regions of arbitrary shape. However, as reported in [3], there can be more than 15 million pixels per page digitized at 400 dpi. That is why it usually takes much time to verify and edit such GTs. The evaluation of the segmentation quality is achieved by testing the overlap between sets of pixels. It consists of computing the number of splittings and mergings needed to obtain correct segmentation.

The representation of regions as rectangular zones takes less memory than the pixel-based one, but it is less efficient because it is only suitable for representing constrained layouts, where all regions have a rectangular shape and no adjacent bounding rectangles are overlapped. The segmentation performance is evaluated by the overlap and alignment of zones in the GT and those obtained after segmentation [3]. No calculation of how zones were split, merged, missed, or inserted is done by suggesting that this information is difficult to derive and therefore it will be quite difficult to make the right conclusions based on it. It is important to notice that the method [3] only checks the spatial correspondence of zones by ignoring their class labels (text, graphics, background) so that the obtained GTs do not contain complete information about segmentation errors.

The representation of page regions as isothetic polygons [2] tries to combine the advantages of pixel- and zone-based schemes, while aiming to overcome their drawbacks. It is supposed that any arbitrarily shaped region can be described by an isothetic polygon that can be decomposed into a number of rectangles called horizontal intervals in [2]. Such a scheme seems to be flexible and efficient enough to deal with complex nonrectangular regions and nonuniform region orientation, and it describes regions without significant excess to the background. Also it has lower memory requirements than a bitmap (though they may be higher than those for the zone-based scheme). The GTs are automatically generated by using the method described in [1]. After that, a manual correction of the GT may be necessary. The correspondence between the GT and segmentation results is determined by interval overlap. In spite of the advantages of isothetic polygons, it is still not very clear how to obtain those

[1] We concentrate in our article on this type of degradation and do not consider other possible types such as scaling, which typically less frequently occur for page images.

[2] Document registration may be able to solve a more complex and general task than we consider but at the expense of higher complexity and as a result, of higher chances for failures. In contrast, we aim at a simpler solution while sacrificing some generality.

polygons *automatically* from images with color and textured background (the authors in [2] claim that it is possible, but no examples were given). The method [1] that is used to construct such polygons, exploits the concept of white tiles (or white background spaces) surrounding document regions. In the case of color images, for example, the background may not be white or it may not have a single color for all regions and it may not be uniform.

3. OUR APPROACH TO THE PROBLEM

As one can see from the brief discussion of various ground truthing strategies, one of the first tasks is to choose a proper representation for page regions. The region representation affects the method of comparing a GT and segmented image.

To summarize, there are three main alternatives: set of pixels, rectangular zone, and polygon. When a region is described as a set of pixels [12], it will take a lot of memory to keep this representation and a lot of time to manipulate it. Rectangular zones [3] represent regions in a more compact manner than pixels, but they cannot accurately enough describe complex-shaped nonrectangular regions and they are not suitable when skew is present. Polygons can deal with both problems mentioned, but they are not flexible enough when it is necessary to access the data inside them (it will usually take much more time than when using rectangles). One exception is isothetic polygons [2], but they need to be partitioned into a number of rectangles before being really convenient to use, and it seems that this representation is restricted to binary images only.

Since no representation scheme has clear advantages over the others and there are no established standards, we introduce our own representation based on image partitioning into small nonoverlapping blocks of $N \times N$ pixels, where N depends on the image resolution and is determined by the formula $N \leq 2 \cdot (\text{res}/25)$, where res is the resolution in dots per inch (dpi), res/25 is the resolution in dots per mm, and 2 means two mm. We assume that in any case the corresponding block on the paper should not occupy an area larger than $2 \times 2\,\text{mm}^2$. For example, the maximum value of N is equal to 24 for a resolution of 300 dpi. In our opinion, the block partitioning combines advantages of other representations such as compactness, easiness and direct access to data, applicability to different image types (binary, grayscale, and color) and various document layouts (rectangular and arbitrary), and tolerance to skew. It is also widely used in image compression.

The block-based GT can be defined as a 2D array of numbers, each of which is a block label associated with the following classes: text (T), background (B), binary graphics (G), and grayscale or colored image (I). If a block contains data of several classes, it has a composite label including the labels of all classes. That is, 11 composite labels are possible in addition to 4 noncomposite ones. Given that $n_{\text{row}} \times n_{\text{col}}$ are sizes of the original image, sizes of the block-based GT associated with this image are much smaller ($\lfloor n_{\text{row}}/N + 0.5 \rfloor \times \lfloor n_{\text{col}}/N + 0.5 \rfloor$), where $\lfloor \cdot \rfloor$ means the smallest integer closest to a given number; 0.5 is introduced to deal with image sizes that are not multiples of N.

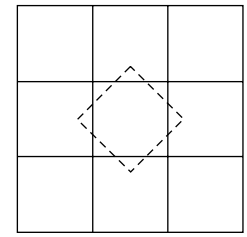

Figure 1: Direct approach to SGT generation.

Based on such a definition of the GT, the correspondence between the GT and segmentation results can be found in a straightforward way—*by computing the difference of class labels of the blocks*. Difference statistics, found for each of the 11 cases, fully quantify and qualify segmentation results. It is also quite easy to convert the pixel-, zone-, and even some polygon-based (such as composed of a set of rectangles) GTs to the block-based format.

4. CONCEPT OF THE REPRESENTATIVE SQUARE

Let us suppose that a block-based GT for an upright image (UGT) is available (we will show how to generate it below). By having this GT and by knowing the skew angle α and N, the problem now is how to obtain the GT for the skewed image (SGT). A straightforward solution could be to partition the skewed image into $N \times N$ pixel blocks, to rotate the corner points of each block in that image by $-\alpha$ around the image center, and finally to match the rotated block and blocks in the upright image in order to determine a label of the block in question. We call this the direct approach. However, labeling may not be simple, especially if one would like to assign labels based on the intersection area of blocks (see Figure 1) because those areas are polygons rather than rectangles.

Because of this reason, we employ the concept of the representative square [9].

Definition 1. The representative square (RS) of a given block in the skewed image is a square slanted by angle α ($\alpha \in [-90°, +90°]$) and placed inside the block so that its corner points lie on block sides (see Figure 2).

The angle by which the RS is slanted is equal to the desired skew angle for which the GT needs to be created.

The following proposition is true for any RS (proof is given in [9]).

Proposition 1. *Let $ABCD$ and $A'B'C'D'$ be a block in the skewed image and its RS, respectively. In this case, the ratio $r = S_{\text{RS}}/S_{\text{Block}} \geq 0.5$, where S_{RS} and S_{Block} are the* RS *and block areas, respectively.*

That is, the RS always occupies at least half the area of a block in the skewed image. Since the maximum sizes of this block on the paper are very small, we can use the RS instead

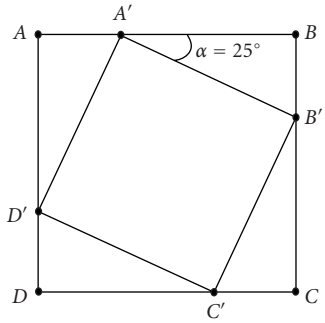

Figure 2: Representative square $A'B'C'D'$ of the block $ABCD$.

of the corresponding block (i.e., why it is called representative) without loosing much data.

Notice that all that is necessary to do is to model the document skew. However, we neither scan the skewed page nor rotate the upright image, but use the GT for the upright image and *the concept of the representative square* in order to generate the GTs for the skewed images. As will be shown, this leads to very fast processing, while the results are still accurate to apply our technique in practice.

5. SGT GENERATION METHOD

Given a skew angle α (we assume that positive (negative) angles correspond to clockwise (counterclockwise) rotation), N, and the corresponding block-based UGT, then the SGT generation, using the concept of the representative square, consists of the following steps.

(1) Given the width and height of the upright image, compute the width and height for the skewed image.
(2) Partition the skewed image into nonoverlapping blocks of $N \times N$ pixels. Each block can be considered as one big pixel so that it is possible to speak about rows and columns of blocks.
(3) Compute coordinates of the point A' (see Figure 2) for the upper-left block in the skewed image and coordinates of this point and the point C' in the upright image when rotating both points about the image center by $-\alpha$.
(4) Scan the blocks from left to right, top to bottom. For each block, find the position of the rotated (about the center of the skewed image by $-\alpha$) RS in the upright image and classify the corresponding block in question based on the labels of the UGT blocks it crosses.

Given H and W (height and width of the upright image), the corresponding H' and W' of the skewed image are computed as $W' = H|\sin\alpha| + W\cos\alpha$ and $H' = H\cos\alpha + W|\sin\alpha|$.

From the proof of Proposition 1, RS width (height) d_{RS} is equal to $N/(|\sin\alpha| + \cos\alpha)$. Coordinates of the point A' of the upper-left block in the skewed image are (here and further we assume that the upper-left image corner has coordinates (1,1))

$$x_{A'}^{\text{skewed}} = \begin{cases} d_{\mathrm{RS}}\sin\alpha & \text{if } \alpha \geq 0, \\ 1 & \text{otherwise,} \end{cases}$$
$$y_{A'}^{\text{skewed}} = \begin{cases} 1 & \text{if } \alpha \geq 0, \\ d_{\mathrm{RS}}|\sin\alpha| & \text{otherwise.} \end{cases} \tag{1}$$

When rotating A' about the center of the skewed image by $-\alpha$, we obtain that its coordinates in the upright image are determined for $\alpha \geq 0$ as

$$\begin{aligned} x_{A'}^{\text{upright}} &= x_{\text{center}}^{\text{upright}} + (1 - x_{\text{center}}^{\text{skewed}})\cos\alpha + (1 - y_{\text{center}}^{\text{skewed}})\sin\alpha \\ &\quad + d_{\mathrm{RS}}\sin\alpha\cos\alpha, \\ y_{A'}^{\text{upright}} &= y_{\text{center}}^{\text{upright}} - (1 - x_{\text{center}}^{\text{skewed}})\sin\alpha + (1 - y_{\text{center}}^{\text{skewed}})\cos\alpha \\ &\quad - d_{\mathrm{RS}}\sin\alpha\sin\alpha, \end{aligned} \tag{2}$$

and for $\alpha < 0$ they are

$$\begin{aligned} x_{A'}^{\text{upright}} &= x_{\text{center}}^{\text{upright}} + (1 - x_{\text{center}}^{\text{skewed}})\cos\alpha + (1 - y_{\text{center}}^{\text{skewed}})\sin\alpha \\ &\quad + d_{\mathrm{RS}}|\sin\alpha|\sin\alpha, \\ y_{A'}^{\text{upright}} &= y_{\text{center}}^{\text{upright}} - (1 - x_{\text{center}}^{\text{skewed}})\sin\alpha + (1 - y_{\text{center}}^{\text{skewed}})\cos\alpha \\ &\quad + d_{\mathrm{RS}}|\sin\alpha|\cos\alpha. \end{aligned} \tag{3}$$

Coordinates of C' in the upright image are trivial to compute because the rotation transformation preserves distances, that is, $x_{C'}^{\text{upright}} = x_{A'}^{\text{upright}} + d_{\mathrm{RS}}$ and $y_{C'}^{\text{upright}} = y_{A'}^{\text{upright}} + d_{\mathrm{RS}}$.

By knowing $x_{A'}^{\text{upright}}, y_{A'}^{\text{upright}}, x_{C'}^{\text{upright}}, y_{C'}^{\text{upright}}$ computed for the upper-left block in the skewed image, it is now very easy to find these coordinates for the other blocks without any rotation. Thus this will speed up SGT generation because the number of floating point operations is greatly reduced. It is easy to verify that the following formulas are correct for two adjacent blocks *previous* and *next* in the same row:

$$\begin{aligned} x_{A'}^{\text{upright}}(\text{next}) &= x_{A'}^{\text{upright}}(\text{previous}) + N\cos\alpha, \\ y_{A'}^{\text{upright}}(\text{next}) &= y_{A'}^{\text{upright}}(\text{previous}) - N\sin\alpha, \\ x_{C'}^{\text{upright}}(\text{next}) &= x_{A'}^{\text{upright}}(\text{next}) + d_{\mathrm{RS}}, \\ y_{C'}^{\text{upright}}(\text{next}) &= y_{A'}^{\text{upright}}(\text{next}) + d_{\mathrm{RS}}. \end{aligned} \tag{4}$$

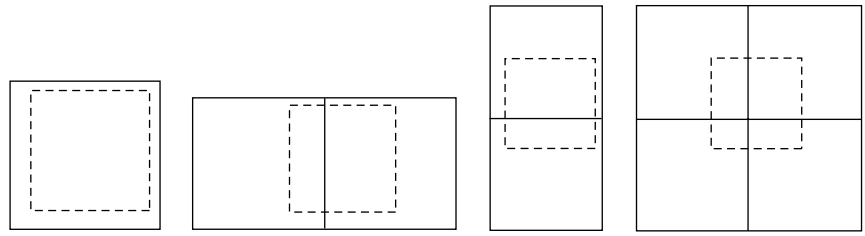

Figure 3: Four cases of intersection of a rotated RS (shown dashed) and blocks in the upright image.

For two adjacent blocks *previous* and *next* located in the same column, the formulas are

$$\begin{aligned} x_{A'}^{\text{upright}}(\text{next}) &= x_{A'}^{\text{upright}}(\text{previous}) + N\sin\alpha, \\ y_{A'}^{\text{upright}}(\text{next}) &= y_{A'}^{\text{upright}}(\text{previous}) + N\cos\alpha, \\ x_{C'}^{\text{upright}}(\text{next}) &= x_{A'}^{\text{upright}}(\text{next}) + d_{\text{RS}}, \\ y_{C'}^{\text{upright}}(\text{next}) &= y_{A'}^{\text{upright}}(\text{next}) + d_{\text{RS}}. \end{aligned} \tag{5}$$

After the coordinates of A' and C' in the upright image have been computed, the task is to label blocks in the skewed image. Four cases are only possible as shown in Figure 3. Four corner points of each RS are checked to determine which UGT blocks they belong to. Initially all blocks in the skewed image are labelled as background. Blocks whose RSs lie completely inside the zone of class T change their labels to T, while blocks whose RSs cross the border between two zones with different labels obtain both labels.

6. GROTTO

Having described the theory behind block-based ground truth generation, GROTTO is introduced in detail in this section. Its main modes of operation are the following.

(i) Given a raw image (either upright or skewed), that is, one without GT, generate a block-based GT.
(ii) Given a block-based GT for an upright image, generate one or several block-based GTs for the specified skew angle and range of skew angles.
(iii) Given a block-based GT for an upright or skewed image, verify the accuracy of generation for this GT.

6.1. Mode 1: GT generation for raw images

With this mode, the input image has no GT and the task is to generate a block-based GT. After opening the image and setting two parameters (image resolution in dpi and block size in pixels), a user should manually detect page regions either as rectangles or as arbitrary polygons and assign a class label to each detected region. Rectangles can be nested in other rectangles or polygons but polygons cannot. Furthermore, the user can edit regions by resizing rectangles, changing region class labels, deleting polygons or rectangles, and partitioning polygons into rectangles. The last operation is done automatically and it is mandatory because we need to have a zone-based representation of regions before GT generation. This representation is also useful to obtain because there are many other GTs based on it. It is worthy to say that many known ground truthing methods rely on manual region extraction before GT generation. This is because the regions should be extracted very accurately, otherwise ground truthing is meaningless. The manual extraction can guarantee this if done carefully, while there is no page segmentation method that can outperform all the others and not be prone to errors.

Polygon partitioning is done with a split-and-merge type procedure. First, the bounding box of the polygon in question is divided into $N \times N$ pixels blocks and the center of each block is checked to determine whether this block belongs to the polygon or not. When doing this, each block gets one of the labels (0:background; 1:block belongs to the polygon in question; and 2:block belongs to another polygon or rectangle). After that, the polygon's bounding box recursively splits into smaller rectangles in alternating horizontal and vertical directions until a given rectangle contains all 1's or/and 0's or its size is smaller than one block. After each split step, all rectangles obtained by then are checked for possible merging. A criterion for merging is that two rectangles should have the same length of their common border. Rectangles containing only 0's or/and 2's are eliminated from further processing and excluded from the list of rectangles. The representation obtained after polygon partitioning is called a *zone-based GT*. Examples of polygon partitioning are given in Figure 4.

To obtain a UGT, we analyze zones one-by-one by superimposing each on a block-based partitioned upright image. For each zone, all blocks located completely inside or crossing it get a class label of that zone. In the first case, blocks are not further processed, while in the second case, the area of intersection between the zone and each block is computed and accumulated in the 12 most significant bits of a 16- bits value associated with a block (the 4 least significant bits are used as a class label).

After all zones have been processed, blocks with values higher than 15 ($2^4 - 1$) are analyzed because they contain data from several classes. If the intersection area for a block is more than (it can happen when adjacent zones are overlapped) or equal to $N \times N$, it means that there is no background space in a block; otherwise, the label "background" is added to the other labels. Because N cannot be more

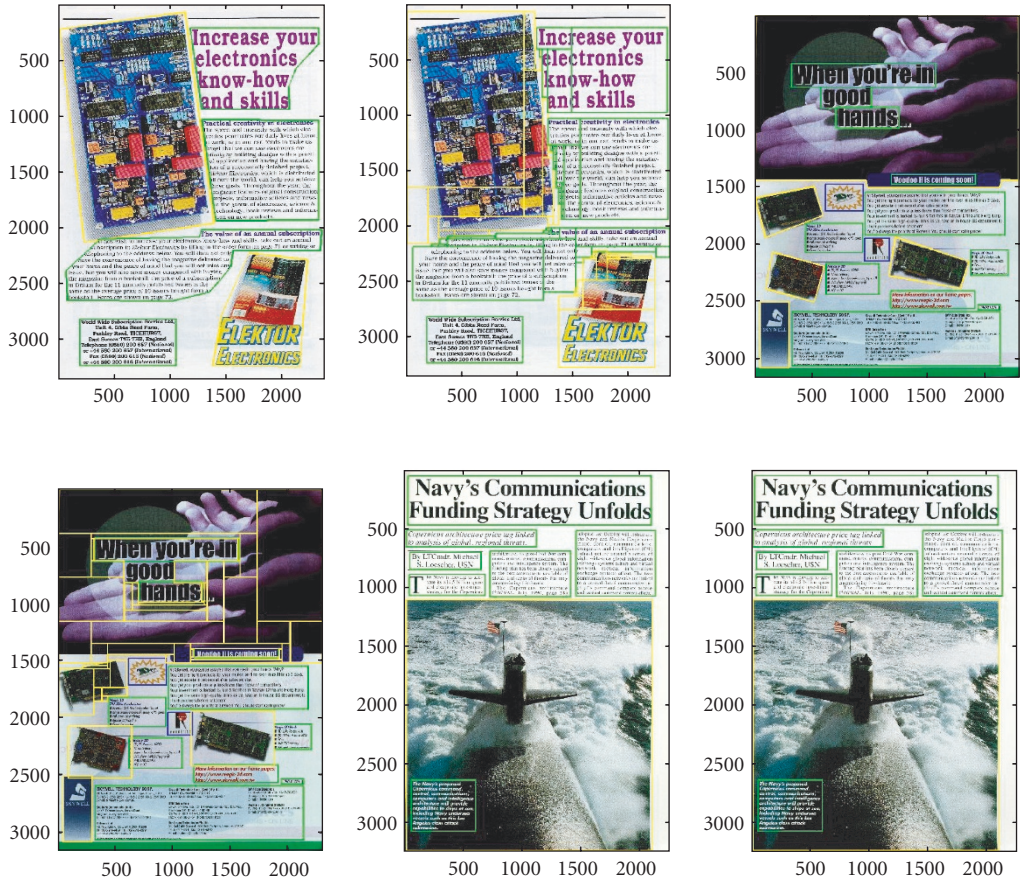

FIGURE 4: Results of manual region detection and automatic polygon partitioning into rectangles. Yellow, green, and blue rectangles and polygons enclose image, text, and graphics regions, respectively.

than 24 for 300 dpi and 32 for 400 dpi, which are the most wide-spread image resolutions, 12 bits are quite enough to represent values of the intersection area in many real cases.

6.2. Mode 2: GT generation for skewed images

By having a zone-based region representation, the user can *automatically and quickly* generate SGTs for a specified range of skew angles or just for a single angle. The process of SGT generation is described in Section 5 in detail and it needs a UGT. The basic operations are the same as they are for Mode 1.

6.3. Mode 3: verification of GT generation

Having a generated SGT, one may ask whether it was accurately generated based on the respective UGT or not. Another question would be how accurate this generation was. To answer to these questions, GROTTO allows a user to automatically verify the quality of an SGT by comparing the generated SGT and the so-called *ideal* GT (IGT). This comparison defines *the absolute performance of a system*. In other words, it means a comparison between the performance of our method of GT generation and that of the ideal one.

Though the IGT is also block-based, its generation employs different principles. First, a list of adjacency is created that includes IDs of the zones in a UGT that either overlap or have a common border. For such zones, if a block contains data from two zones, one does not need to add a background label. It is quite trivial to determine whether background space is between two zones or not.

There is no background between zones, that is, two zones are adjacent, only if the following conditions are both satisfied:

$$\begin{aligned} W_1 + W_2 &\geq \max\left(x_2^{ur} - x_1^{ul} + 1, x_1^{ur} - x_2^{ul} + 1\right), \\ H_1 + H_2 &\geq \max\left(y_2^{ll} - y_1^{ul} + 1, y_1^{ll} - y_2^{ul} + 1\right), \end{aligned} \tag{6}$$

where W_i (H_i) $(i = 1, 2)$ stands for width (height) of the ith zone, (x_i^{ul}, x_i^{ur}) means x-coordinates of the upper-left and upper-right corners of the ith zone, and (y_i^{ul}, y_i^{ll}) means y-coordinates of the upper-left and lower-left corners of the ith zone.

After the list of adjacency is created, the corner points of all zones in a UGT are rotated by the skew angle, sizes of a skewed image are computed, and this image is partitioned into $N \times N$ pixels blocks. For every block, each of its four

corner points is checked to determine which zones it belongs to. A class label of that zone is then assigned to the block in question. If two corner points belong to different zones, we look at the list of adjacency. If IDs of zones are absent in this list, a background label is added to other labels.

The IGT generation is quite slow because we match every corner point to every zone in order to have as accurate and complete information as possible. We assume that this procedure can give the most accurate result at the block level.

The comparison of SGT and IGT is very simple and it consists of matching the class labels of the blocks in two GTs. To do this, we created a special look-up table containing complete coverage of all possible cases.

7. EXPERIMENTS

GROTTO was implemented in MATLAB for Windows with no part of the code written in C. Our experiments with GROTTO included SGT generation for different skew angles from -90° to $+90^\circ$ in 1° steps and accuracy verification for the obtained SGTs as described in Section 6.3. The value of N was set to 24 pixels because the original images had a resolution of 300 dpi. The test set consisted of 30 color images of advertisements and magazine articles. We will use three of them to demonstrate experimental results.

The original upright images did not have ground truths so we manually extracted regions, labeled them, and automatically partitioned polygons into rectangular zones. Results of region detection and polygon partitioning are shown in Figure 4. The last image does not have any regions detected as polygons, therefore no polygon partitioning was done.

Once the zone-based GT was available, the block-based UGT and SGT were automatically created, given N and the skew angle or range of angles. The accuracy of the obtained SGT was verified by comparing this ground truth to the block-based IGT. This operation provides us information about errors in the SGT generation process. Several SGTs, IGTs, and the so-called difference maps highlighting the blocks having different labels in an SGT and IGT are shown in Figure 5 for the images in Figure 4.

To create a difference map, a special look-up table was used to match IGTs and SGTs. In this table, the first row corresponds to class labels of blocks in an IGT, while the first column represents class labels of blocks in an SGT. An intersection of the nth row and the mth column gives a specific value for a certain case, for example, one class missing or added. There are 11 cases as follows: (1) one class label is missing; (2) two class labels are missing; (3) three class labels are missing; (4) one class label is added; (5) one class label is wrong, but other class labels are correct; (6) one class label is missing and one class label is wrong, but at least one class label is correctly assigned; (7) two class labels are added; (8) one class label is added and one class label is wrong, but at least one class label is correctly assigned; (9) three class labels are added; (10) all class labels are correctly assigned (completely correct result); (11) all class labels are wrong (completely wrong result).

It can be seen in Figure 5 that composite labels in SGTs/IGTs, that is, those including multiple classes, occur only in the borders of different classes. Because an IGT provides a more accurate way to generate ground truth than an SGT, the number of composite blocks in an IGT is smaller than that in an SGT for a certain skew angle and block size (red lines in Figure 5 are thicker for SGTs than for IGTs). As for difference maps, three to four cases clearly dominate over others. They are cases 1, 4, 7, and 10, with case 10 (completely correct labeling) far exceeding the others.

To proceed from visual to quantitative estimation of SGT generation, we computed the number of blocks falling under each case as a percentage of the total number of blocks for every pair of SGT and IGT. It turned out that on average more than 90 percent of the blocks in each SGT obtained correct labels (see Figure 6 for an accumulated statistics over all tested images). Due to the cautious way of GT generation when blocks can rather obtain several labels than only one label in ambiguous situations, case 4 was more frequent than case 1. Because of the background spaces between regions and domination of these two cases, we concluded that the background was missing or added. This factor, in our opinion, does not significantly degrade the SGT generation accuracy because the background does not contain much useful information. Cases 1 and 4 correspond to the missing rate and false alarm rate, respectively, and in Figure 6, the missing rate is five times smaller than the false alarm rate! This manifests high accuracy of ground truth generation with GROTTO.

Finally, Figure 7 provides the distributions over the whole angle range from -90° to $+90^\circ$ in 1° steps for cases 1, 4, and 10. For angles of 0, -90, and 90 degrees, IGTs and SGTs completely coincide so that the percentage of case 10 is equal to 100.

By analyzing plots in Figure 7, we can say that all the distributions have a "two-arcs" shape. We observed similar "two-arcs" distributions for all images so that our method of SGT generation produces quite stable results. It is possible to explain why these distributions have such a shape. This is related to the area of the representative square. For angles between -90° and 0°, this area reaches a maximum at extreme values of this range and it is minimal at -45°, that is, the plot of the area versus angle is concave. For angles between 0° and $+90^\circ$, this plot has the same shape because of symmetry with maxima at extreme values and a minimum at $+45^\circ$. As a result, when the area of the representative square is small, that is, it does not cover a significant part of the block it represents, the chance to miss a class label is high and the chance to add an extra label is low. When the area is large, the chance to miss is low and the chance to add is high. It is also easy to see that there is an inverse dependence between the occurrence of cases 1 and 4. That is, when the number of case 1 is large for a particular angle, that of case 4 is small for the same angle, and vice versa.

The average time for SGT generation, excluding the time spent on manual operations, was 0.38 s (Pentium-IV, CPU 3 GHz, 1 GB of RAM), while it took on average of 64.15 s to

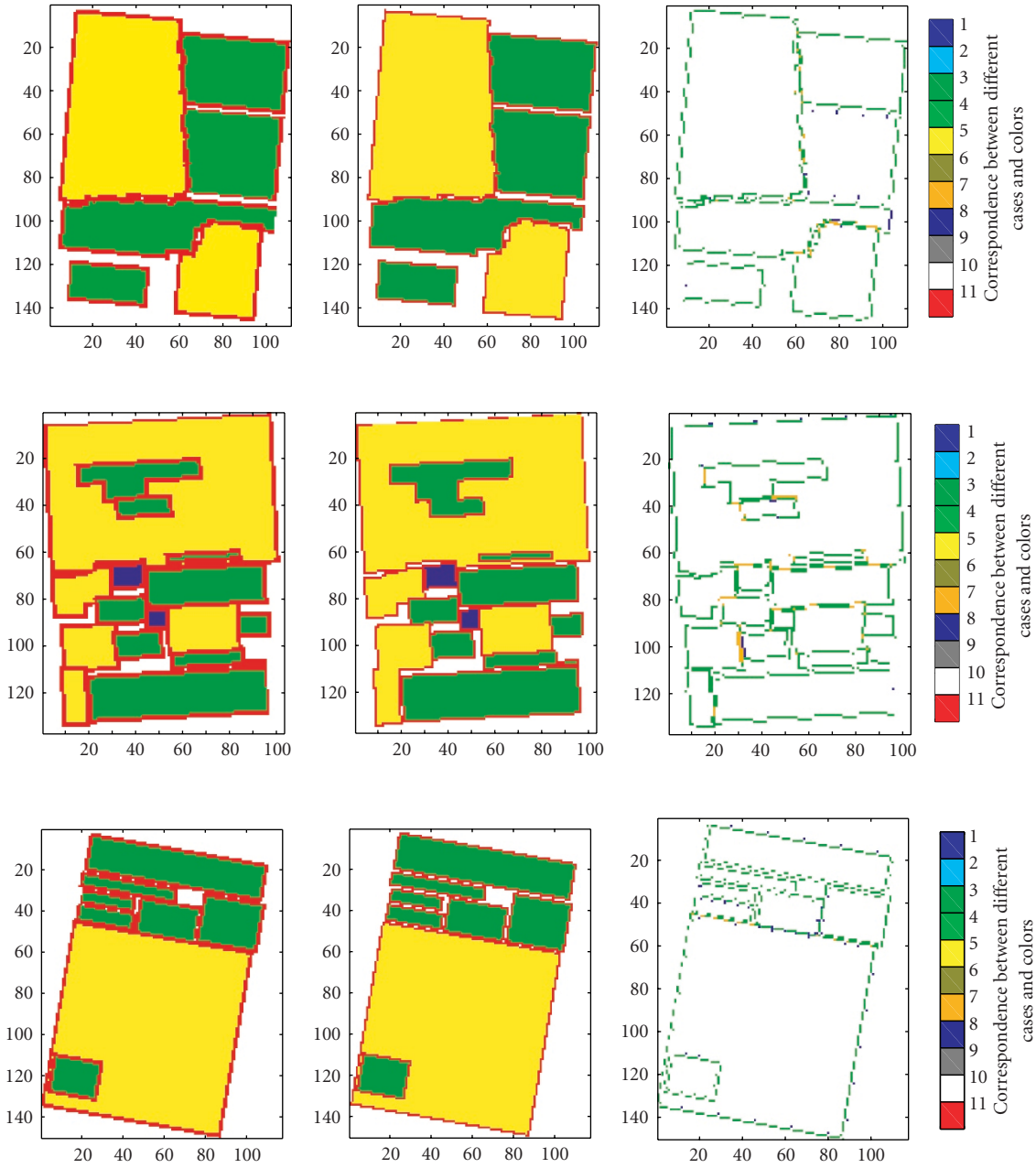

FIGURE 5: SGT, IGT, and their difference for the images in Figure 4. Skew angles are +5°, −3°, and +10°, respectively. In each row there are: SGT (left image), IGT (center image), and difference map (right image). For images displaying SGTs and IGTs, red color points to a composite label of a block, while yellow, green, and blue colors correspond to pure image, text, and graphics labels. In difference maps, each of the 11 cases is displayed by distinct color with colorbar shown under each map.

obtain an IGT. A major time-consuming operation was region detection that took up to several minutes per image, depending on the image complexity and the number of regions. The time for IGT generation greatly depended on the number of rectangular zones manually detected or/and resulting from polygon partitioning. As block sizes grew smaller, the accuracy of SGT generation for a certain image resolution also increased, as indicated by the verification procedure in Section 6.3, but at the expense of much higher processing time.

8. DISCUSSION AND CONCLUSION

After experimenting with GROTTO applied to complex advertisement and magazine page images, we can conclude that it is useful for quickly producing GTs when it is necessary to

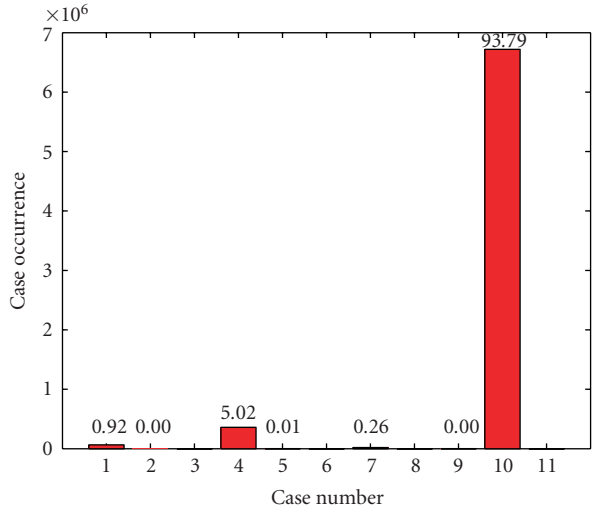

Figure 6: Accumulated statistics for all tested images. Figures over each bar correspond to frequency of occurrence for a given case in percent.

evaluate the skew tolerance of page segmentation algorithms. This opinion is based on the sufficiently high accuracy rate (more than 90%) and low missing rate (around 1%) as well as the small time (less than 0.5 s) needed for ground truth generation.

Though the current version of GROTTO does not yet allow users to compare a generated GT and results of a certain page segmentation algorithm, we describe possible scenarios where GROTTO can be useful.

Suppose that one has a certain page segmentation algorithm and a set of upright images. The task is to verify the skew-invariance of page segmentation carried out by this algorithm. In this case, a possible scenario can be as follows.

(1) Set N and skew angle α.
(2) Produce a UGT.
(3) Rotate the original upright image around its center by α and segment it into homogeneous regions, using a given page segmentation algorithm.
(4) Transform the segmentation results into the block-based format.
(5) Knowing α, generate an SGT and if necessary, IGT.
(6) Match the results of page segmentation and the SGT as well as the SGT and IGT to obtain descriptive statistics.

When the skew angle is unknown, it needs to be precomputed before GROTTO can be applied. Having determined this angle, the use of GROTTO is quite similar to that described above.

Now let us consider the task of image segmentation in general. Suppose that we have a collection of face images and the task is to separate faces from the background, that is, to detect faces in the images. To produce ground truths for this kind of images, an operator manually detects a rectangular area containing face and labels it. After that, a UGT is quickly produced, which is then compared with results of the face

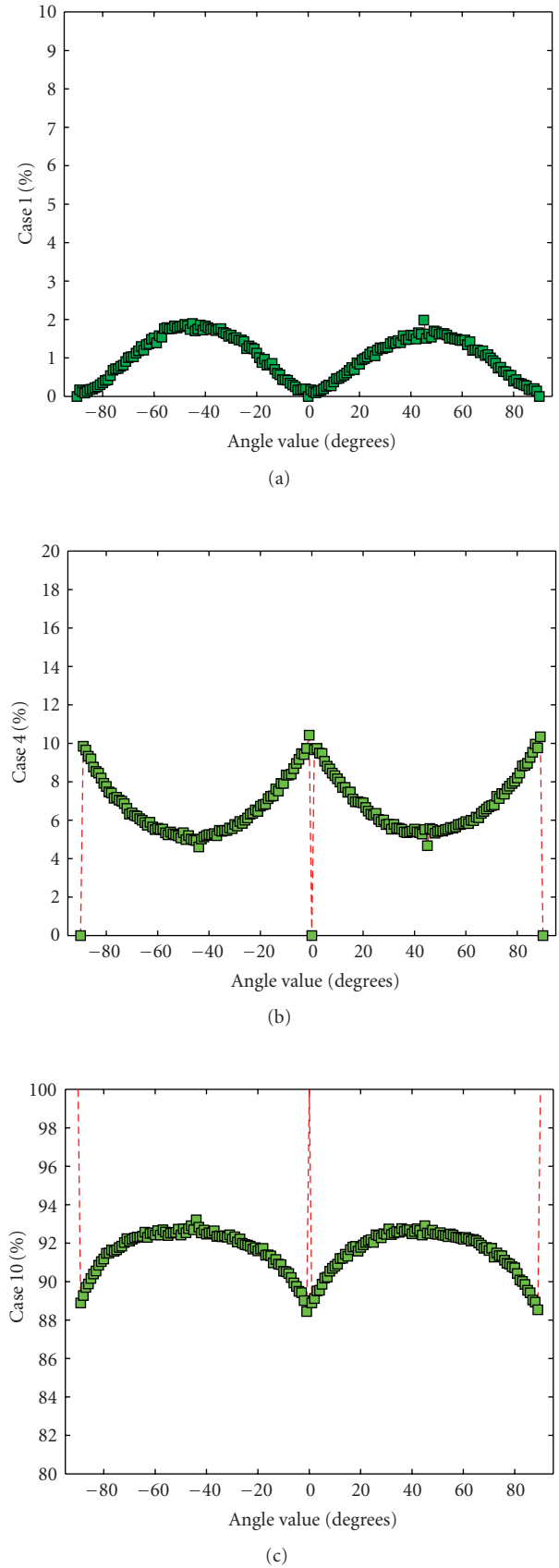

Figure 7: Typical occurrences of cases 1, 4, and 10 versus skew angle.

detection algorithm in question. This algorithm does not have to be block-based, but its result must be converted into the block-based representation used in GTs.

ACKNOWLEDGMENT

The authors are thankful to the reviewers for valuable comments that helped to significantly improve the paper.

REFERENCES

[1] A. Antonacopoulos, "Page segmentation using the description of the background," *Computer Vision and Image Understanding*, vol. 70, no. 3, pp. 350–369, 1998.

[2] A. Antonacopoulos and A. Brough, "Methodology for flexible and efficient analysis of the performance of page segmentation algorithms," in *Proc. 5th International Conference on Document Analysis and Recognition (ICDAR '99)*, pp. 451–454, Bangalore, India, September 1999.

[3] M. D. Garris, "Evaluating spatial correspondence of zones in document recognition systems," in *Proc. International Conference on Image Processing (ICIP '95)*, vol. 3, pp. 304–307, Washington, DC, USA, October 1995.

[4] M. D. Garris, S. A. Janet, and W. W. Klein, "Federal register document image database," in *Document Recognition and Retrieval VI*, D. P. Lopresti and J. Zhou, Eds., vol. 3651 of *Proceedings of SPIE*, pp. 97–108, San Jose, Calif, USA, January 1999.

[5] J. D. Hobby, "Matching document images with ground truth," *International Journal on Document Analysis and Recognition*, vol. 1, no. 1, pp. 52–61, 1998.

[6] J. J. Hull, "Performance evaluation for document analysis," *International Journal of Imaging Systems and Technology*, vol. 7, no. 4, pp. 357–362, 1996.

[7] J. Kanai, S. V. Rice, T. A. Nartker, and G. Nagy, "Automated evaluation of OCR zoning," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, no. 1, pp. 86–90, 1995.

[8] T. Kanungo and R. M. Haralick, "An automatic closed-loop methodology for generating character groundtruth for scanned documents," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 21, no. 2, pp. 179–183, 1999.

[9] O. Okun and M. Pietikäinen, "Automatic ground-truth generation for skew-tolerance evaluation of document layout analysis methods," in *Proc. 15th International Conference on Pattern Recognition (ICPR '00)*, vol. 4, pp. 376–379, Barcelona, Spain, September 2000.

[10] I. T. Phillips, J. Ha, R. M. Haralick, and D. Dori, "The implementation methodology for a CD-ROM English document database," in *Proc. 2nd International Conference on Document Analysis and Recognition (ICDAR '93)*, pp. 484–487, Tsukuba Science City, Japan, October 1993.

[11] S. Randriamasy and L. Vincent, "Benchmarking page segmentation algorithms," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '94)*, pp. 411–416, Seattle, Wash, USA, June 1994.

[12] B. A. Yanikoglu and L. Vincent, "Pink Panther: a complete environment for ground-truthing and benchmarking document page segmentation," *Pattern Recognition*, vol. 31, no. 9, pp. 1191–1204, 1998.

Oleg Okun received his Candidate of Sciences (Ph.D.) degree from the Institute of Engineering Cybernetics, Belarusian Academy of Sciences, in 1996. Since 1998, he joined the Machine Vision Group of Infotech Oulu, Finland. Currently, he is a Senior Scientist and Docent (Senior Lecturer) in the University of Oulu, Finland. His current research focuses on image processing and recognition, artificial intelligence, machine learning, data mining, and their applications, especially in bioinformatics. He has authored more than 50 papers in international journals and conference proceedings. He has also served on committees of several international conferences.

Matti Pietikäinen received his Doctor of Technology degree in electrical engineering from the University of Oulu, Finland, in 1982. In 1981, he established the Machine Vision Group at the University of Oulu. The research results of his group have been widely exploited in industry. Currently, he is a Professor of Information Engineering, the Scientific Director of Infotech Oulu research center, and the Leader of Machine Vision Group at the University of Oulu. From 1980 to 1981 and from 1984 to 1985, he visited the Computer Vision Laboratory at the University of Maryland, USA. His research interests are in machine vision and image analysis. His current research focuses on texture analysis, face image analysis, and machine vision for sensing and understanding human actions. He has authored about 180 papers in international journals, books, and conference proceedings, and about 100 other publications or reports. He is the Associate Editor of Pattern Recognition journal and was the Associate Editor of the IEEE Transactions on Pattern Analysis and Machine Intelligence from 2000 to 2005. He was the Chairman of the Pattern Recognition Society of Finland from 1989 to 1992. Since 1989, he has served as a Member of the governing board of the International Association for Pattern Recognition (IAPR) and became one of the founding fellows of the IAPR in 1994. He has also served on committees of several international conferences. He is a Senior Member of the IEEE and Vice-Chair of the IEEE, Finland Section.