Transactions on Rough Sets V, LNCS 4100 (2006), 63-101.

Matching 2D Image Segments with Genetic Algorithms and Approximation Spaces

Maciej Borkowski and James F. Peters

Department of Electrical and Computer Engineering,
University of Manitoba
Winnipeg, Manitoba R3T 5V6 Canada
{maciey,jfpeters}@ee.umanitoba.ca

Abstract. [1] This article introduces an approach to matching 2D image segments using approximation spaces. The rough set approach introduced by Zdzisław Pawlak provides a ground for concluding to what degree a particular set of similar image segments is a part of a set of image segments representing a norm or standard. The number of features (color difference and overlap between segments) typically used to solve the image segment matching problem is small. This means that there is not enough information to permit image segment matching with high accuracy. By contrast, many more features can be used in solving the image segment matching problem using a combination of evolutionary and rough set methods. Several different uses of a Darwinian form of a genetic algorithm (GA) are introduced as a means to partition large collections of image segments into blocks of similar image segments. After filtering, the output of a GA provides a basis for finding matching segments in the context of an approximation space. A coverage form of approximation space is presented in this article. Such an approximation space makes it possible to measure the the extent that a set of image segments representing a standard covers GA-produced blocks. The contribution of this article is the introduction of an approach to matching image segments in the context of an approximation space.

Keywords: Approximation space, coverage, genetic algorithm, image, 2D matching, rough sets, image segment.

1 Introduction

Considerable work on the application of rough set methods in image processing has been reported (see, e.g., [37,2,18,51,52]). This paper introduces an approach to matching image segments in the context of approximation spaces. The basic model for an approximation space was introduced by Pawlak in 1981 [30], elaborated in [28,32], generalized in [46,47,50], and applied in a number of ways (see, e.g., [36,38,39,48,11]). An approximation space serves as a

[1] Transactions on Rough Sets V, 2006, to appear.

formal counterpart of perception or observation [28], and provides a framework for approximate reasoning about vague concepts. Image segmentation (see, e.g.,[43,4,8,13,14,23,29,56,54]), and the image segment matching problem (see, e.g.,[12,55,42,53]) have been widely studied . The goal of an image-matching system is to match the segments from the two given images. Color and overlap are the two features of image segments that are commonly used to solve the matching problem. To achieve more accuracy in matching image segments, a combination of an evolutionary approach to finding sets of similar segments and approximation spaces are used. The evolutionar approach is realized with a genetic algorithm (GA) that partitions collections of image segments into sets of similar image segments. Filtering out GA-produced sets of image segments with the best match is carried out in the context of an approximation space. This approach makes it possible to solve the image segment matching problem with larger sets of features that yield more information about segments. This approach also results in more accurate matching of image segments. An overview of the 2D image segment matching method presented in this article is shown in Fig. 1.

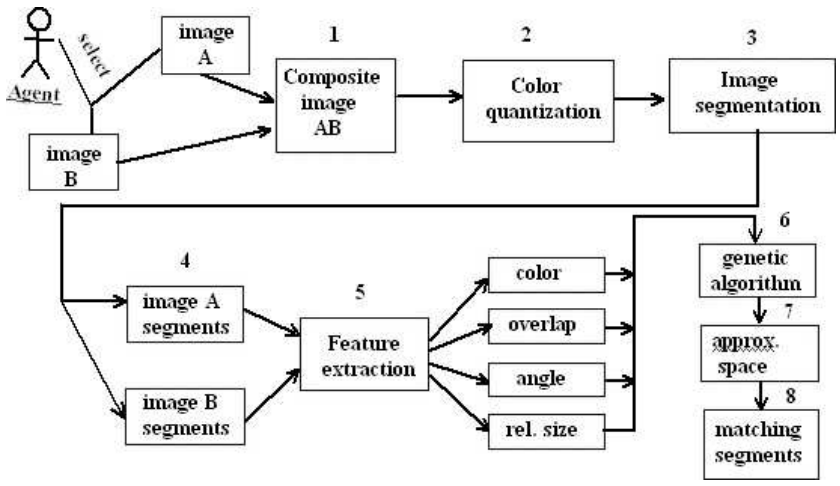

Fig. 1: 2D Image Segment Matching Steps

The matching process begins by forming a composite of a pair of images, then carrying out color quantization (step 2 in Fig. 1). After that, the quantized image is segmented, which results in a pair of segmented images. Next, feature values of image segment pairs are obtained in step 5 in Fig. 1. Then a GA is applied to a collection of image segment pairs, which are partitioned into sets. After eliminating non-disjoint sets of segment pairs, the coverage of the remaining sets of segment pairs is measured relative to a standard (norm), which is a set of image segment pairs that represent certain knowledge. The end result in step 8 of

Fig. 1 is a collection of best matching pairs of image segments. This is in keeping with the original view of approximation spaces as counterparts of perception (in this case, approximations provide a framework for visual perception). The contribution of this paper is the use of approximation spaces to solve the image segment matching problem.

This paper is organized as follows. A brief introduction to rough set theory is given in Sect. 2. Set approximation is presented in Sect. 2.1, and the structure of generalized approximation spaces is given in Sect. 2.2. The basic structure of a Darwinian form of genetic algorithm is presented in Sect. 3. Fundamental terminology for 2D digital images and classical 2D image processing techniques are presented in Sect. 4 and Sect. 5, respectively. Upper and lower approximation of sets of image segment pairs is described in Sect. 6 and Sect. 7, respectively. A detailed presentation of GAs for image processing is given in Sect. 8. An approach to matching image segments is presented in Sect. 9.

2 Basic Concepts About Rough Sets

This section briefly presents some fundamental concepts in rough set theory that provide a foundation for the image processing described in this article. In addition, a brief introduction to approximation spaces is also given, since approximation spaces are used to solve the 2D matching problem.

2.1 Rough Set Theory

The rough set approach introduced by Zdzisław Pawlak [31,32,33] provides a ground for concluding to what degree a set image segment pairs representing a standard cover a set of similar image segment pairs. The term "coverage" is used relative to the extent that a given set is contained in standard set. An overview of rough set theory and applications is given in [40,21]. For computational reasons, a syntactic representation of knowledge is provided by rough sets in the form of data tables. A data (information) table IS is represented by a pair (U, A), where U is a non-empty, finite set of elements and A is a non-empty, finite set of attributes (features), where $a : U \longrightarrow V_a$ for every $a \in A$. For each $B \subseteq A$, there is associated an equivalence relation $Ind_{IS}(B)$ such that $Ind_{IS}(B) = \{(x, x') \in U^2 | \forall a \in B, a(x) = a(x')\}$. Let $U/Ind_{IS}(B)$ denote a partition of U determined by B (i.e.,$U/Ind_{IS}(B)$ denotes the family of all equivalence classes of $Ind_{IS}(B)$), and let $B(x)$ denote a set of B-indiscernible elements containing x. $B(x)$ is called a block, which is in the partition $U/Ind_{IS}(B)$. For $X \subseteq U$, the sample X can be approximated from information contained in B by constructing a B-lower and B-upper approximation denoted by B_*X and B^*X, respectively, where $B_*X = \cup\{B(x)|B(x) \subseteq X\}$ and $B^*X = \cup\{B(x)|B(x) \cap X \neq \emptyset\}$. The B-lower approximation B_*X is a collection of blocks of sample elements that can be classified with full certainty as members of X using the knowledge represented by attributes in B. By contrast, the B-upper approximation B^*X is a collection of blocks of sample elements representing both certain and possibly uncertain

knowledge about X. Whenever B_*X is a proper subset of B^*X, i.e., $B_*X \subset B^*X$, the sample X has been classified imperfectly, and is considered a rough set.

2.2 Approximation Spaces

This section gives a brief introduction to approximation spaces. The basic model for an approximation space was introduced by Pawlak in 1981 [30], elaborated in [28,32], generalized in [46,47,50], and applied in a number of ways (see, e.g., [36,38,48,11]). An approximation space serves as a formal counterpart of perception or observation [28], and provides a framework for approximate reasoning about vague concepts.

A very detailed introduction to approximation spaces considered in the context of rough sets is presented in [40]. The classical definition of an approximation space given by Zdzisław Pawlak in [30,32] is represented as a pair (U, Ind), where the indiscernibility relation Ind is defined on a universe of objects U (see, e.g., [44]). As a result, any subset X of U has an approximate characterization in an approximation space. A generalized approximation space was introduced by Skowron and Stepaniuk in [46,47,50]. A *generalized approximation space* is a system $GAS = (\ U,\ N,\ \nu\ )$ where

- U is a non-empty set of objects, and $\mathcal{P}(U)$ is the powerset of U,
- $N : U \rightarrow \mathcal{P}(U)$ is a neighborhood function,
- $\nu : \mathcal{P}(\mathrm{U})\ \mathrm{x}\ \mathcal{P}(\mathrm{U}) \rightarrow [0, 1]$ is an overlap function.

A set $X \subseteq U$ in a GAS if, and only if X is the union of some values of the neighborhood function. In effect, the uncertainty function N defines for every object x a set of similarly defined objects [45]. That is, N defines a neighborhood of every sample element x belonging to the universe U (see, e.g., [35]). Generally, N can be created by placing constraints on the value sets of attributes (see, e.g., [40]) as in (1).

$$y \in N(x) \Leftrightarrow \max_a \{ dist_a(a(x), a(y)) \} \leq \epsilon. \tag{1}$$

where $dist_a$ is a metric on the value set of a and ϵ represents a threshold [40]. Specifically, any information system $IS = (U, A)$ defines for any $B \subseteq A$ a parameterized approximation space $AS_B = (U, N_B, \nu)$, where $N_B = B(x)$, a B-indiscernibility class in the partition of U [45]. The rough inclusion function ν computes the degree of overlap between two subsets of U. Let $\mathcal{P}(U)$ denote the powerset of U. The overlap function ν is commonly defined as standard rough inclusion (SRI) $\nu : \mathcal{P}(\mathrm{U})\ \mathrm{x}\ \mathcal{P}(\mathrm{U}) \rightarrow [0, 1]$ as defined in (2).

$$\nu_{SRI}(X, Y) = \begin{cases} \frac{|X \cap Y|}{|X|}, & \text{if X } \neq \emptyset, \\ 1 \quad , & \text{if X} = \emptyset. \end{cases} \tag{2}$$

for any $X, Y \subseteq U$, where it is understood that the first term is the smaller of the two sets. The result is that $\nu_{SRI}(X, Y)$ represents the proportion of X that is

"included" in Y. However, we are interested in the larger of the two sets (assume that the $card(Y) \geq card(X)$) because we want to see how well Y "covers" X, where Y represents a standard for evaluating sets of similar image segments. Standard rough coverage (SRC) ν_{SRC} can be defined as in (3).

$$\nu_{SRC}(X,Y) = \begin{cases} \frac{|X \cap Y|}{|Y|}, & \text{if Y} \neq \emptyset, \\ 1 \quad , & \text{if Y} = \emptyset. \end{cases} \tag{3}$$

In other words, $\nu_{SRC}(X,Y)$ returns the degree that Y covers X. In the case where $X = Y$, then $\nu_{SRC}(X,Y) = 1$. The minimum coverage value $\nu_{SRC}(X, Y) = 0$ is obtained when $X \cap Y = \emptyset$ (i.e., X and Y have no elements in common).

3 Genetic Algorithms

Evolution has been characterized as an optimization process [9,19,25]. Darwin observed "organs of extreme perfection" that have evolved [5]. Genetic algorithms (GAs) belong to a class of evolutionary algorithms introduced by John Holland in 1975 [19] as a means of studying evolving populations. A GA has three basic features:

- **Representation**: each population member has a representation,
- **Method of Selection**: fitness of each population member is evaluated,
- **Method of Variation (Crossover)**: create new population member by combining the best features from pairs of highly fit individuals.

Crossover is the fundamental operation used in classical genetic algorithms. Mutation is another method used in GAs to induce variations in the genes of a chromosome representing a population member. The basic steps in a genetic algorithm are described as follows. Let $P(t)$ denote an initial population of individual structures, each with an initial fitness at time t. Then an iteration begins. Individuals in $P(t)$ are selected for mating and copied to a mating buffer $C(t)$ at time step t. Combine individuals in $C(t)$ to form a new mating buffer $C'(t)$. Construct a new population P_{t+1} from P_t and $C'(t)$. A desired fitness is used as a stopping criterion for the iteration in a GA. A representation of a very basic GA that uses only the crossover operation is given in Alg. 1.

GAs have proven to be useful in searching for matching segments in pairs of images (see Sect. 8). In preparation for the GA approach to matching images, some basic terminology (see Sect. 4), rudiments of classical 2D image processing (see Sect. 5), and image matching using rough set methods (e.g., upper approximation approach in Sect. 6 and lower approximation approach in Sect. 7) are presented.

1: Basic GA

Input : population P_t, mating pool C_t
Output: evolved population P_T at time T
$t = 0$;
Initialize fitness of members of P_t;
while (*Termination condition not satisfied*) **do**
  $t = t + 1$;
  Construct mating pool C_t from P_{t-1};
  Crossover structures in C_t to construct new mating pool C'_t;
  Evaluate fitness of individuals in C'_t;
  Construct new population P_t from P_{t-1} and C'_t;
end

4 Classical 2D Matching Terminology

This section gives an introduction to the basic definitions of technical terms associated with the classical approach to 2D matching images.

Definition 1. ([12,24,17]) **Pixel.** *A pixel (also referred to as a* image element*,* picture element*, or pel) is an element of a digital image.*

Definition 2. ([12]) **Color.** *A color of a pixel is a mapping from a space of all colors perceived by humans into a finite set of integer numbers grouped into three components. Each component Red, Green and Blue is represented by a number from 0 to 255. The total number of different colors represented by a pixel is 16,777,216.*

Definition 3. ([12]) **Grayscale.** *Grayscale represents a subset of RGB space, where all components have equal values, eg. Red = Green = Blue. There are only 256 such combinations and therefore the grayscale values can be represented only by one number from the range 0 to 255.*

In a digital image, a pixel (short for *picture element*) is an element that has a numerical value that represents a grayscale or RGB intensity value. Pixels are part of what are known as 4-neighborhoods, which provide a basis for identifying image segments.

Definition 4. ([12,24]) **4-Neighborhood.** *A pixel p with coordinates (x,y) has 4 neighbors (2 vertical and 2 horizontal neighbors) at coordinates*

$$(x+1, y), (x-1, y), (x, y+1), (x, y-1)$$

Definition 5. ([12]) **Pixel Membership.** *Pixel* $p_1 = (x_1, y_1)$ *belongs to a 4-neighborhood of pixel* $p_2 = (x_2, y_2)$ *if and only if exactly one coordinate of* p_2 *differs from the corresponding coordinate of* p_1*. This difference must be equal to 1.*

Definition 6. Segment. *A segment is a collection of 4-neighborhood connected pixels, which have the same color.*

The process of matching segments described in this article is based on four parameters described in the section 5.3. These parameters are *degree of overlap* between segments, *angle of rotation* between segments, *distance between mean colors* of segments and *ratio of cardinalities* of both segments. A combination of a genetic algorithm and rough set-based post processing is used to combine the information from all four parameters to find the best matches between the segments.

The problem of finding the match between image segments is not trivial. The four parameters required for matching image segments sometimes contain contradictory information about the quality of match. Thus it is impossible to find proper matches using only one or two of these parameters. The simplest approach is to find the matches with the smallest distance in the space defined by the four parameters. This space is denoted by Ω and consists of vectors where each coordinate value is the difference of some parameter values.

Definition 7. Image Segment Parameter Space. *Define space Ω to be a subspace of $\mathbb{R}^4$ such that*

$$\Omega = \mathbb{C} \times \mathbb{O} \times \mathbb{A} \times \mathbb{RC} \subseteq \mathbb{R}^4.$$

where $\mathbb{C}$, $\mathbb{O}$, $\mathbb{A}$ and $\mathbb{RC}$ denote domains of the four parameters' values, i.e. the distance between mean colors of segments, degree of overlap between segments, the angle of rotation between segments and ratio of cardinalities of both segments, respectively.

The match between two points $s,\ t \in \Omega$ can be calculated as a weighted distance between their parameters' values.

Definition 8. Distance in Image Segment Parameter Space. *The distance between two vectors s and t such that $s,\ t \in \Omega$ is defined to be a distance between these points in the space Ω weighted by the vector $\omega = (C, O, A, Rc)$.*

$$\|s - t\|_\omega = \sqrt{C \cdot (s_1 - t_1)^2 + O \cdot (s_2 - t_2)^2 + A \cdot (s_3 - t_3)^2 + Rc \cdot (s_4 - t_4)^2}.$$

where $s = (s_1, s_2, s_3, s_4)$ and $t = (t_1, t_2, t_3, t_4)$.

Here, the weight vector (C, O, A, Rc) denotes the importance of each parameter. Each such vector and an ideal vector ζ define a measure of the quality of a match.

Definition 9. Measure of Quality. *A measure of quality of a match between two segments s parametrized by the vector $\omega = (C, O, A, Rc)$ and the ideal solution ξ is given by the distance between points s and ξ in Ω space.*

$$Q_{\omega,\xi}(s) = \|s - \xi\|_\omega .$$

The problem with the Def. 9 is with defining the ideal solution ξ. The first two parameters in ω can be defined, where the difference in color $C = 0$ and the overlap between two segments $O = 1$. The remaining two parameters (A and Rc) in ω can be defined only with respect to some set of matches. It does not make sense to define the ideal angle of rotation between segments, since it depends on the images and can be different for any pair of images. Therefore, the ideal solution can be defined only in the first two positions. In order to make the remaining two parameters not influential, the ω vector must contain zeros in the third and fourth position. As a result, the ideal vector is defined as

$$\zeta = (0, 1, 0, 0)\,, \omega = (x, y, 0, 0)\,.$$

where $x, y \in \mathbb{R}$. Unfortunately, this solution uses only two parameters instead of four. This can lead to wrong classification as shown in the Fig. 9 or Fig. 12 ($\circ$ denotes the correct match and $+$ denotes the closest match using $Q_{\omega,\xi}$ measure).

An algorithm which uses all four image segment features should generate a set of possible good matches. A genetic algorithm (GA) is the example of such an algorithm. It is possible to design a genetic algorithm (see, e.g., GA Alg. 6 and Alg. 7) which orders image segments. This form of GA can be considered an image segment matching algorithm, which uses all four features in the image segment feature space. The basis for this form of image segment matching algorithm is explained in Sect. 5.

5 2D image processing

This section introduces the basic concepts that will be used in a GA-based image segment matching algorithm. At the 2D image processing level, the information available about digital images comes from the locations of pixels and their RGB values. The main goal of 2D image matching is to match all pixels from one image with corresponding pixels from a second image. This operation is known as *image registration* [7], [57], [3].

5.1 Image Segmentation

Quantization has been defined as a process of converting analog signal to digital signal [10]. A quantizer is defined as a mapping from an uncountably infinite space of values into a finite set of *output levels*. In proposed system the source signal is digital image. Its domain is a finite set (pixels) of integer numbers (colors). Since colors are represented by three components, namely *Red*, *Green* and *Bule*, and each component is described by one byte, the input signal is already finite. Thus, the term quantization is rather used as mapping from a finite set of numbers to another finite set of numbers, where the cardinality of the destination set is smaller than the source set. In what follows, the Lloyd quantization algorithm [10] has been used, see Alg. 2.

2: The Lloyd Algorithm [10] (alg. Q_n)

Input: image I, required number of colors n
Output: optimal codebook with n entires C_{opt}

Initialize codebook C_1 with n entries randomly, set $m = 1$
repeat
  Based on codebook C_m and using nearest neighbour condition partition the image I into the quantization cells R_m
  Using centroid condition find optimal codebook C_{m+1} for cells R_m
  Set $m = m + 1$
until *distortion caused by* C_m *small enough*
Set $C_{opt} = C_m$

A quantization mapping is usually expressed by a codebook. A codebook is a set of n colors which are used to represent the original image. The mapping is performed by replacing the original color with the closest color from the codebook. The optimal codebook of size n is the set of colors which minimizes the distortion caused by the codebook. Here, the distortion is calculated as the squared difference between all components of the original color and its nearest neighbor from the codebook.

The Lloyd algorithm consists of main two steps, which are repeated until the distortion caused by the codebook is small enough. The first step is the partitioning of the input image based on the current codebook. The partitioning is performed using nearest neighbour condition, e.g. each pixel is assigned to the cell closest to the color of given pixel. In the second step, a new codebook is created based on the partitioning from the first step. Each codebook entry is replaced by a centroid of all colors of pixels from the corresponding cell.

Color quantization is used as an aid in image segmentation. It works only in the color space. The actual segmentation needs to take into account also a spacial information, namely the position of pixels. Only the combination of color and spatial information leads to identification of image segments. The averaging step fills the gap regarding the use of spatial information. It's only purpose is to average information carried out by pixels representing similar colors. The term *similar* is in this context precisely defined. Assume, that an original image denoted by I_o is given. First, quantization reducing number of colors to n_1 is performed. This step is denoted by formula 4 to obtain a quantized image denoted by $I_{q_{n_1}}$(symbol Q represents the algorithm 2, where I_o is the input image I and n_1 is the required number of colors n).

$$I_o \xrightarrow{Q_{n_1}} I_{q_{n_1}} \tag{4}$$

As a result of quantization, the quantized image $I_{q_{n_1}}$ contains only n_1 colors. In the next step, the information from $I_{q_{n_1}}$ image is used to average the colors among all pixels, which are *connected*.

In quantized image, regions of pixels of the same color can be identified. These regions create segments. To each such segment is assigned a color, which

is an average of all original colors from pixels belonging to this region. This step is denoted by the formula in (5), see also Alg. 3.

$$I_{q_{n_1}} \xrightarrow{Av_{I_o}} I_{Av_{n_1}} \tag{5}$$

The image $I_{Av_{n_1}}$ resulting from (5) has more than n_1 colors, where pixels are grouped into segments. This procedure, namely steps defined in (4) and (5), is repeated. The number of colors gradually decreases in consecutive iterations so that the creation of segments can be observed.

3: The Spacial Color Averaging (alg. $Av_{I_{s_n}}$)

Input: image I
Output: averaged image I_A

Mark all pixels from I as *not processed*
foreach *not processed pixel* p *in* I **do**
  Find segment $S(p)$ in I containing pixel p
  Assign to each corresponding pixel in I_A from $S(p)$ an average color of all pixels from $S(p)$
  Set all pixels from $S(p)$ as *processed*
end

The unwanted effect of the algorithm defined this way is that if a segment is created at some step, there are no chances to change it in consecutive steps. In other words, the first quantization plays a crucial role in entire process. In addition, the resulting image still contains a lot of details (even though the number of colors was reduced). An example of such image processed using seven iterations described by the succession of mappings in (6), where numbers n_i for $i = 1, 2, ..., 6$[2] are 256, 64, 32, 16, 12, 8 and 4 are shown in left side of figure 2.

$$I_{s_{n_{i-1}}} \xrightarrow{Q_{n_i}} I_{q_{n_i}} \xrightarrow{Av_{I_{s_{n_{i-1}}}}} I_{Av_{n_i}} \tag{6}$$

To make the entire segmentation process more robust and force the creation of bigger segments, one extra step for each stage defined by (6) is added. That is, after the colors are recreated from the original image, a 3 by 3 median filter is used. This causes almost uniform areas to blur even more and allows edges of neighboring segments to overlap. As a result, all small details from the image are lost, and big uniform segments are formed instead. The final formula describing one step of this iterated algorithm is shown in (7).

$$I_{s_{n_{i-1}}} \xrightarrow{Q_{n_i}} I_{q_{n_i}} \xrightarrow{Av_{I_{s_{n_{i-1}}}}} I_{Av_{n_i}} \xrightarrow{M_{3x3}} I_{s_{n_i}} \tag{7}$$

[2] For $i = 0$ it is assumed that $I_{s_{n_0}} = I_o$, and after each iteration $I_{s_{n_{i-1}}} = I_{Av_{n_{i-1}}}$.

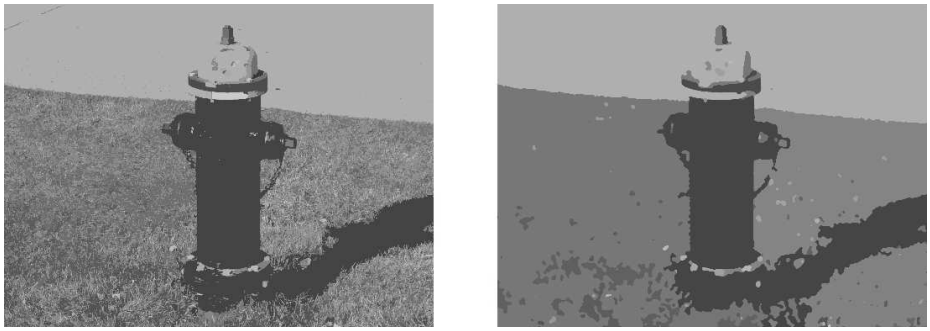

Fig. 2: Hydrant image after 7 iterations of (6) (left) and (7) (right)

The M_{3x3} symbol denotes the median filter which is applied to each pixel from an input image. The median filter is applied to 3 by 3 neighborhood of given pixel $p(x, y)$.

$$M_{3x3}(p) = median\{p(x-1, y-1), p(x, y-1), p(x+1, y-1), p(x-1, y),$$

$$p(x, y), p(x+1, y), p(x-1, y+1), p(x, y+1), p(x+1, y+1)\} \tag{8}$$

In order to find the median, all pixels are sorted by their color value and the one in the middle (e.g. at the 5-th place) is chosen.

The right side of figure 2 shows the result of applying seven-step iterative algorithm (with the same values as in previous example), where each step is described by (7). There are still many small segments, but comparing with the corresponding image, where the median filter was not used, their number was greatly reduced.

Figure 3 shows all steps of applying formula (7). The image in the first row and leftmost column is the original image. The second image in the first row, is a result of 8-bit quantization. The third image shows the result of applying 3 by 3 median filter. In the second row, the second iteration is shown. Leftmost image shows the result of averaging colors in segments from previous step. Middle image shows result of 5-bit quantization and rightmost image shows result of applying 3 by 3 median filter. The remaining five rows are organized the same way as the second row.

5.2 Segment Selection

At this stage it is assumed that a digital image is divided into segments. To increase the chances of identifying the same segments in both images, image quantization is performed on one large image, which is a composite of two individual images placed next to the other. After segmentation of the composite image, the two images in the composite are extracted and the analysis continues on the separate images. In this step, only some segments from all of the segments

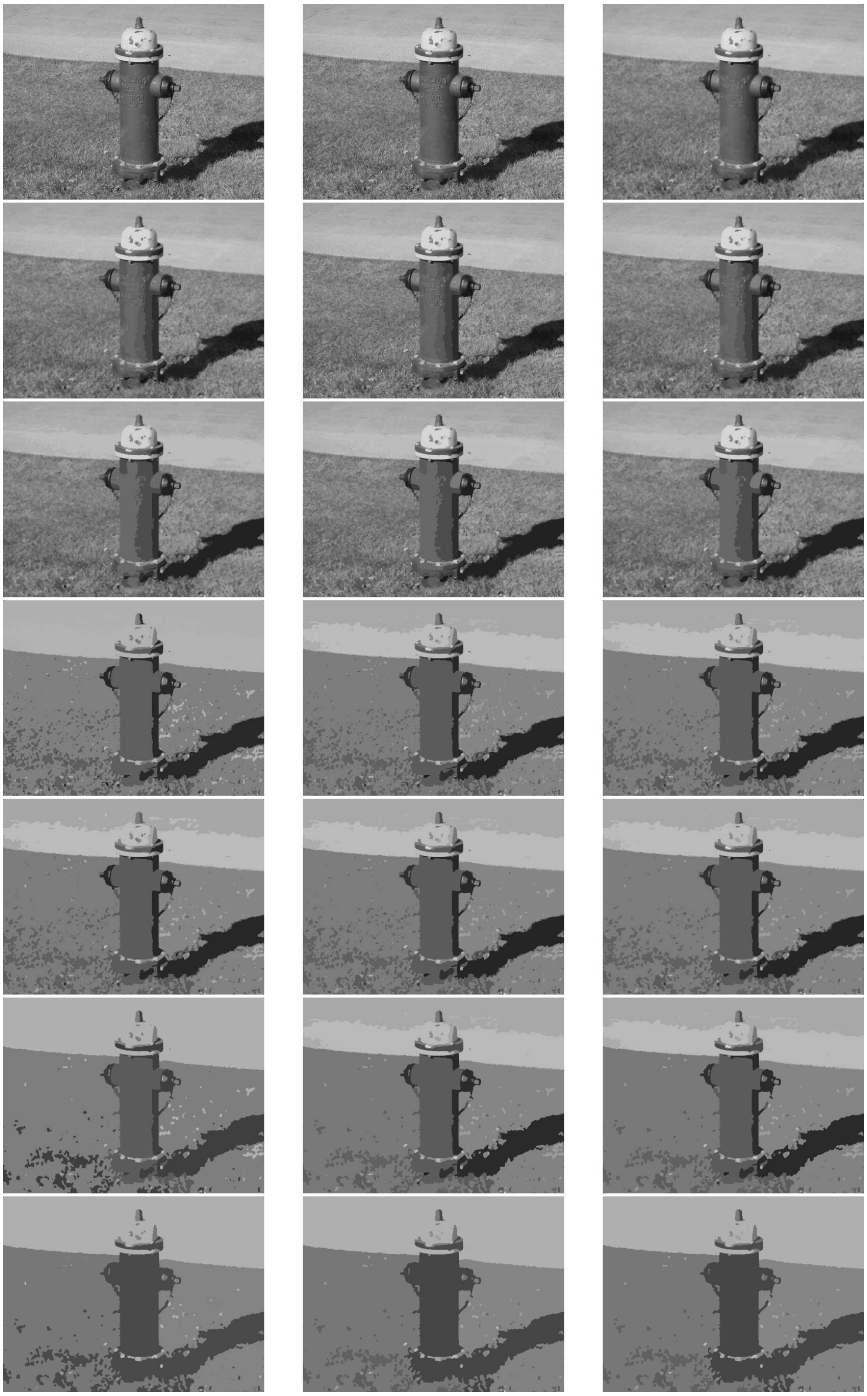

Fig. 3: Quantized images obtained by iterating (7)

created so far are selected. The reason for this is the high number of segments and their shape. During a procedure to match shapes (described in section 8), all segments from both images are matched using a GA search for segments, which satisfy specific criteria. A GA is used because there is a need to work with the smallest number of segments possible. In addition, the matching algorithm requires that each segment satisfy some additional properties.

- **Lower bound on segment size**: avoid too few segments,
- **Upper bound on segment size**: avoid too many segments,
- **Convexity factor**: avoid perspective distortion.

Segment size is measured by the number of pixels belonging to a given segment. A lower bound on segment size is need for the following reasons. First, if there are not too many pixels, for example less than 10, the pixels can describe only a small number of distinctive shapes. Matching of such shapes is very difficult, since such a small number of pixels does not have enough power to uniquely represent fairly distinctive shape. Second, if all tiny segments are considered, the search space for matching segments becomes too large. There is a small chance that these tiny shapes can be uniquely matched.

The explanation for upper bound of segment size is motivated by characteristics of most images. Usually, images contain large areas of a solid color. For outdoor images it can be the sky, for indoor images it can be the walls of the room the image is shot in. These solid areas function as a background for the given scene. The shape of the background is not unique and it changes due to perspective transformation. By setting upper bound for segment size all segments, which can be part of the background are filtered out. For this research, this limit is set to be 30% of the entire image area.

The last constraint in matching image segments is the convexity factor, which deals with perspective distortion and filters out shapes, which are difficult to match. To get a deeper insight into this problem, consider what detected segments represent and how they differ from image to image. Each image is a 2D representation of a 3D scene. Similarly, segments which are flat represent 3D objects. The transformation from 3D space into 2D images flattens objects in a sense that the information from different parts of an object is represented in a small area. For example, consider the silhouette of a tripod. Given one segment representing an entire tripod, each leg is separated from the other legs by some background pixels (at least at the bottom of the tripod). Depending on the angle of the camera, some legs can be quite close to each other. The shape of a tripod changes dramatically with the change of view angle. Attempts to match such shapes should be avoided. This example shows that objects which are spread in all three dimensions are separated by some pixels not belonging to the object. This condition is expressed for flat images in terms of convexity. A segment S is *convex* if each point from a straight line connecting any two points in S also belongs to S [12].

Definition 10. Convexity factor. *The convexity factor for a segment S is a number $C_f(S)$ between 0 and 1 specifying how many lines between all combinations of points from segment S lie entirely inside the segment S.*

$$C_f(S) = \frac{\#\ lines\ entirely\ inside\ S}{\#\ all\ possible\ lines}.$$

To filter out segments which are potentially difficult to match, a threshold for a convexity factor is set and only segments greater than the threshold are selected. Based on experiments, a threshold of 0.5 has worked well.
Implementation of an algorithm used to calculate a convexity factor from the definition requires n^2 lines to be tested, where n is the number of segment pixels. In order to speed up the calculations the estimation is performed. The estimation process is applied on two levels. First, not all combinations of points are checked. Instead, randomly selected $50 \cdot n$ pairs of points are chosen. Second, instead of checking if an entire line is contained within a segment, only checks for 7 points are performed: middle point of a line, one fourth, three fourths and remaining multiples of 1/8, namely, $1/8^{th}$, $3/8^{th}$, $5/8^{th}$ and $7/8^{th}$ of the line. In order to calculate each of these points only as few as two additions and two divisions are required, which makes this algorithm very fast with a complexity of $O(n)$.

5.3 Feature Generation

In this section, the following four features used for matching segments are elaborated.

- **degree of overlap** between segments,
- **angle of rotation** between segments,
- **distance between mean colors** of segments,
- **ratio of cardinalities** of segment pairs.

This section describes how these features are extracted from two sets of segments (one set of segments for each image). Sect. 8 elaborates about how the actual matching is performed using these features. First, recall that segments are only two dimensional representations of three dimensional objects. Due to the change of view angle, segments undergo transformation, which alters their shape. Therefore, simple comparison of shapes is not enough to pair segments.

Before the matching can start, values for the four features for all combinations of segments from both sets are generated. First two features are generated by an algorithm which tries to find the biggest overlap between two segments.

Overlap. This parameter measures the overlap between two segments. To calculate the overlap, two segments are plotted in one image using the same color (one_seg denotes the number of pixels belong to one segment). Pixels which belong to both segments are denoted by a second color (two_seg denotes the number of pixels belonging to both segments). A measure of the overlap between a pair of image segments is computed using Eq. 9.

$$overlap = e^{-\frac{|P_{one\_seg}|}{|P_{two\_seg}|}}. \quad (9)$$

where P_i denotes pixels of i-th color. For $|P_{one\_seg}| \neq 0$ and $|P_{two\_seg}| = 0$ it is assumed that $\frac{|P_{one\_seg}|}{|P_{two\_seg}|} = \infty$. In other words, *overlap* measures how well one segment matches the other. The minimum value for *overlap* is zero. In this case, the number of pixels belonging to both segments is equal to zero, which means that the segments do not intersect. A maximum $overlap = 1$ occurs when both segments have the same same shape and are located at the same position. In the case where one_seg $= 0$, $e^0 = 1$. For all other cases, $overlap\ \epsilon(0, 1)$.

The formula 9 was chosen for two reasons. First, it rescales the range of overlap values from $(0, \infty)$ to $(0, 1)$ interval. A finite interval is easier to handle than the infinite one. Second, the exponential function compresses the output of the original $\frac{|P_{one\_seg}|}{|P_{two\_seg}|}$ function in the range where P_{one_seg} is much greater than P_{two_seg} (for example, where $\frac{|P_{two\_seg}|}{|P_{one\_seg}|} < 2.5$, see figure 4). The absolute value of the slope of the overlap function from Eq. 9 is much smaller than the slope of the $\frac{|P_{one\_seg}|}{|P_{two\_seg}|}$ function. This allows for easier comparison of overlap values in the last stage of overlapping, e.g. when there is much more common pixels than not matched ones. The smaller slope means that small changes in the ratios of common/not matched pixels will not cause huge changes of the overlap function.

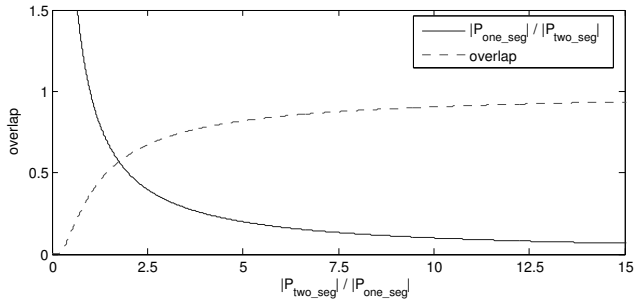

Fig. 4: The result of applying exponential into overlap function

Fig. 5 illustrates best overlap. For better visualization, the two segments are plotted using different colors. The intersection is denoted by the brightest shade of gray. The lefthand side of Fig. 5 shows both segments with their original rotation, scale and position. The righthand side of Fig. 5 shows the two segments with maximum $overlap = 0.85$. Observe that the area occupied by only one segment has been significantly decreased in comparison with the original configuration.

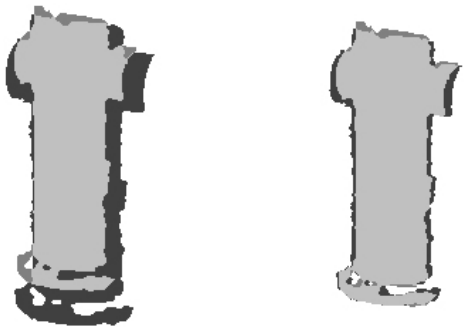

Fig. 5: Preliminary overlap of 2 segments(left), and best overlap (right)

Example 1. The figure 6 shows three sample steps out of many steps performed during segment matching of the figure 2. These three steps explain the idea behind the overlap formula introduced in the Eq. 9.

The first image in the figure 6 shows the first stage when the two segments do not have any pixels in common. The area of the first image is 41083 pixels and the area of the second one is 36447 pixels. Since there are no common pixels $|P_{one\_seg}| = 41083 + 36447 = 77530$ and $|P_{two\_seg}| = 0$. From the assumption for $|P_{one\_seg}| \neq 0$ and $|P_{two\_seg}| = 0$ the fraction $\frac{|P_{one\_seg}|}{|P_{two\_seg}|} = \infty$. Because of the formula 9 the overlap is not equal to infinity but a finite number $e^{-\infty} = 0$. It is easier to deal with finite numbers than infinite.

The second image shows one of the intermediate steps, where the two segment have a lot of pixels in common, but also a lot of non overlapping pixels. The area of the first segment, which does not intersect with the second one is 18219 pixels. The area of the second segment, which dos not intersect with the first one is 13583 pixels. The area of overlap between these segments is 22864 pixels. Therefore, $|P_{one\_seg}| = 18219 + 13583 = 31802$ and $|P_{two\_seg}| = 22864$. The overlap is equal to $overlap = e^{-\frac{|P_{one\_seg}|}{|P_{two\_seg}|}} = e^{-\frac{31802}{22864}} = e^{-1.391} = 0.248$.

The third image shows the final result of search for the best overlap. The first segment, was being translated, rotated and rescaled to maximize the overlap function. In this position the overlap is maximum. Here, the $|P_{one\_seg}| = 5540 + 1097 = 6637$ and $|P_{two\_seg}| = 35350$. Thus, $overlap = e^{-\frac{6637}{35350}} = e^{-0.187} = 0.828$.

The figure 7 shows the entire process of finding the best overlap for segments from figure 2. The horizontal axis denotes the iteration number. In each iteration the position, rotation and scale for the first segment is altered to minimize the overlap function. In the left part of the figure 7 a ratio $\frac{|P_{one\_seg}|}{|P_{two\_seg}|}$ is plotted. For the first several iterations it takes on high values compared to the end of matching process. In fact, the ending of the matching process is more important, since it can detect small differences in segments' shapes. Therefore, the overlap function, showed in the right part of the figure 7, is more sensitive to changes in the second half of the matching process. When two segments do not overlap

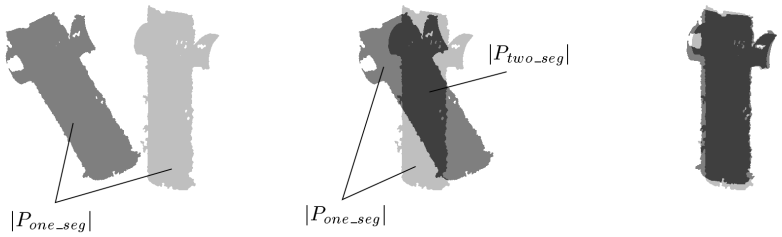

Fig. 6: Three sample steps of segment matching.

significantly, the overlap function is close to zero. Only after there is a lot of overlap between segments, see the middle image from the figure 6, the overlap function changes more rapidly to emphasis the change in overlap.

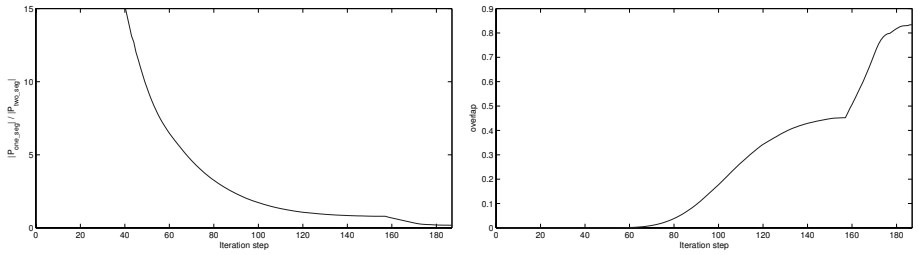

Fig. 7: The process of finding the best overlap for $\frac{|P_{one\_seg}|}{|P_{two\_seg}|}$ and *overlap* parameter.

Angle. The angle of rotation is the relative angle which one segment must be rotated to maximize the overlap between two segments.

Color. The previous two parameters (*overlap* and *angle*) dealt with geometrical properties of segments. The *color* parameter takes into account the color of a segment. Recall that all pixels from one segment are assigned the mean value of the colors from the original image. $C_{diff}(i,j)$ denotes the distance between the RGB vectors of colors for a pair of segments. If the *i-th* segment's color is denoted by $C_i = (R_i, G_i, B_i)$ and *j-th* segment's color is denoted by $C_j = (R_j, G_j, B_j)$, then $C_{diff}(i,j)$ is defined by Eq. 10.

$$C_{diff}(i,j) = |C_i - C_j| = \sqrt{(R_i - R_j)^2 + (G_i - G_j)^2 + (B_i - B_j)^2}. \qquad (10)$$

Ratio of Cardinalities. The $RatioofCardinalities$ parameter is a measure of the relative size of a pair of segments. Let S_i, S_j denote sets of 4-neighborhood connected pixels for image segment i and j, respectively. Further, let RC denote a measure of the *Ratio of Cardinalities*, which is defined by Eq. 11.

$$RC(i,j) = \frac{|S_i|}{|S_j|}. \tag{11}$$

5.4 Exhaustive Feature Matching

The goal of the matching algorithm is to produce a set of segment pairs so that each segment of a pair belongs to a different images. Given n_1 segments identified in the first image and n_2 segments identified in the second image, the total number of possible pairs is $n_1 \cdot n_2$. From $n_1 \cdot n_2$ matches only small number corresponds to the correct matches. In order to allow for grouping of several image segment matches a hypothesis is introduced.

The central notion of the searching algorithm is a *hypothesis*. A hypothesis it is a set of image segment matches. A hypothesis is created by assuming that all four parameters for correctly matched segments are in the same range of values. In other words, a hypothesis identifies a set of paired segments.

The algorithm searches through the space of matches using hypotheses to validate each pair. It can be characterized by the average rotation angle between segments and average ratio of cardinalities of both segments, where the average is taken with respect to all pairs in the hypothesis. The rotation angle between segments corresponds to the rotation between images and the ratio of cardinalities corresponds to the difference in distances between object and the camera for the two views. Thus, a hypothesis contains only pairs of segments, which are similar to each other with respect to these two conditions. If the difference in a segment's shape is not caused by the change of view point, for example, the difference comes from the fact that non-matching segments are being considered. In that case, the values for relative rotation and ratio of cardinalities are random for different pairs. When there is a big difference in these two parameters, it is not possible to extract segments. On the other hand, if the difference in these parameters is caused by the change of the view point, it is the same for any two correctly paired segments. This allows for creation of bigger hypotheses with higher probability that each contains only correct matches.

The four parameters are denoted by the following tables:

$C(i,j)$: color difference,
$O(i,j)$: overlap,
$A(i,j)$: angle of relative rotation,
$RC(i,j)$: ratio of cardinalities.

where (i,j) denotes i-th segment from the first image and j-th segment from the second image, respectively. Next, a brief description of how these parameters are used to evaluate hypotheses, is given.

Ratio of cardinalities. This condition uses the ratio of cardinalities parameter $RC(i,j)$. If all the pairs from a given hypothesis are correct matches, then the value of this parameter for each pair should be in the same range. The minimal and maximal values of $RC(i,j)$ for all pairs from a given hypothesis are found. The minimum and maximum values should be in a $\pm RC_{th}$ range from the mean value of all rations of cardinality for given hypothesis. If any $RC(i,j)$ value from given hypothesis is outside the interval $[(1-RC_{th})*\overline{RC}, (1+RC_{th})*\overline{RC}]$ then a given hypothesis is not valid and is discarded. Otherwise, the next check is performed.

Angle of rotation. This check utilizes the assumption that for correct matches the angles of rotation $A(i,j)$ should be similar to each other for all pairs from a given hypothesis. First, the average angle of all angles is calculated (except for the pair added last). Then the rotation angle from the pair added last, is compared to the average angle. If the absolute value of the difference is greater than some threshold A_{th}, then the given hypothesis fails the check and is removed from the system. Otherwise, the next check is performed.

Triangle property. After passing the *Ratio of cardinalities* and *Angle of rotation* checks, a newly added pair in a given hypothesis is checked against the triangle property. This property assures that a newly added pair preserves the order in which any three segments are arranged in a triangle. Given three segments in one image, one can connect the centroids of these segments creating a triangle. The vertices of this triangle can be ordered in clockwise or counter-clockwise order. After repeating the same procedure for corresponding segments in a second image, a second triangle is formed. By checking the order of the vertices in the second triangle, one can validate the correctness of matches. If the order of vertices is not the same, this does *not* mean that the matching in not correct. The order is preserved between two different views if the triangle of interest is face up on the same side. The centroids of segments need not lay on the plane in the real 3D space. This means that while moving from one view to the second one, the triangle formed by these segments is flipped to the other side, which reverses the order of the vertices. Nevertheless, this effect is very hard to obtain. Notice, that the identified segments would have to look the same from both sides. In most cases, the change in position between the two views is too small to cause this to happen. Hence, despite this special case, the power of discriminating bad matches is very useful for this application and is utilized in this check to decrease the number of hypotheses.

In Alg. 4, the centroid of a segment from a new pair is used to build triangles with all combinations of centroids from the hypothesis. The corresponding triangles for segments from a second image are built as well. If the order of vertices for any of these corresponding triangles do not match, the hypothesis fails the check and is removed from the set M. Otherwise, the algorithm finishes the pruning part and moves to the growing step.

The last part of the matching Alg. 4 identifies the hypothesis, which is the most likely to contain only correct matches. After applying algorithm 4, the set M consists of many hypotheses, which satisfy all conditions. From them, only one

4: Matching Segments

Input: tables $C(i,j), O(i,j), A(i,j), RC(i,j)$, centroids of all segments
Output: hypothesis with highest score, sets of matches M

Set the set of all hypotheses $M = \emptyset$, and $N_M = |M|$;
for *all segments* s_i *in the first image* **do**
  Create pairs $P_i = \bigcup_j P_{ij}$ with all segments S_j from second image;
  Remove from P_i all pairs P_{ij} such that
  $C(i,j) > C_{th}$ or $O(i,j) < O_{th}$;
  Add $N_P = |P_i|$ pairs to N_M existing hypotheses
  producing total of $N_M + N_M \cdot N_P + N_P$ hypotheses;
  foreach *hypothesis* $M_k \in M$ **do**
    Set $\overline{RC} = \sum_{i,j} \frac{RC(i,j)}{|M_k|}$
    if $\exists_{RC(i,j)}. RC(i,j) \notin [(1 - RC_{th}) \cdot \overline{RC}, (1 + RC_{th}) \cdot \overline{RC}]$ *or*
    $|A(i_{new}, j_{new}) - mean_{P(i,j) \in M_k \setminus P(i_{new}, j_{new})} A(i,j)| > A_{th}$ *or*
    $P(i_{new}, j_{new})$ *changes triangle order* **then**
      Remove M_k from M;
    end
  end
end

hypothesis is selected. The measure of correctness is the number of hypotheses the pair associated with a pair of image segments. Each pair is assigned a number based on its hypothesis count. Then each hypothesis is assigned a score, which is the sum of all measures of correctness of all pairs belonging to a given hypothesis. The hypothesis with the highest score is selected as the output of Alg. 4. The list of pairs from the selected hypothesis consists of correctly paired segments from both images.

6 Single Point Standard (Upper Approximation)

The goal of an image-matching system is to match segments from two given images. Consider, for example, Fig. 8 shows generated segments for the Wearever®[3] box scene. The left image in Fig. 8 contains 68 segments and the right image contains 51 segments.

Let $IS = (U, A)$ be an information system, where U is a set of pairs of image segments,and A is a set of image segment attributes. The attributes in A are defined relative to two segments, namely, *degree of overlap*, *angle of rotation*, *distance between mean colors* and *ratio of cardinalities*. Hence, each attribute value is indexed by two numbers which are the indices of the segments in a pair $x \in U$. For example, let S_i, S_j be sets of image segments for image i and image j, respectively. Then the subscripting for the image segment pair (s_i, s_j) specifies

[3] Trademark of the WearEver Company, `http://www.wearever.com`

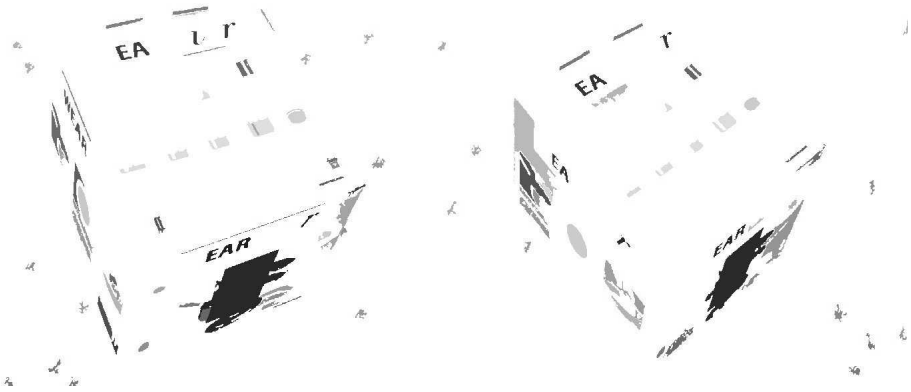

Fig. 8: Generated segments for the Wearever box scene.

that $s_i \in S_i$ and $s_j \in S_j$. Most ranges of values for the segment attributes have been adjusted so that they are in the interval $[0, 1]$ or $[-1, 1]$. A summary of the segment attributes is given in the table 1.

color	range $[0, 1]$. 0 means identical colors; 1, all channels differ by the maximal value, e.g. the color value is 1 if one segments has color $(0, 0, 0)$ in RGB space and the second segment has color $(255, 255, 255)$ in RGB space (where for each channel the range of values is from 0 to 255).
overlap	range $[0, 1]$. 0 means no overlap between segments; 1, identical segments (after translation, rotation and scaling).
angle of rotation	range $[-1, 1]$. 0 means no rotation between segments; 1, rotation by 180 degrees, where the sign denotes the direction of rotation.
RC	range $[0, \infty]$. 1 means that both segments have the same area. For $RC \in [0, 1]$ the second segment is greater than the first one. For $RC > 1$ the first segment is greater than the second one.

Table 1: Attributes' ranges

The angle of rotation for a proper match is unknown. Hence, use of this attribute does not introduce any new information and is not considered in what follows, since the standard for this attribute is unknown.

The rough matching is performed relative to the standard set $B^*\mathcal{Z}$, which is an ideal match of two image segments. In other words, $B^*\mathcal{Z}$ is the optimal case

for matching two identical image segments and such case may, but does not have to exist in the real data.

All segment pairs are ranked based on the information represented by $B^*\mathcal{Z}$. Different upper approximations can be constructed by changing the equivalence relation and subsets of attributes used to obtain $B^*\mathcal{Z}$. In the original K-means clustering algorithm [27], data points are arranged so that they are clustered around K centers. In this work, an equivalence relation based on the K-means clustering algorithm has been introduced (see, e.g., [37]), and which we summarize in this section. Briefly, two segments s_i and s_j are in relation $Ind_K(B)$ if and only if the values of all attributes for s_i and s_j are associated with the same cluster. $Ind_K(B)$ is formally defined in Eq. 12.

$$Ind_K(B) = \left\{ \begin{array}{l} (s_i, s_j) \in U^2 | \ \forall a \in B, \\ \exists l.1 \leq l \leq K, a\left(s_i\right) \in C_l \wedge a\left(s_j\right) \in C_l \end{array} \right\}. \tag{12}$$

where C_l denotes the *l-th* cluster from the set of K clusters. Let the set $\mathcal{Z}$ be defined as in 13.

$$\mathcal{Z} = \left\{ \begin{array}{l} x \in U \times U | \\ color(x) = 0, \\ overlap(x) = 1. \end{array} \right\} \tag{13}$$

The set $\mathcal{Z}$ consists of matched pairs of segments with attribute values specified in 13. Let $B(x)$ be a block in the partition of U, which is a set of B-indiscernible pairs of image segments containing x. At this point, there is interest in finding the upper approximation of $\mathcal{Z}$, which is described in (14).

$$B^*(\mathcal{Z}) = \{x \mid B(x) \cap \mathcal{Z} \neq \emptyset\}. \tag{14}$$

Alg. 5 gives the steps for ranking segment matches using the upper approximation. To each vote is assigned the same unit weight. Because cases where 2 attributes are used include cases where 1 attribute is used, the effective weights are greater for cases with multiple attributes used. The table 2 show the effective voting weights for the algorithm 5.

Figures 9 and 10 show sample voting results for two segments. A circle $\circ$ denotes the good match made by visual inspection of the two images, and a cross + denotes the segment which is the closest to the standard $\mathcal{Z}$. That is, for a given segment i from the first image, a + denotes the segment j_{min} from the second image such that

$$j_{min} = \min_j |\mathcal{Z} - \{C(i,j), O(i,j)\}|.$$

Fig. 9 shows how the information is extracted from the generated attributes using the upper approximation $B^*(\mathcal{Z})$ of the set $\mathcal{Z}$. The cross + shows that the best match using the distance between the given three parameters is with segment number 44. However, the correct match is with segment number 18. The number of votes for the segment number 18 is higher than the number of

5: Matching Segments Using Upper Approximation

Input: set of attributes $\mathcal{A} = \{C(i,j), O(i,j)\}$
Output: ranking of all segment pairs s_{ij}

```
for (all segments s_i in the first image) do
    for (K=2 to (# of segments in the second image)/2) do
        Perform K-means clustering for each attrib. separately
        for (each subset B of the set of all attributes A) do
            Find B*(Z)
            for (each segment s_j from the second image) do
                if (s_j ∈ B*(Z)) then
                    vote for pair s_ij
                end
            end
        end
    end
end
```

# of attributes in B	# of votes	effective # of votes
1	1	1
2	1	3

Table 2: Voting table $\mathcal{T}$ for algorithm 5

votes for the segment 44. This means that using this algorithm, segment 18 is more likely to be chosen as the match than segment 44.

The problem which is still to be solved is the high number of segments with high votes. For example, in Fig. 9, segments 18, 20 and 33 have high votes and it is not possible to select the best match. Hence, there is interest in considering the lower approximation $B_*(\mathcal{Z})$ of the set $\mathcal{Z}$.

7 Interval Standard (Lower Approximation)

This section presents an extension of the method described in Sect. 6. The lower approximation $B_*(\mathcal{Z})$ is derived relative to $\mathcal{Z}$, which is defined as the approximation of a set of image segment pairs that constitute a perfect match. However, this is a bit unrealistic and not flexible. To allow for some tolerance in concluding that an image segment pair constitutes a match, an interval interpretation of the attribute values of image segment pairs is introduced. That is, the attribute values associated with image segment pairs in the set $\mathcal{Z}$ are parametrized by a parameter δ, which denotes the optimal value for each attribute. In effect, each attribute value of each image segment pair $x \in \mathcal{Z}$ belongs to a small interval containing δ. Using this approach, $\mathcal{Z}$ is defined as in Eq. 15.

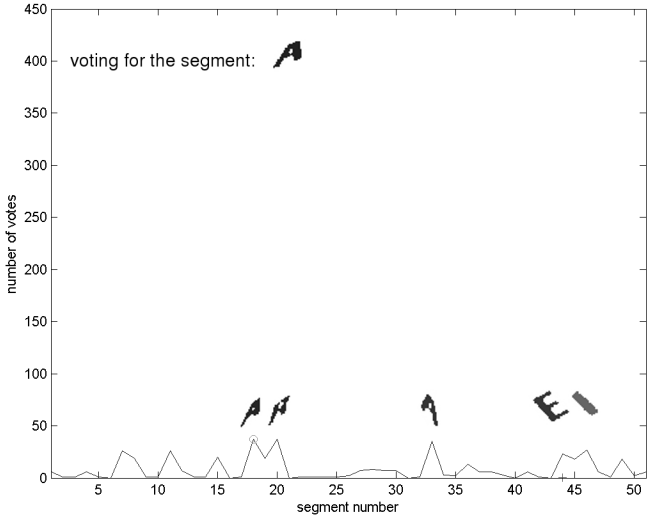

Fig. 9: Voting results. ∘ good match, + the closest match.

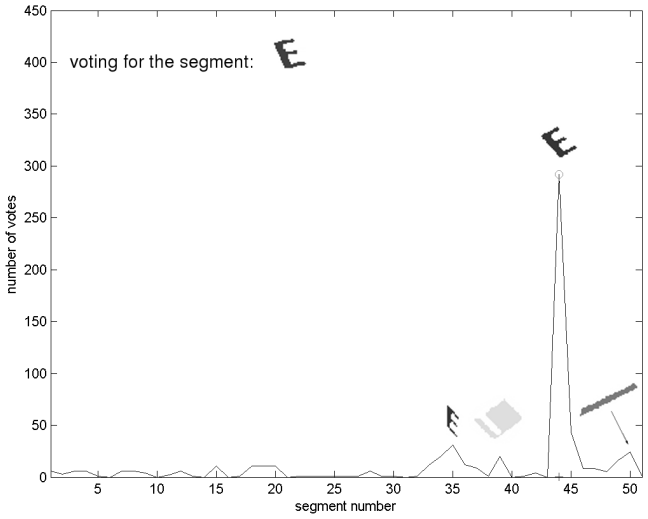

Fig. 10: Voting results: ∘ good match, + the closest match.

$$\mathcal{Z} = \left\{ \begin{array}{l} x \in U \times U| \\ color(x) \in (0, \delta)\,, \\ overlap(x) \in (1-\delta, 1)\,. \end{array} \right\}. \tag{15}$$

where the attribute values for each image segment pair x in $\mathcal{Z}$ belong to intervals for color and overlap specified in Eq. 15.

For experiments, the parameter δ was set to 0.1. The results for different δ values did not differ significantly from the ones shown here. The formula for calculating the lower approximation is given in 16.

$$B_*(\mathcal{Z}) = \{x \mid B(x) \subseteq \mathcal{Z}\}. \tag{16}$$

The new matching algorithm is essentially the same as Alg. 5, except that a *Find* $B_*(\mathcal{Z})$ operation has been added.

As can be seen from Fig. 12, the best results are obtained for the 'single point standard'. The notation circle $\circ$ in Fig. 12 denotes a good match made by visual inspection of the two images, and a cross + indicates a segment pair which is the closest to $\mathcal{Z}$. The 'interval standard' method fails to yield one segment pair as a good match. Instead, it yields several segments with equally high votes. This means that this method cannot be used by itself as the deciding method for solving the matching problem. However, the interval standard method can be used as an aid, since the correct solution is usually among the segment pairs with the highest votes.

Example 2. The figure 11 shows two images used to explain in more detail the idea of the standard $\mathcal{Z}$. Left part of the figure 11 shows the first image. It consists of twelve segments created from the letters of a word "Matching". Notice, there are only eight letters in a word "Matching". The remaining four segments are: white area in the letter 'a' (denoted by a.), a dot in 'i' letter (denoted by i$_\cdot$), upper white area in the letter 'g' (denoted by g$^\cdot$) and lower white area in the letter 'g' (denoted by g$_\cdot$).

The right part of the image 11 shows the same letters as the first image. Only, for the second image, they underwent geometrical transformations: image warping, rescaling and rotation. In addition, brightness of each letter from the second image was randomly altered.

In order to construct the standard $\mathcal{Z}$ the color difference and overlap parameters were calculated. The color difference values are shown in the table 3 and overlap values are shown in the table 4. Values corresponding to proper matches are denoted by bold face font in both tables. For example, the pair of segments 'M' from both images is characterized by the pair (0.070,0.614), where the first number denotes the color difference and the second number denotes the overlap value for these two segments.

The creation of standard $\mathcal{Z}$ for given parameter δ is straight forward. From the equation 15, standard is a set of all segment pairs for which the color difference and the overlap values are in some interval, e.g. color difference is less than δ and overlap is greater than $1 - \delta$. For example, for $\delta = 0.1$ there are

Fig. 11: Segments for example 2

Table 3: Color table

	M	a	t	c	h	i	i·	n	g	a.	g·	g.
M	**0.070**	0.574	0.671	0.066	0.572	0.532	0.532	0.639	0.068	0.784	0.784	0.784
a	0.579	**0.064**	0.707	0.549	0.606	0.492	0.492	0.213	0.647	0.778	0.778	0.778
t	0.654	0.693	**0.048**	0.595	0.091	0.875	0.875	0.528	0.657	0.773	0.773	0.773
c	0.118	0.539	0.628	**0.011**	0.532	0.545	0.545	0.596	0.126	0.792	0.792	0.792
h	0.571	0.608	0.179	0.531	**0.045**	0.753	0.753	0.449	0.585	0.647	0.647	0.647
i	0.481	0.452	0.898	0.564	0.760	**0.069**	0.069	0.554	0.558	0.483	0.483	0.483
i·	0.482	0.454	0.894	0.565	0.756	0.075	**0.075**	0.552	0.559	0.477	0.477	0.477
n	0.649	0.305	0.556	0.636	0.449	0.551	0.551	**0.108**	0.710	0.579	0.579	0.579
g.	0.082	0.617	0.679	0.171	0.573	0.507	0.507	0.666	**0.078**	0.705	0.705	0.705
a.	0.719	0.727	0.800	0.790	0.679	0.542	0.542	0.657	0.768	**0.014**	0.014	0.014
g·	0.715	0.726	0.799	0.786	0.677	0.540	0.540	0.657	0.764	0.017	**0.017**	0.017
g.	0.718	0.728	0.800	0.790	0.679	0.542	0.542	0.659	0.767	0.014	0.014	**0.014**

Table 4: Overlap table

	M	a	t	c	h	i	i·	n	g	a.	g·	g.
M	**0.614**	0.084	0.019	0.058	0.372	0.040	0.001	0.242	0.210	0.001	0.001	0.002
a	0.367	**0.383**	0.143	0.206	0.221	0.165	0.010	0.497	0.276	0.003	0.005	0.010
t	0.141	0.330	**0.832**	0.292	0.249	0.001	0.001	0.151	0.153	0.016	0.142	0.203
c	0.181	0.266	0.202	**0.722**	0.228	0.250	0.043	0.316	0.220	0.069	0.081	0.121
h	0.052	0.411	0.331	0.139	**0.714**	0.247	0.002	0.549	0.251	0.008	0.004	0.021
i	0.301	0.275	0.001	0.306	0.134	**0.716**	0.133	0.230	0.166	0.267	0.508	0.423
i·	0.003	0.256	0.001	0.113	0.032	0.216	**0.913**	0.033	0.018	0.421	0.675	0.807
n	0.333	0.390	0.116	0.368	0.239	0.149	0.012	**0.847**	0.280	0.001	0.002	0.024
g.	0.234	0.268	0.118	0.109	0.155	0.115	0.001	0.091	**0.713**	0.001	0.001	0.010
a.	0.001	0.045	0.320	0.102	0.028	0.185	0.726	0.001	0.007	**0.861**	0.818	0.721
g·	0.001	0.060	0.152	0.054	0.015	0.274	0.730	0.003	0.002	0.675	**0.931**	0.873
g.	0.001	0.108	0.251	0.130	0.042	0.425	0.618	0.337	0.039	0.726	0.865	**0.860**

only two segments satisfying the above requirements. These pairs are ($i^{\cdot}$,$i^{\cdot}$) for which color($i^{\cdot}$,$i^{\cdot}$)= 0.075 < 0.1, overlap($i^{\cdot}$,$i^{\cdot}$)= 0.913 > 0.9, and ($g^{\cdot}$,$g^{\cdot}$) for which color($g^{\cdot}$,$g^{\cdot}$)= 0.017 < 0.1, overlap($g^{\cdot}$,$g^{\cdot}$)= 0.931 > 0.9, see tables 3 and 4. Therefore, for $\delta = 0.1$ the standard $\mathcal{Z} = \{(i^{\cdot},i^{\cdot}), (g^{\cdot},g^{\cdot})\}$. This means that the segments $i^{\cdot}$ and $g^{\cdot}$ are the most similar segments in both images. The matching of the remaining segments is performed relative to this match.

Table 5: $\mathcal{Z}$ table vs. δ parameter

δ	$\mathcal{Z}$	(color, overlap)
0.05	$\emptyset$	
0.1	$\{(i^{\cdot},i^{\cdot}), (g^{\cdot},g^{\cdot})\}$	{(0.075,0.913), (0.017,0.931)}
0.15	$\{(i^{\cdot},i^{\cdot}), (a.,a.), (g^{\cdot},g^{\cdot}),$ $(g_{\cdot},g^{\cdot}), (g^{\cdot},g_{\cdot}), (g_{\cdot},g_{\cdot})\}$	{(0.075,0.913), (0.014,0.861), (0.017,0.931), (0.014,0.865), (0.017,0.873), (0.014,0.860)}
0.2	{(t,t), $(i^{\cdot},i^{\cdot})$, (n,n), (a.,a.), $(a.,g^{\cdot})$, $(g^{\cdot},g^{\cdot})$, $(g_{\cdot},g^{\cdot})$, $(g^{\cdot},g_{\cdot})$, $(g_{\cdot},g_{\cdot})$}	{(0.048,0.832), (0.075,0.913), (0.108,0.847), (0.014,0.861), (0.014,0.818), (0.017,0.931), (0.014,0.865), (0.017,0.873), (0.014,0.860)}
0.3	{(t,t), (c,c), (h,h), (i,i), $(i^{\cdot},i^{\cdot})$, (n,n), (g,g), (a.,a.), $(g_{\cdot},a.)$, $(a.,g^{\cdot})$, $(g^{\cdot},g^{\cdot})$, $(g_{\cdot},g^{\cdot})$, $(a.,g_{\cdot})$, $(g^{\cdot},g_{\cdot})$, $(g_{\cdot},g_{\cdot})$}	{(0.048,0.832), (0.011,0.722), (0.045,0.714), (0.069,0.716), (0.075,0.913), (0.108,0.847), (0.078,0.713), (0.014,0.861), (0.014,0.726), (0.014,0.818), (0.017,0.931), (0.014,0.865), (0.014,0.721), (0.017,0.873), (0.014,0.860)}

The table 5 shows several sets $\mathcal{Z}$ for different values of parameter δ. The bigger the parameter δ the more matches are included in the standard $\mathcal{Z}$. This is the core of the rough set approach, where the definition of an 'ideal match' is not fixed, but can be adjusted based on available information and data. Notice, for smaller δ values the standard $\mathcal{Z}$ is an empty set, which means that the identical segments are not present in both images. On the other hand, bigger δ values produce a standard, which contains incorrect matches. Nevertheless, at this stage, the matching correctness is not crucial, in fact, segments ($g_{\cdot}$,$g^{\cdot}$) form better match than ($g_{\cdot}$,$g_{\cdot}$), since (color, overlap) values for the first pair are (0.014, 0.865) and for the second pair (0.014, 0.860). The correctness of this match cannot be determined using only color difference and overlap values, but other parameters must be used as well.

As shown in this example, the δ parameter allows for adjusting how strict the definition of an ideal match is. This was not possible in the Upper Approximation based approach, see equation 13.

8 Genetic Approach for Matching

The genetic approach for segment matching creates a framework for the search based on any set of features extracted from images. Features can be extracted using segments, lines or Harris corners [15]. In addition, because the genetic

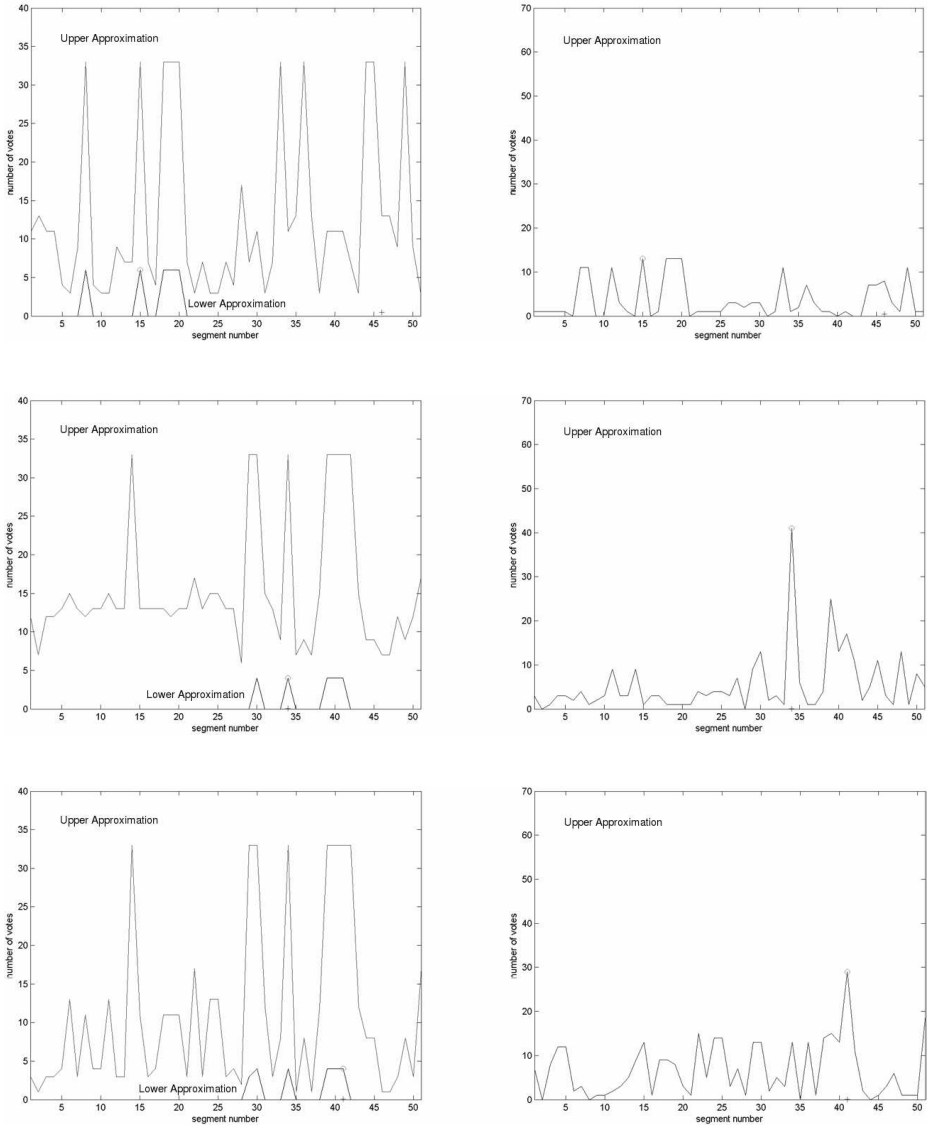

Fig. 12: Voting results: 'interval standard' (left) and 'single point standard' (right).

algorithm is used, the search space can be very large. This allows for selection of matches from large set of pixels from both images.

The central notion of a classical, Darwinian form of genetic algorithm is the chromosome. It consists of genes. A *gene* represents a pair of matched features for two images. There are many types of features that can be used in the algorithm. More abstract forms (called also *shapes*) are source of features. Three methods of deriving features have been considered so far, and are summarized in the table 6. One of the functions of the gene is to hide the differences between the features for the genetic algorithm. The genetic algorithm does not discern between the genes and processes them the same way. The set of features creates a *chromosome* (or *hypothesis* as described in Sect. 5.4). A genetic algorithm tries to select the best hypothesis, which consists of only correct matches. The term *hypothesis* is used in the context of matching features from the images, since a hypothesis identifies possible matches of features. The term *chromosome* is used in the context of the genetic algorithm, since the chromosome is the member of the population.

Abstract Form	Complexity	Features
point	simple	cross-correlation
line	moderate	cross-correlation, angle, RC
segment	complex	color, overlap, angle, RC

Table 6: Features and their corresponding abstract forms.

The overview of the structures used in the algorithm is given in the figure 13. Three kinds of feature generators are denoted by three paths at the bottom of the image. The tasks of image processing blocks are generation of points, lines and segments. After this step, the identified shapes are passed to the feature extraction blocks. These blocks use selected shapes to generate features. The term 'feature' needs more explanation. Usually, the term $feature$ means a mapping of observed object in the universe to an attribute value. In this case, a feature is not an attribute or aspect of a single image segment or pixel, but a result of a comparison of a pair of image segments or pixels. In the case of the color of a pair of image segments, a feature is the difference in the average colors of the two segments. In general, a feature $\mathcal{F}(x, y)$ for a pair of observed objects x and y is a scalar from some pre-defined range $[a, b]$ as defined in (17).

$$\mathcal{F} \ : \ (x, y) \longrightarrow c \in \mathbb{R} \quad a \leq c \leq b. \tag{17}$$

In addition, there is one point $\kappa \in [a, b]$, which denotes the value for which a pair of objects are not discernible with respect to the given feature. For example, for colors of image segment pairs, the possible range of color differences can be defined as $[0, 1]$, where 0 denotes two identical colors and 1 denotes the maximum difference of colors allowed by an image's color depth. In this case, $\kappa = 0$.

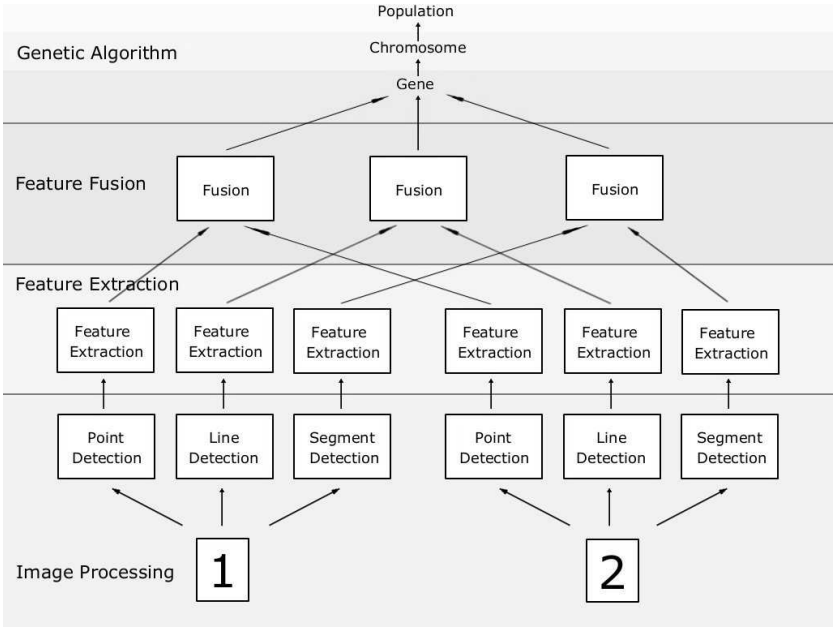

Fig. 13: The overview of matching algorithm.

The fact that the features are calculated as the difference between attributes for pairs of digital images is denoted by the 'fusion' box in the figure 13. The procedure represented by the fusion box combines the information from pairs of images to generate feature values.

Next, the description of the chromosome is given (see figure 14). A gene represents a match between two shapes from a pair of images. This is indicated by a pair of indices of two corresponding shapes. Genes in each chromosome are divided into three blocks: point block, line block and segment block. Each block contains indices of matched shapes of a given type, namely point, line or segment.

The number of genes in each group can be zero. The genetic algorithm does not discern between different types of genes as long as both halves of the gene are of the same type. The order of the chromosome is the sum of lengths of all blocks, e.g. $np + nl + ns$.

The current version of the genetic algorithm has only image segment genes implemented. Therefore, point and line blocks are always empty. The creation of the genes is constrained by the rules, which assure that only reasonable matches are considered. These rules control the color difference, overlap and the ratio of cardinalities. Let $color\_th$, $overlap\_th$, rc_th denote the maximum values allowed for the color difference, overlap and the ratio of cardinalities for a pair of image segments, respectively. For the ratio of cardinalities parameter the threshold denotes the the maximum difference of the areas of a pair of segments,

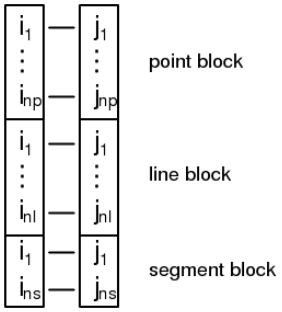

Fig. 14: The chromosome. Each block contains indices of matched shapes.

e.g. for $rc\_th = 2$ it means that $\frac{1}{2} \leq RC \leq 2$. Table 7 gives constraints for feature values during the creation of genes.

Feature	**Condition**
Color	$\leq color\_th$
Overlap	$\geq overlap\_th$
RC	$\geq 1/rc\_th \wedge \ \leq rc\_th$

Table 7: Rules for creating genes.

Let Ch denote a chromosome from a population evaluated by a genetic algorithm. Further, let ang, $\overline{ang}$, rc_th, ang_th denote angle of rotation, average angle of rotation, ratio of cardianlities threshold, and angle threshold, respectively. The $RC(i_k, j_k)$ parameter was defined in Eq. 11 and the $\overline{RC}$ symbol denotes the mean value of ratios of cardinalities for all genes from given chromosome. The subscript k iterates from 1 to the number of genes in given chromosome such that subscripts (i_k, j_k) iterate through all image segments contained in the chromosome, see figure 14. The fitness of Ch is determined using Eq. 18.

$$Fitness(Ch) = \begin{cases} 1 & \forall_k \ \ (1 - RC_{th}) \cdot \overline{RC} \leq RC(i_k, j_k) \leq (1 + RC_{th}) \cdot \overline{RC} \\ & \wedge \ \ max_k \ |ang(i_k, j_k) - \overline{ang}| < ang\_th \\ & \wedge \ \text{passes the triangle check,} \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

The fitness function given in Eq. 18 is maximally selective, e.g. it causes chromosomes to survive and reproduce with equal probability or die and be removed from the population.
Alg. 6 is based on the standard procedure for genetic algorithms described in [6]. The only genetic operator implemented so far is the crossover operation. New

6: Genetic Algorithm

Input: tables $C(i,j), O(i,j), A(i,j), RC(i,j)$, centroids of all segments
Output: ordered set of matches $\mathcal{O}$

Create the set of genes S using rules from table 7
while *(stop condition is not true)* **do**
  Apply genetic operator: crossover
  Evaluate fitness function
  Remove chromosomes with fitness function below some threshold
end
Order all genes into set $\mathcal{O}$ using *GA based ordering* algorithm, see alg. 7

genes are not created by Alg. 6. All unique genes appearing in the population are created before evolutionary iteration starts. The crossover operation cannot split halfs of existing genes. Two chromosomes of lengths n_1 and n_2 can only be concatenated to form a new chromosome of a length $n_1 + n_2$, which contains all genes from the original two chromosomes. The repetition of left and right handed parts of the gene within a chromosome is not allowed either. This means that only two chromosomes with different sets of left and right handed parts can mate and create an offspring.

After implementing Alg. 6, all genes are scored using the Alg 7. The symbol γ denotes partitioning introduced by the chromosomes. Each chromosome forms a block of genes. Genes within a block (a chromosome) are considered to be indiscernible. Since, different chromosomes can contain the same genes this partitioning forms a tolerance relation.

The genes are sorted by the number of chromosomes they appear into. The chromosome which contains the most common genes is selected as the output of the simulation. All sorted genes are returned in the set $\mathcal{O}$.

7: GA based ordering

Input: set of genes G, partitioning of this set γ
Output: ordered set of matches $\mathcal{O}$

Create the set of counters for all genes in the set G
Set initial values of these counters to 0
for *(all blocks from γ)* **do**
  for *(all genes s_i from given block)* **do**
    Increase counter of gene s_i by 1
  end
end
Return sorted in descending order list of genes $\mathcal{O} = sort(G)$

9 2D Matching with Approximation Spaces

This section considers an approach to processing the output from the genetic algorithm 6 within the context of an approximation space. Let S be a set of n best genes returned by Alg. 6. Let $B(x)$ be a block of genes equivalent to x, and let B_*S be the lower approximation of the set S. There is advantage in using B_*S as a standard, and measuring how well B_*S "covers" each block. In this way, it is possible to select a block covered to the greatest extent by the standard, and which represents the set of best image segment matches. The steps of this approach to finding the set of best matches are given in Alg. 8.

8: The algorithm for selecting best matches using rough coverage

Input: set of all possible matches
Output: ordered set of matches

Run the GA to get the partitioning of the genes
Create and initialize to 0 the rough coverage weights for each gene
Create a set S of top n genes
for *(each block $B(x_i)$)* **do**
  Evaluate rough inclusion value

$$Rcover(B(x_i), S) = \frac{|B(x_i) \cap B_*S|}{|B_*S|}$$

  Increase weights of genes from $B(x_i)$ by $Rcover(B(x_i), S)$
end

The results are shown in the plot in the figure 15. In the plot from the figure 15, $n = 114$ is called the pool of genes. The most important thing to observe in this plot is that rough coverage does better between genes numbers 37 and 49. This means Alg. 8 sorts the genes better than the pure GA represented in Alg. 6.

9.1 Tolerance Relation vs. Equivalence Relation

The output of the GA in Alg. 6 is a set of hypotheses. In other words, Alg. 6 produces sets containing pairs of image segments. Each such set (hypothesis) corresponds to one chromosome. The crossover operation in Alg. 6 produces a chromosome, which is a copy of two input chromosomes. Therefore, the resulting partitioning is a tolerance relation (see property 1). Alg. 9 converts the tolerance relation induced by Alg. 6 into an equivalence relation. This is done by removing the overlapping sets in the partition created by Alg. 6.

Observe that Alg. 9 searches for the chromosomes with the highest weight (starting from the longest chromosomes) and removes all chromosomes which

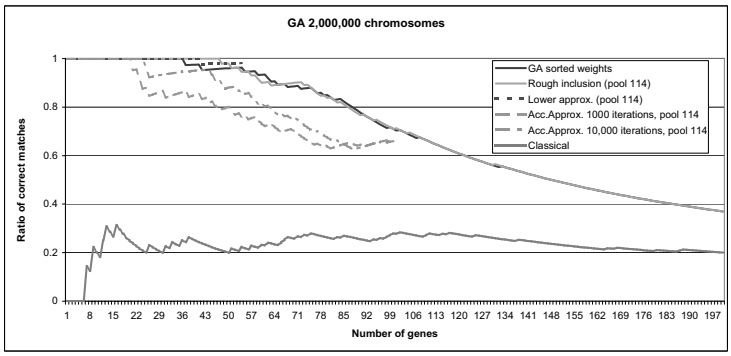

Fig. 15: Rough coverage vs. % correct matches with 2,000,000 chromosomes.

have non-empty intersection with a given chromosome. The resulting partitioning of all genes C forms an equivalence relation (see theorem 2).

The down side to converting the tolerance relation induced by Alg. 6 into an equivalence relation is the reduction of the number of blocks. For example, in case of the system consisting of ≈818,000 chromosomes with 4920 genes where the longest chromosome has length 29, the conversion to equivalence relation decreases the number of blocks with cardinality greater than one to 1229. This means that the number of blocks after the conversion is less than 0.16% of the number of tolerance relation blocks.

Experimental results show that the equivalence relation does not have enough power to improve the ordering produced by the GA in Alg. 6. This is due to the small number of blocks in the equivalence relation. Fig. 16 shows sample results for 818,000 chromosomes.

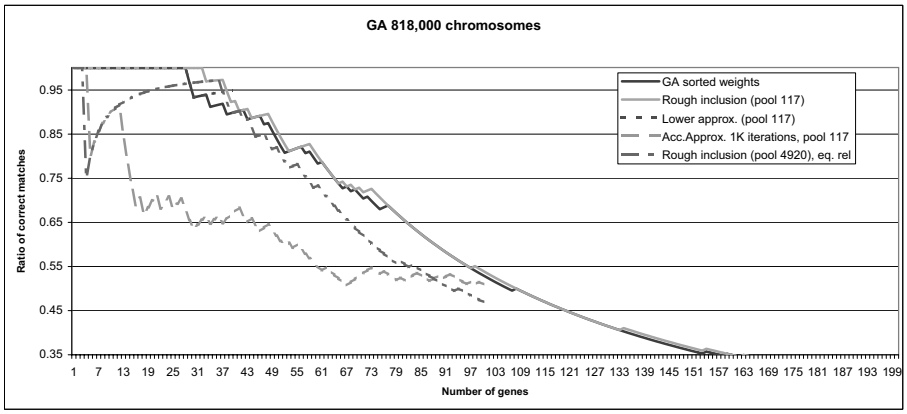

Fig. 16: Ratio of correct matches for tolerance and equivalence relation

9: Conversion of tolerance rel. to equivalence relation

Input: set of genes G, set of chromosomes $\mathcal{Ch}$, partitioning γ of set G expressed by sets $\mathcal{Ch}_I \in \mathcal{Ch}$

Output: partitioning Ind of set G, which is an equivalence relation

```
1  Order all genes using GA based ordering, see alg. 7
2  Set Ind to ∅
   /* starting from the longest chromosomes                    */
4  for (all chromosomes' lengths I) do
5  |  for (all chromosomes Ch_I of the length I) do
6  |  |  Find the chromosome ch_max ∈ Ch_I with the highest score and
   |  |  move it to Ind
7  |  |  for (all chromosomes ch_k in Ch) do
8  |  |  |  if (ch_k ∩ ch_max ≠ ∅) then
10 |  |  |  |  Remove ch_k from Ch
11 |  |  |  end
12 |  |  end
13 |  end
14 end
```

Definition 11. ***Tolerance Relation*** *(from [21])*
A binary relation $\tau \subseteq X \times X$ is called a **tolerance relation** if and only if τ is

1. reflexive, an object is in relation with itself $x\tau x$,
2. symmetric, if $x\tau y$ then $y\tau x$.

Definition 12. ***Equivalence Relation*** *(from [21])*
A binary relation $R \subseteq X \times X$ is called an **equivalence relation** if and only if R is a tolerance relation and is

1. transitive, if xRy and yRz then xRz.

Property 1 The crossover operator in the GA in Alg 6 produces a partitioning of the set of genes G, such that created blocks have non-empty intersection.

Proof. Let $\mathcal{Ch}$ denote the set of all chromosomes ch_k such that $\mathcal{Ch} = \bigcup_k ch_k$. If L denotes the longest chromosome in $\mathcal{Ch}$, then all chromosomes can be grouped into subsets of $\mathcal{Ch}$, namely $\mathcal{Ch}_L, \mathcal{Ch}_{L-1}, \ldots, \mathcal{Ch}_I, \ldots, \mathcal{Ch}_1$, where $\mathcal{Ch}_I \subseteq \mathcal{Ch}$ and index I denotes the length of the chromosome[4]. The crossover operator $crossov$ can be defined as follows:

$$crossov \ : \ \mathcal{Ch}_I \times \mathcal{Ch}_J \rightarrow \mathcal{Ch}_K, \quad ch_k = crossov(ch_i, ch_j)$$

where $ch_i \in \mathcal{Ch}_I$, $ch_j \in \mathcal{Ch}_J$, $ch_k \in \mathcal{Ch}_K$, $K = I + J$, $K \geq 2$ and $I, J \geq 1$.

[4] The index written with a small letter by a chromosome ch does not indicate the length of the chromosome.

After applying the crossover operator, the chromosomes ch_i and ch_j are not removed from the set of all chromosomes $\mathcal{C}h$, i.e. $ch_i, ch_j, ch_k \in \mathcal{C}h$. This means that a gene g_t which belongs originally to ch_i belongs also to the chromosome ch_k.

Now, for any chromosome ch_k of order greater than 1.

$$\forall_{g_t \in ch_k} \; \exists l \neq k \mid g_t \in ch_l$$

From the fact that the order of a chromosome ch_k is greater than 1, we have $ch_k = crossov(ch_i, ch_j)$. What follows is that $l = i$ or $l = j$.

■

The above proof shows that for each chromosome ch_k of order $K > 1$ there exists a chromosome of order less than K, which is a part of ch_k. Therefore, for each such chromosome ch_k there exist at least two chromosomes that have non empty intersection with it.

Theorem 1. The GA in Alg 6 produces a partitioning of the set of all genes, which corresponds to the tolerance relation γ (and not an equivalence relation).

Proof. Chromosomes ch_k produced by the GA consist of the genes from the set G. Each chromosome can be considered as a block of indistinguishable genes. Thus, they create a partitioning of the set G. This partitioning corresponds to the tolerance relation if the relation determined by this partitioning is reflexive and symmetric. The relation γ is based on the fact that two elements belong to the same chromosome, i.e.

$$x \gamma y \;\; \text{iff} \;\; \exists_i \mid \;\; x \in ch_i \;\; \text{and} \;\; y \in ch_i$$

where $ch_i \in \mathcal{C}h$. From the definition, belonging to a set is symmetric and reflexive.

The partitioning $\mathcal{C}h$ covers all genes G because $\mathcal{C}h$ includes the set of chromosomes of order one $\mathcal{C}h_1$, which is identical with the set of all genes G, i.e. $\mathcal{C}h_1 = G$, $\mathcal{C}h_1 \in \mathcal{C}h$ therefore $G \subseteq \mathcal{C}h$.

The relation γ is not an equivalence relation because chromosomes ch may have non-empty intersections (from property 1). As the result, the transitivity constraint is not satisfied. If for $k \neq l$ holds $x \in ch_k$, $y \in ch_k$, $y \in ch_l$, $z \in ch_l$ and $x \notin ch_l$ and $z \notin ch_k$ then $x \gamma y$, $y \gamma z$ and x is not in relation with z.

■

Theorem 2. Alg. 9 converts the partition γ produced by the GA in Alg. 6 into a partition Ind which is an equivalence relation.

Proof. The partitioning Ind is a subset of the partition γ. Thus, there are two conditions that must be satisfied for the partition Ind to be an equivalence relation:

– all blocks from Ind must have empty intersection:

  The step 9 from algorithm 9 assures that all subsets from Ind have empty intersection.
– all blocks from Ind must cover the entire space of genes G:

  The step 9 from algorithm 9 starts with the longest chromosomes and ends with the shortest $Ch_L, Ch_{L-1}, \ldots, Ch_I, \ldots, Ch_1$, where L is the length of the longest chromosome in the system. The shortest chromosome is of length one, i.e. it is a gene. Notice, none of the genes which are not included in the set Ind will be removed from the set Ch_1 because their intersection with ch_{max} is empty. This means that in the last iteration of loop 9 all missing genes will be added to the set Ind.

■

9.2 Classical vs. Rough Matching Methods

Classical 2D image segment matching method is usually limited to two features, namely, color difference and the overlap between two segments (see, e.g., [12,55,24]). This is a severe limitation because these two features do not yield enough information to permit accurate image segment matching. By contrast, in designing genes in chromosomes used in evolutionary 2D segment matching, the number of features associated with a gene can be quite large.

In this study, four parameters for each gene and 2,000,000 chromosomes have been used. Similarly, using the rough coverage methods, the number of features (parameters) associated with an image segment can be large. In this study, four features are used, namely, the distance between mean colors of segments, degree of overlap between segments, the angle of rotation between segments and ratio of cardinalities of both segments. In addition, rough coverage values computed within the context of an approximation space, represent a comparison between each of the blocks containing similar pairs of image segments and a set representing a norm (e.g., B_*S). In effect, the rough coverage matching method yields better results because it uses more information about the image segments being compared. This is one way to explain the plots in Fig. 15.
In Fig. 17, the left upper corner of the plot from Fig. 15 is shown. Fig. 17 illustrates the advantage of the rough coverage approach compared to the other methods. Recall, that the problem of segment matching is considered in the context of 2D to 3D conversion. The 2D to 3D conversion algorithm takes as an input paired pixels, which are generated from paired segments. Any mismatch at the segment matching stage propagates to the pixel matching stage and finally into 2D to 3D conversion. Therefore, a crucial requirement for image segment matching is to generate as little wrong matches as possible. Fig. 17 shows that the rough coverage approach yields the biggest number of correct matches, i.e. the first wrong match occurs after finding 47 good matches. For the weights generated by the GA used in in Alg. 8, the first mismatch occurs after only 36 correct matches. Hence, rough coverage greatly reduces the number

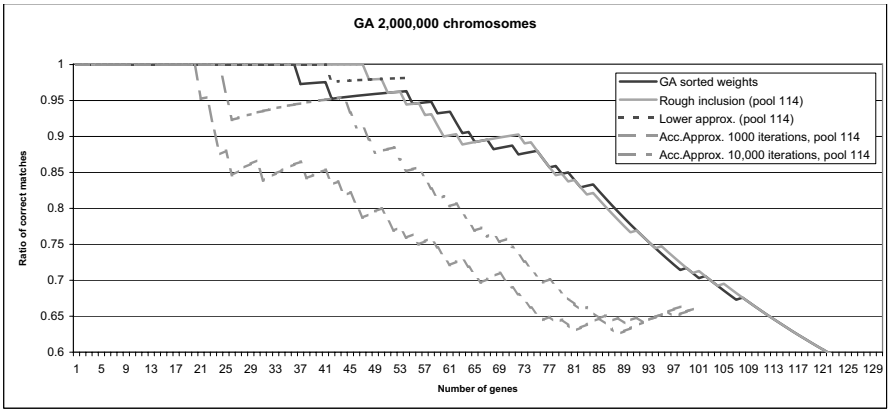

Fig. 17: Rough coverage vs. % of correct matches (zoomed Fig. 15).

of mismatches, which improves the robustness of the overall 2D to 3D conversion process.

10 Conclusion

An approach to using a combination of genetic algorithms and approximation spaces in solving the image segment matching problem is given in this paper. Approximation spaces are used as a form of visual perception of pairs of images, which is step towards 2D image classification in the case where one of the paired images plays the role of a reference image for matching purposes.

Acknowledgements

The authors would like to thank Andrzej Skowron for his observations concerning approximation spaces and his helpful suggestions concerning this article. This research has been supported by Natural Sciences and Engineering Research Council of Canada (NSERC) grant 185986 and grant T247 from Manitoba Hydro.

References

1. Bishop, C.M.:*Neural Networks For Pattern Recognition*. Oxford Univ. Press, UK (1995).
2. Borkowski, M.: Digital Image Processing in Measurement of Ice Thickness on Power Transmission Lines: A Rough Set Approach, M.Sc. Thesis, Supervisor: J.F. Peters, Department of Electrical and Computer Engineering, University of Manitoba (2002).

3. Gottesfeld Brown,L.: *A Survey Of Image Registration Techniques*. ACM Computing Surveys, 24, No. 4, December (1992).
4. Caelli,T., Reye,D.: On the classification of image regions by color, texture and shape, Pattern Recognition, 26 (1993) 461-470.
5. Darwin, C.: On the Origin of the Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life. Murray, London (1859).
6. Duda,R.O., Hart,P.E., Stork, D.G: *Pattern Classification*, 2nd Edition. John Wiley & Sons, Chichester (2001) 373-377.
7. Fitzpatrick, J.M, Hill, D.L.G., Calvin,R.M. Jr.: *Chapter 8: Image Registration*. In: Sonka, M., Fitzpatrick, J.M., *Handbook of Medical Imaging, Vol. 2: Medical Image Processing and Analysis*. SPIE PRESS Vol. PM80, Bellingham, USA, (2000).
8. Fu, K.S., Mui, J.K.: A survey of image segmentation, Pattern Recognition, 13 (1981) 3-16.
9. Eshelman, L.J.: Genetic algorithms. In: Back, T., Fogel, D.B., Michalewicz, Z. (Eds.), Handbook of Evolutionary Computation. Oxford University Press, Oxford, UK (1997).
10. Gersho,A., Gray,R.M.: *Vector Quantization And Signal Compression*. Kluwer Academic Publishers,Dordrecht,The Netherlands (1992).
11. Gomolinska, A.: Rough validity, confidence, and coverage of rules in approximation spaces, Transactions on Rough Sets III (2005) 57-81.
12. Gonzalez,R.C., Woods,R.E.: *Digital Image Processing*, 2nd Edition. Prentice Hall, NJ (2002).
13. Gray, R.M., Neuhoff, D.L.: Quantization, IEEE Transactions on Information Theory, 44 (1998) 1-63.
14. Haralick, R.M., Shapiro, L.G.: Image segmentation techniques, Computer Vision, Graphics, and Image Processing, 29 (1985) 100-132.
15. Harris,C., Stephens,M.: *A Combined Corner And Edge Detector*. In: Proceedings of the 4th Alvey Vision Conference, Manchester (1988) 189192.
16. Hartigan, J.A., Wong, M.A.: A K-means clustering algorithm, Applied Statistics, 28 (1979) 100-108.
17. Heaton, K.G.: Physical Pixels, M.Sc. Thesis, MIT (1994).
18. Hirano, H., Tsumoto, S.: Segmentation of medical images based on approximation in rough set theory. In: J.J. Alpigini, J.J., Peters, J.F., Skowron, A., Zhong, N. (Eds.), LNAI 2475. Springer-Verlag, Berlin (2002) 554-563.
19. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI (1975).
20. Paton, R.C.: Principles of genetics. In: T. Back, T., Fogel, D.B., Michalewicz, Z. (Eds.), Handbook of Evolutionary Computation. Oxford University Press, Oxford, UK (1997).
21. Komorowski,J., Pawlak,Z., Polkowski,L., Skowron,A.: *Rough Sets: A Tutorial*. In: Pal, S.K., Skowron, A. (Eds.), Rough Fuzzy Hybridization. A New Trend in Decision-Making. Springer-Verlag Singapore Pte. Ltd., Singapore (1999) 1-14.
22. Konrad, E., Orlowska, E., Pawlak, Z.: Knowledge Representation Systems. Definability of Informations. Institute for Computer Science, Polish Academy of Sciences Report 433, April (1981).
23. Lee, S.Uk, Chung, S.Y., Park, R.H.: A comparative performance study of several global thresholding techniques for segmentation, Computer Graphics and Image Processing, vol. 52 (1990) 171-190.
24. M. Loog, B.v. Ginneken, R.P.W. Duin: Dimensionality reduction by canonical contextual correlation projections. In: Pajdla, T., Matas, J. (Eds.), Lecture Notes in Computer Science 3021. Springer, Berlin (2004) 562-573.

25. Mayr, E.: Toward a New Philosophy of Biology: Observations of an Evolutionist. Belknap, Cambridge, MA (1988).
26. Mees, C.E.K., James, T.H.: The Theory of the Photograpahics Process. Macmillan, UK (1966).
27. Moody, J., Darken, C.J.: Fast learning in networks of locally-tuned processing units, Neural Computation 1/2 (1989) 281-294.
28. Orłowska, E.: Semantics of Vague Concepts. Applications of Rough Sets. Institute for Computer Science, Polish Academy of Sciences Report 469, March (1982).
29. Pal, N.R., Pal, S.K.: A review of image segmentation techniques, Pattern Recognition, 26/9 (1993).
30. Pawlak, Z.: Classification of Objects by Means of Attributes. Institute for Computer Science, Polish Academy of Sciences Report 429, March (1981).
31. Pawlak, Z.: Rough Sets. Institute for Computer Science, Polish Academy of Sciences Report 431, March (1981)
32. Pawlak, Z.: Rough sets, *International J. Comp. Inform. Science*, 11 (1982) 341-356.
33. Pawlak, Z.: *Rough Sets. Theoretical Reasoning about Data*, Theory and Decision Library, Series D: System Theory, Knowledge Engineering and Problem Solving, vol. 9. Kluwer Academic Pub., Dordrecht (1991).
34. Peters, J.F.: Approximation space for intelligent system design patterns. *Engineering Applications of Artificial Intelligence*, 17(4), 2004, 1-8.
35. Peters, J.F., Skowron, A., Synak, P., Ramanna, S.: Rough sets and information granulation. In: Bilgic, T., Baets, D., Kaynak, O. (Eds.), Tenth Int. Fuzzy Systems Assoc. World Congress IFSA, Instanbul, Turkey, *Lecture Notes in Artificial Intelligence* 2715. Springer-Verlag, Heidelberg (2003) 370-377.
36. J.F. Peters, Approximation spaces for hierarchical intelligent behavioral system models. In: Keplicz, B.D., Jankowski, A., Skowron, A., Szczuka, M. (Eds.), Monitoring, Security and Rescue Techniques in Multiagent Systems, *Advances in Soft Computing*. Springer-Verlag, Heidelberg (2004) 13-30.
37. Peters, J. F., Borkowski, M.: K-means indiscernibility relation over pixels. In: Tsumoto, S., Slowinski, R., Komorowski, J., Grzymala-Busse, J.W., *Lecture Notes in Artificial Intelligence* (LNAI), 3066. Springer-Verlag, Berlin (2004), 580-535
38. Peters, J.F.: Rough ethology, Transactions on Rough Sets III (2005) 153-174.
39. Peters, J.F., Henry, C.: Reinforcement learning with approximation spaces. Fundamenta Informaticae 71 (2006) 1-27.
40. L. Polkowski, *Rough Sets. Mathematical Foundations*. Springer-Verlag, Heidelberg (2002).
41. Polkowski, L., Skowron, A. (Eds.), Rough Sets in Knowledge Discovery 2, *Studies in Fuzziness and Soft Computing* 19. Springer-Verlag, Heidelberg (1998).
42. Rosenfeld, A.: Image pattern recognition, Proc. IEEE, 69 (1981).
43. Seemann, T.: Digital Image Processing Using Local Segmentation, Ph.D. Thesis, supervisor: Peter Tischer, Faculty of Information Technology, Monash University, Australia, April (2002).
44. Skowron, A.: Rough sets and vague concepts. *Fundamenta Informaticae*, XX (2004) 1-15.
45. Skowron, A., Stepaniuk, J.: Modelling complex patterns by information systems. *Fundamenta Informaticae*, XXI (2005) 1001-1013.
46. Skowron, A., Stepaniuk, J.: Information granules and approximation spaces, in Proc. of the 7^{th} Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU98), Paris, (1998) 1354–1361.

47. Skowron, A., Stepaniuk, J.: Generalized approximation spaces. In: Lin, T.Y.,Wildberger, A.M. (Eds.), Soft Computing, Simulation Councils, San Diego (1995) 18-21.
48. Skowron, A., Swiniarski, R., Synak, P.: Approximation spaces and information granulation, Transactions on Rough Sets III (2005) 175-189.
49. Sahoo, P.K., Soltani, S., Wong, A.K.C.: A survey of thresholding techniques, Computer Vision, Graphics and Image Processing, 41 (1988) 233-260.
50. Stepaniuk, J.: Approximation spaces, reducts and representatives, in [41], 109–126.
51. Szczuka, M.S., Son, N.H.: Analysis of image sequences for unmanned aerial vehicles. In: Inuiguchi, M., Hirano, S., Tsumoto, S. (Eds.), Rough Set Theory and Granular Computing. Springer-Verlag, Berlin (2003) 291-300.
52. WITAS project homepage: `http://www.ida.liu.se/ext/witas/`
53. Yang, G.Z., Gillies, D.F.: Computer Vision Lecture Notes, Department of Computing, Imperial College, UK, 2001. See `http://www.doc.ic.ac.uk/ gzy`.
54. Zhang, Y.J.: Evaluation and comparison of different segmentation algorithms, Pattern Recognition Letters, 18 (1997) 963-974.
55. Zhang, C., Fraser, C.S.: Automated registration of high resolution satellite imagery for change detection, Research Report, Department of Geomatics, University of Melbourne (2003).
56. Zhang, Y.J., Gerbrands, J.J.: Objective and quantitative segmentation evaluation and comparison, Signal Processing, 39 (1994) 43-54.
57. Zitová, B., Flusser,J.: *Image Registration Methods: A Survey*, Image and Vision Computing 21 (2003).