Morphological Operations for Color–Coded Images

Christoph Busch and Michael Eberle

Computer Graphics Center
Wilhelminenstraße 7, D–64283 Darmstadt, Germany
Tel: +49 6151 155 236, Fax: +49 6151 155 480
E–mail: busch@igd.fhg.de

Abstract

The subject of this paper is the semantically based postprocessing of color–coded images such as classification results. We outline why the classical definition of mathematical morphology suffers if it is used for processing of coded image data. Therefore we provide an extension for morphological operations such as dilation, erosion, opening, and closing. With a new understanding of morphology we introduce bridging and tunneling as further combinations of dilation and erosion. The extensions are applied to medical image data, where the semantic rules stem from basic anatomical knowledge.

Keywords: image analysis, morphological operations, color–coded images, dilation, erosion, opening, closing, bridging, tunneling.

1 Introduction

Morphological operations are widely in use in computer graphics and image analysis applications. They form a well defined system of operators that interact with shapes. The basic mathematical work, which is closely related to the set theory, was done by Hadwiger [7]. A first application on digital images was reported by Kirsch [12], both in 1957. Since then, morphology has played an important part in image analysis in a variety of applications that range from image processing, feature extraction to object recognition and integration into machine vision systems. Application reports can be found in [3], [11] and [14].

The subject of this paper is image post–processing. The intent is an optimization of image data in such a way that noise and other irrelevancies are reduced, but the basic characteristics in the image such as shape and extensions of dominating regions are untouched at the same time. Our plan was the further development of the work of Haralick, Serra et al., who formulated fundamental definitions in [8] and [15]. In general, the definition of morphological operations is in no way limited to a 2D–space as a binary image. The definitions still hold when they are extended and applied to gray–scale images [9] and [14].

Even for color–scale images, such as temperature maps, we will find that the existent definitions are sufficient, since the colors are linearly scaled. Therefore we could simply employ the definitions that were set up for gray–scale morphology. Unfortunately, the existing definitions are not acceptable for color–coded images such as classified images as in [1],[5] or [6]. This insufficiency stems from two substantial deficiencies. On the one hand the colors in color–coded images are not linearly scaled and on the other hand the color itself encodes a semantic meaning. To set up a sophisticated and semantically based processing method we need to incorporate semantic rules and relations in the processing operation. This aspect in combination with our motivating application inspired us to extend the existing definitions. The medical application, which we refer to in Section 4, encodes basic anatomical rules which steer the operation. With the combination of mathematical morphology and corresponding semantic information we can achieve a new understanding of color–coded morphology in terms of a fusion of image–processing and image analysis with one single approach.

This paper is organized as follows: It reviews the foundations of binary and gray–scale morphology in Section 2. The extension on color–coded images will be derived in detail in Section 3. In order to demonstrate the powerful usage of the proposed operations Section 4 shows some experimental results. We conclude with a discussion in Section 5.

2 Foundations of Mathematical Morphology

Mathematical morphology is based on the classical operations dilation and erosion. An effective combination of these simple functions builds up a powerful concept for image processing. These mathematical operations can be applied on sets in the euclidean space Z^N. In the case of binary images a scene can be seen as a set *A* in a two–dimensional space which contains all the foreground pixels. The translation of the scene *A* by the translation vector *v* can be expressed as

$$A_v = \{c \in Z^N \mid c = a + v,\ a \in A\} \tag{1}$$

The morphological operator *B* (structuring element) is a set containing all the translation vectors. Using the above definition the basic operation dilation which is commonly denoted as $A \oplus B$, is defined by

$$\textit{dilation:} \qquad A \oplus B = \{c \in Z^N \mid c = a + b,\ \exists a \in A,\ \exists b \in B\} \tag{2}$$

The morphological opposite to dilation is erosion $A \ominus B$, which is defined as

$$\textit{erosion:} \qquad A \ominus B = \{c \in Z^N \mid (c + b) \in A, \forall b \in B\} \tag{3}$$

With the basic definitions in equation 2 and 3 it is possible to derive the opening and closing operations.

$$\textit{opening:} \qquad A \circ B = (\ A \ominus B\ ) \oplus B \tag{4}$$

The opening of an object *A* with structuring element *B* will select exactly those pixels from object *A* where the positioned structuring element entirely matches in all points. Openings are widely used for the smoothing of binary scenes and removal of background noise that is of smaller extension than the structuring element. The counterpart to opening is a closing of object *A* with a structuring element *B*

$$\textit{closing:} \qquad A \bullet B = (\ A \oplus B\ ) \ominus B \tag{5}$$

which can be thought of as an opening on all background pixels. Closings are used for contour smoothing and removal of noise within a scene. Further morphological operations are explained in detail in [8], [15] and [16].

A remark should be made about the *reference pixel* of a structuring element *B*. The reference pixel is defined as the origin of the coordinate system of structuring element *B*. In this paper, we presume that the reference pixel itself is always a part of the morphological operator. This presumption is made to simplify matters of operation definitions and is of no limitation to a universal validity.

3 Color–Coded Morphology

Color–coded images are used quite frequently in Computer Graphics. One might think of continuously colored images as temperature maps, where blue colored pixels represent low temperature values and red color tones stand for high values. We do not refer to those types of images, rather than that, we understand color–coded images as an image type, in which each color embodies a semantic meaning. It should be pointed out that the distance of two colors in the RGB–space does not necessarily correspond to their semantic relation.

A field where we face color–coded images is in the area of classification applications, where raw data is processed by classification techniques to assign a semantic meaning to each pixel. Texture analysis [5], [10], satellite image analysis for landuse classification [6], or medical image analysis for segmentation purposes of different tissue types are typical applications in that field. When classification tasks are solved with pixel–oriented approaches, one might get noisy result images due to misclassifications.

For the proper understanding of the next subsection, it is essential, to keep the importance of *semantic sensitive processing* in mind. If e.g. we take a tomographic image from the brain and submit it to a postprocessing analysis, we need another treatment of a *liquor* pixel adjacent to *white–* or *grey matter* than for a *liquor* pixel adjacent to a *background* pixel. The latter case does not satisfy the simple semantic rule that all anatomically possible neighbors of a *background* pixel must be either *background* or *fat tissue*.

The postprocessing of a color–coded image could be achieved using three different morphological approaches:

I. Transformation of color–coded images into binary ones.
II. Consideration of color–coded images as gray–scale images.
III. Extension of the morphological operations.

Approach I. is a simple solution, but it is not sufficient since a binary morphological operation as it is defined by equations 2 – 5 would delete information that was contained in the input image.

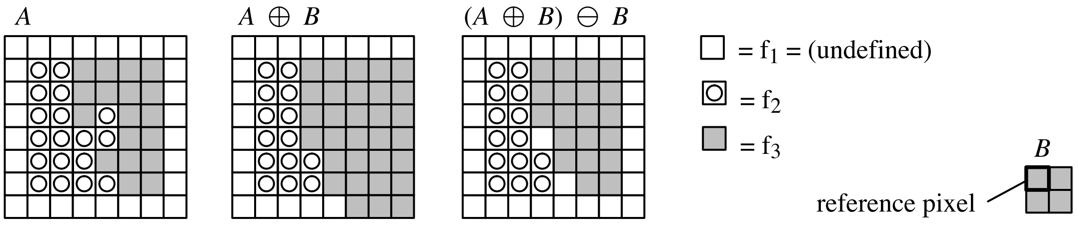

Figure 1. *Closing of a color–coded image with a binary closing on color f_3*

Figure 1 illustrates a binary closing where color f_3 was selected to be the binary foreground. After dilation on f_3 – that is all pixels of object *A* belonging to color f_3 – some pixels of color f_2 are superseded. The following erosion leaves them in the undefined state f_1.

Approach II. uses the original definition in such a way that the color values are understood as the third dimension of an image. In contrary to approach I., we could allow two or more colors per pixel but we still could not fulfill the required *semantic sensitive* behavior of our operations. Therefore, we choose approach III. extending the classical definitions in the next subsections.

3.1. Color–Coded Images

Now, with the understanding of the previous subsection, we can lay out some basic definitions for color–coded images. First of all we define an application–dependent color space *F* as the set containing all *k* possible color values.

color space: $$F = \{f_1,\ f_2,\ \dots, f_k\} \qquad (6)$$

We define a color set $\mathfrak{F}$ of all possible subsets of *F*.

color set: $$\mathfrak{F} = \{\{\ \},\ \{f_1\},\dots,\{f_k\},\ \{f_1,\ f_2\},\dots\dots,\{f_1,\ f_2,\ \dots,f_k\}\} \qquad (7)$$

Note that the empty set itself is an element of $\mathfrak{F}$. We can now define a color–coded image as an object A, which is a subset $A \subseteq Z^N$ and an associated function $\mathcal{A}$, which defines a projection from A to the set $\mathfrak{F}$:

color–coded image A / object A: $$A \subseteq Z^N,\ \mathcal{A} : A \rightarrow \mathfrak{F} \qquad (8)$$

According to this definition, function $\mathcal{A}(a)$ assigns a color–code to each pixel *a*. The color–code – simply named *color* – is one element from the color set $\mathfrak{F}$. The color itself might consist of an arbitrary number of color values $f_1, f_2, \dots$ from the color space.

For simplification of our notation, we will denote in the following *A(x)* instead of $\mathcal{A}(x)$.The color–code for the reference pixel is defined as *A(0), B(0), etc*. According to the remark in Section 2 we require, that the reference pixel is always an element of a structuring object. A special case of a color–coded object is a monochrome object, which has the same color for all elements.

monochrome object A:

$$A(a) = A(0)\ \forall\ a \in A \quad \wedge \quad A(0) = f_i\ ,with\ i \in \{\ 1,2,...,k\} \qquad (9)$$

For a clear description of the impact of a structuring element *B* on image *A* we define

$$\textit{object B at pixel p covers pixel q in image A:} \qquad \exists b \in B\ with\ p + b = q \qquad (10)$$

The definition of a color–code for a pixel requires a specific understanding of matching. Therefore, we define a *match* between two object elements as

$$\textit{pixel a matches pixel b:} \qquad A(a) \cap B(b) \neq \{\} \qquad (11)$$

In this case, the elements *a* and *b* have at least one common color value from the color space in their colors. If object *B* is moved with its reference pixel to position *a,* and condition 11 is fulfilled for all elements *b* out of *B* and their corresponding *covered* pixels in *A,* then it is a *match* between object *B* and object *A* :

$$\textit{object B matches at pixel a to object A:} \qquad B(b) \cap A(a + b) \neq \{\}\ \forall\ b \in B \qquad (12)$$

Operating with structuring element *B* on image *A* we can differ between two morphological operation modes, either *superseding* or *transparent*. The characteristics of *superseding* operations are similar to those of binary operations. A pixel *b* with an assigned color *B(b)* will supersede the previous color *A(a),* if *B* covers *a*. A *superseding* mode will be denoted with subindex o. The contrary *transparent* operation will preserve the information that was set in image *A*. In other words, if *B* covers pixel *a* the resulting color in the result image *C* will be a union of both previous colors

$$\textit{transparent operation:} \qquad C(a) = A(a) \cup B(b) \qquad (13)$$

We call it transparent because the previous color is still contained in the result and can be accessed after additional operations. A transparent operation will be denoted with a subindex t.

3.2. Dilation

Similar to binary operations, the dilation is one of the two basic operations in color–coded morphology. The definition from equation 2 is now extended for the usage in color–coded images as introduced with equation 8 . We define the transparent version of dilation formally as

$$C = A \oplus_t B = \{c \in A\} \text{ and} \qquad (14)$$
$$C(c) = A(c) \cup \bigcup_{a \in A,\ b \in B,} \{B(b) \mid (A(a) \cap B(0)) \neq \{\},\ a + b = c\}$$

To express equation 14 , a transparent dilation $C{=}A \oplus_t B$ consists of all pixels $c \in A$. Their corresponding color *C(c)* is a union of the original color *A(c)* and the union of all colors *B(b)* if and only if there exists a pixel $a \in A$ and an element $b \in B$, such that the coordinate *a* translated with coordinate *b* equals *c* and furthermore the intersection of *A(a)* and *B(0)* is not the empty set.

We admit that the above definition is difficult to follow and more than that it doesn't express how a color–coded operation works. Since a morphological operation can be imagined as an object that moves across an input image and, under certain conditions, stops at selected coordinates and computes the result, we should rather define the operations in pseudo–code, which allows a better understanding of the underlying process (All keywords will be in bold print). The transformed equation 14 in pseudo–code is more convenient for practical application and implementation.

transparent dilation:

```
C = A ⊕_t B:                                                                  ( 15 )
C = A          /* initialization of result C with a copy of input image A */
for all a ∈ A and A(a) ∩ B(0) ≠ {} do        /* reference pixel of B matches a */
  for all b ∈  B do                          /* for all points of  B */
       C(a+b) = C(a+b)  ∪ B(b)               /* add new color values to color */
```

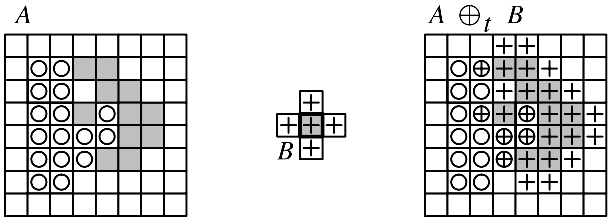

Figure 2. *Transparent dilation with a color–coded structuring element*

For the superseding version of dilation, the color of structuring element *B* will replace a color that was originally set at a covered position in *A*

superseding dilation:

```
C = A ⊕_o B:                                                                  ( 16 )
C = A
for all a ∈ A and A(a) ∩ B(0) ≠ {} do        /* reference pixel of B matches a */
  for all b ∈  B do                          /* for all points of  B */
       C(a+b) = B(b)
```

with associated definition area for structuring element B:

$$\{B \mid B \text{ is object, } B(\mathrm{i}) = B(\mathrm{j}) \; \forall \; \mathrm{i,j} \in B\} \tag{17}$$

Differing from the transparent dilation, the superseding dilation defined by equation 16 might repeatedly change the color of a pixel. The result image *C* would then depend on the sequence of access in *A*. In order to eliminate this dependency we restrict with 17 the definition area of structuring element *B* such that all elements $b \in B$ are assigned to the same color. Note that this does not mean that *B* is a monochrome object.

The definition of color–coded operations includes a powerful instrument for realizing *semantic sensitive processing* if the operations themselves are sensitive. Following this concept, we require that any operation can only add or supersede color values to the color of a pixel *a* if the actual color of that pixel *A(a)* does not include any writeprotected color value.

sensitive transparent dilation:

```
C = A ⊕_s(G),t B;  with G ⊂ F:                                                ( 18 )
C = A
for all a ∈ A and A(a) ∩ B(0) ≠ {} do        /* reference pixel matches a */
  for all b ∈  B do                          /* for all points of B */
         if ( A(a+b) ∩ G = {} )              /* if the covered pixel is not protected */
            C(a+b) = C(a+b)  ∪ B(b)
```

The writeprotected color values are elements of a color subspace G ⊂ F. We can define sensitive instances of the dilation versions and denote them with an index s*(G)* that expresses the sensitivity to a protected color space. The corresponding *sensitive superseding dilation* is defined straightforwardly by replacing the union

C(a+b)=C(a+b)∪B(b) with an explicit assignment C(a+b)=B(b) with the color of the covering element. Figure 3 illustrates an example for a sensitive superseding dilation on an image that contains one protected color.

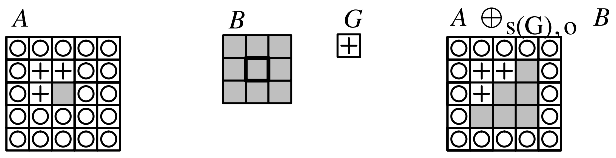

Figure 3. *Sensitive superseding dilation of image* A

In some applications it might be desired to include a certain condition, for example, the correlation with a second image. In this case, both versions of dilation (15 and 16) are extended in a way that the condition is defined with image *C*.

conditional transparent dilation:

```
D = A ⊕c,t B | C:                                          ( 19 )
D = A
for all a ∈ A and A(a) ∩ B(0) ≠ {} do      /* reference pixel matches a */
    for all b ∈ B do                       /* for all points of B */
        if ( B(b) ∩ C(a+b) ≠ {} )          /* if b matches c=a+b */
            D(a+b) = D(a+b) ∪ B(b)
```

It should be mentioned that several applications require iterative processing with a conditional dilation in order to recover regions that were removed due to previous morphological operations – a method that was used on binary images by [16].

3.3. Erosion

Although we consider erosion as the morphological opposite to dilation this does not hold true for all varieties of instances. The transparent and superseding version are defined as

transparent erosion:

```
C = A ⊖t B:                                                ( 20 )
C = A
for all a ∈ A and A(a) ∩ B(0) ≠ {} do      /* reference pixel matches a */
    if ( for all b ∈ B : A(a+b) ∩ B(b) ≠ {} ) /* structuring element matches */
        C(a) = C(a) ∪ B(0)
    else
        C(a) = C(a) \ B(0)
```

superseding erosion:

```
C = A ⊖o B:                                                ( 21 )
C = A
for all a ∈ A and A(a) ∩ B(0) ≠ {} do         /* reference pixel matches a */
    if ( for all b ∈ B : A(a+b) ∩ B(b) ≠ {} ) /* structuring element matches */
        C(a) = B(0)                           /* superseding with new color */
    else
        C(a) = C(a) \ B(0)
```

Using the transparent erosion the basic difference to 21 is that the image position that is covered by the reference pixel of the structuring element will still keep its previous color. In order to shorten our definitions, we will formulate the following instances of erosion just for the transparent version. The superseding version may be derived by simply replacing C(a) = C(a) ∪ B(0) with C(a) = B(0) in the color assignment of the definition.

For the purpose of sensitive operations the sensitive transparent erosion is defined as

sensitive transparent erosion:

```
**C = A $\ominus_{s(G),t}$ B; with G $\subset$ F:**                                   ( 22 )
C = A
for all a $\in$ A and A(a) $\cap$ B(0) $\neq$ {} do              /* reference pixel matches a */
    if ( (for all b $\in$ B : A(a+b) $\cap$ B(b) $\neq${})         /* structuring element matches */
        and A(a) $\cap$ G = {} )                             /* the covered pixel is unprotected */
            C(a) = B(0)
    else
            C(a) = C(a) \ B(0)
```

The corresponding superseding version is denoted as $C = A \ominus_{s(G),o} B$; with $G \subset F$. The sensitive closing operation, to be introduced in Section 3.6., will use both versions of sensitive erosion.

If one works with color–coded operations, the shape that is encoded by an erosion operator will soon be considered to be too strong or too precise. The matching condition of a structuring element might be fulfilled only in a limited number of cases. For example, if we intend to determine a region of a specified color, we will not match those pixels which either are close to the border of the region or which contain a "misclassified pixel" in their environment. Both cases get even more meaning when the operation is done on three dimensions like with a classified tomography dataset. The described negative impact can be reduced with a modified *relative* definition of the erosion. Let *card (B)* be the number of elements in operator *B,* then we define the

relative transparent erosion:

```
**C = A $\ominus_{r(n),t}$ B; with n $\in$]0;1]:**                                   ( 23 )
C = A
for all a $\in$ A and A(a) $\cap$ B(0) $\neq$ {} do              /* reference pixel matches */
   if ( for n $*$ card B or more b $\in$ B : A(a+b) $\cap$ B(b) $\neq$ {} )        /* a defined
                     percentage of pixels matches at covered pixelposition? */
          C(a) = C(a) $\cup$ B(0)
   else
          C(a) = C(a) \ B(0)
```

Relative transparent or superseding erosions enable us to achieve a more fuzzy description of the matching condition, as for example: "The structuring element *B* matches at position *a*, if a percentage *n* of the neighbored pixels match the corresponding pixels". The percentage *n* is given with respect to the cardinality of the structuring element B. If n is set to 1, the relative erosion is equivalent to the definition of erosion in 20 .

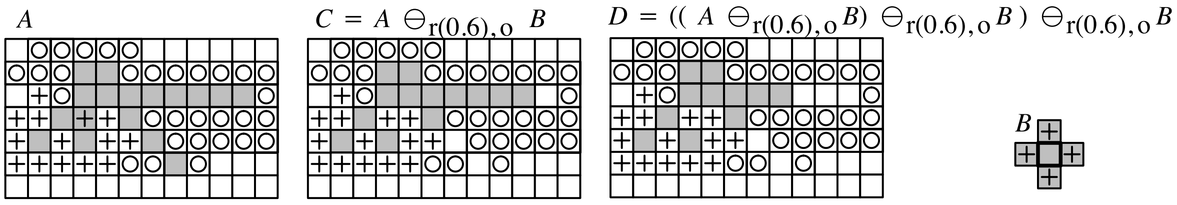

Figure 4. *Relative erosion – a single and iterative application. The matching condition must be fulfilled for at least 60% of the structuring element* B.

3.4. Opening

With our definitions in 3.2. and 3.3. we combine erosion and dilation for a linked color–coded morphological operation. Corresponding to the binary standard our aim is to smooth regions. Thus, in the first attempt we define a *transparent opening* as a *transparent erosion* followed by a *transparent dilation* with the same element, following equation 4 . This intends to achieve an operation that preserves all pixels in *A* under the condition that structuring element B matches at pixel a, which is covered by the reference pixel. Considering an example in Figure 5 we realize that D is not the desired result. An additional operation is required, which intersects object D with the input image A.

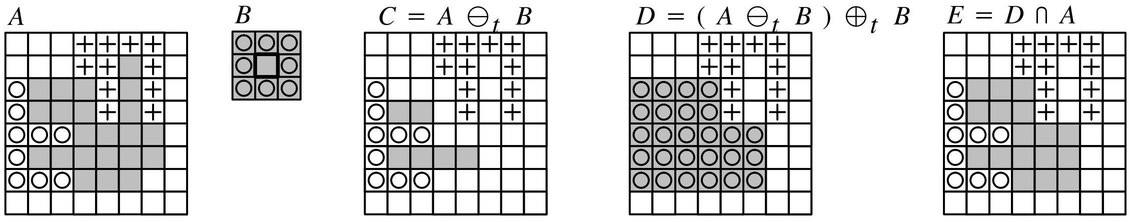

Figure 5. *Transparent opening includes a final intersection with input image* A.

The intersection of D with A results in image E whose characteristics correspond to our intention. Thus we define a transparent opening as

transparent opening: $$A \circ_t B = ((A \ominus_t B) \oplus_t B) \cap A \qquad (24)$$

which includes a final intersection with the input image. The operation smoothes the contour that corresponds to the region of the reference pixel color. A superseding version of 24 is not given here, since it can not be derived in a consistent form. If we defined the superseding operation to be a transparent erosion followed by a superseding dilation we would receive with $F = (A \ominus_t B) \oplus_o B$ an object that would have lost some of the information about the original image. As a consequence, the intersection with the input image will end in the empty set for the color definition of an unbounded number of pixel positions. We should make a note of the identity of transparent and superseding opening for a limited number of structuring elements that have the same monochrome color for all pixels.

An interesting variation to equation 24 can be derived if the envolved erosion that detects the region assigned to the reference pixel's color is encoded in a relative way.

relative transparent opening: $$A \circ_{r(n),t} B = ((A \ominus_{r(n),t} B) \oplus_t B) \cap A \qquad (25)$$

3.5. Closing

The counterpart in combining dilation and erosion defines the closing of a color–coded image in its transparent and superseding versions

transparent closing: $$A \bullet_t B = (A \oplus_t B) \ominus_t B \qquad (26)$$

superseding closing: $$A \bullet_o B = (A \oplus_t B) \ominus_o B \qquad (27)$$

Note that for equation 26 and 27 we must restrict the definition area from structuring element B to achieve a proper working operation. Thus, we require a monochrome structuring element B that has the same color for all pixels. The closing behavior is then similar to a binary closing, as Figure 6 illustrates.

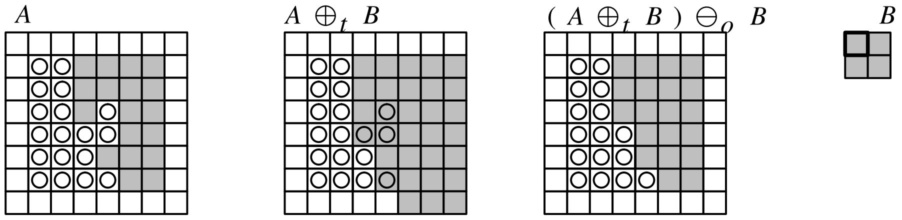

Figure 6. *Superseding closing: one should compare the result with Figure 1.*

Furthermore, we define a *sensitive closing* operation with

sensitive transparent closing: $A \bullet_{s(G),t} B = (A \oplus_t B) \ominus_{s(G),t} B$ (28)

sensitive superseding closing: $A \bullet_{s(G),o} B = (A \oplus_t B) \ominus_{s(G),o} B$ (29)

that protects all pixels in *A* which have common color–values with protection set $G \subset F$. Note that in definition 28 and 29 the erosion, not the dilation was attributed with a sensitive operation mode. The usefulness of this choice is easy to prove. The definitions of the relative versions of 26 and 27 are straightforward

relative transparent closing: $A \bullet_{r(n),t} B = (A \oplus_t B) \ominus_{r(n),t} B$ (30)

relative superseding closing: $A \bullet_{r(n),o} B = (A \oplus_t B) \ominus_{r(n),o} B$ (31)

Again, the relativity which describes the matching percentage of the structuring element is assigned to the erosion. The rough definition implies a fuzzy behavior of the operation. Compared to the standard closing version, there might be a fuzzy extension of closed regions, more than that, the smoothing behavior will change with relativity.

3.6. Bridging

The definitions in the previous subsections were basically derived from their binary analogies. If we allow different operators for combined morphological operations, we open further potentialities. This subsection will show that different operator colors perform sensitive operations without an integration of a protected color space. With the definition of *bridging* which is denoted by $A \boxed{\frown} (B_1,B_2)$, the protection rule is then embedded in the structuring elements.

The purpose of bridging is as follows. We consider a color–coded image with a region associated to a color with color value f_1. Enclosed in this region are noisy pixels with different colors. Those noisy colors are, on the one hand, related to color value f_2, which is considered to be semantically possible, and, on the other hand, related to color value f_3 that stems from misclassification. The intention is to replace all color–values f_3 at enclosed pixel positions with color value f_1. This process will now set up a ”bridge” over pixels of type f_3 to the next pixel from kind f_1. We define a transparent bridging

transparent bridging:

```
D = A $\boxed{\frown}_t$ (B1, B2):                                                     ( 32 )
D = A                  /* copy the input data */
C = A $\ominus_o$ B1          /* determine all pixels that fulfill the ”semantic” rule defined by B1 */
      for all c $\in$ C and C(c) $\cap$ B2(0) $\neq$ {} do        /* transparent dilation for select pixels */
        for all b $\in$ B2 do
              D(a+b) = D(a+b) $\cup$ B2(b)               /* combine with saved input */
```

as a combination of a detective erosion with an additional dilation. The erosion implements the semantic rule, since only unprotected color values are set in all pixel positions that are neighbored to the reference pixel of structuring element B_1. To fulfill our purpose, we replace in definition 32 the transparent dilation by a superseding dilation and obtain a *superseding bridging* that will be denoted by $A \boxed{\frown}_o (B_1,B_2)$. Figure 7 illustrates in which way the superseding bridging fulfills our purpose. Enclosed pixels from type f_3 are replaced. Although the notation of a bridging resembles an opening operation, its behavior is more related to closing. A similar result could have been obtained with a sensitive closing with structuring element *D*.

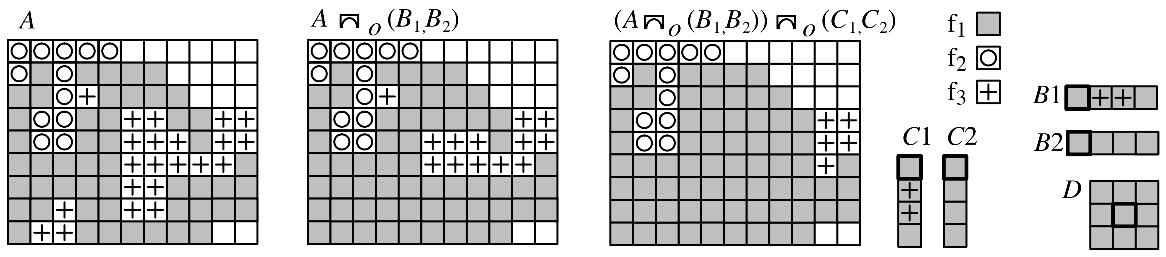

Figure 7. *Superseding bridging*

The proper choice between bridging functionality and an alternative sensitive closing needs to be carefully evaluated. One should definitely consider shape and extension as characteristics of the enclosed noise.

3.7. Tunneling

While bridging is related to closing, tunneling, as a counterpart, is more related to opening. As outlined in Section 3.4., the opening operation fails when the input image is too noisy. Even large regions of one color might disappear after an opening operation if there are isolated enclosed pixels of a different color. This impact can be avoided, if the enclosed pixels are replaced temporarily by a closing operation on the actual color. Therefore, we define the tunneling operation denoted by $A \triangle_o(B_1,B_2)$ as

transparent tunneling:

```
$E = A \triangle_t (B_1, B_2)$:                                           ( 33 )
    $C = A \bullet_t B_1$        /* transparent closing with $B_1$ for temp. removal of enclosed pixels */
    $D = C \circ_t B_2$          /* the opening with $B_2$ extracts the region */
    $E = A \cap D$               /* recovers the enclosed pixels */
```

superseding tunneling:

```
$D = A \triangle_o (B_1, B_2)$:                                           ( 34 )
    $C = A \bullet_t B_1$        /* transparent closing with $B_1$ for temp. removal of enclosed pixels */
    $D = C \circ_t B_2$          /* the opening with $B_2$ extracts the region */
    $E = D \ominus_o B_2(0)$     /* superseding erosion with reference pixel of $B_2$ */
    $F = A \cap E$               /* recovers the enclosed pixels */
```

The definition of the superseding tunneling includes a "superseding opening" that was not defined in Section 3.4. It is realized as a transparent opening and an additional superseding erosion with a structuring element that is just the monochrome reference pixel of the opening operation.

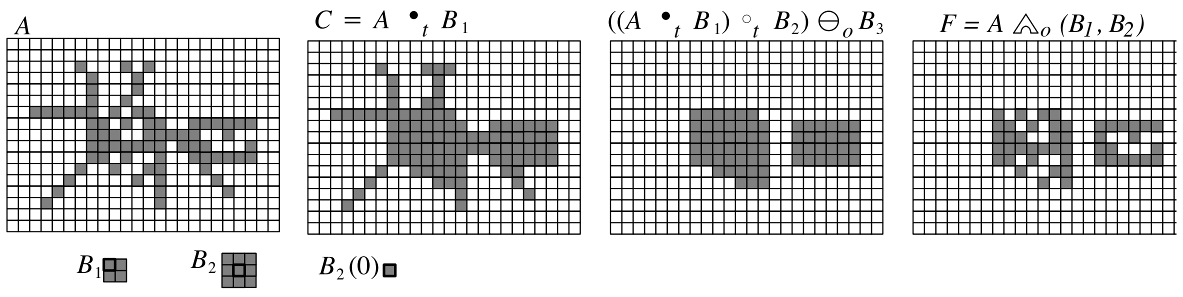

Figure 8. *Superseding tunneling*

Figure 8 illustrates a superseding tunneling. In that case, an opening with B_2 would delete all pixels in A. The tunneling operation preserves the "main regions". The operation can be interpreted in a way that the surviving pixels were able to build imaginary tunnels to their neighbors. Isolated pixels with no neighbors are removed by the operation.

4 Results

The results presented in this section stem from a medical application where 3D image data were analyzed by artificial neural networks [1] in order to segment different tissue types from the raw data. The specific interest lies in detection of pathological tissue, such as a brain tumor in Figure 9.

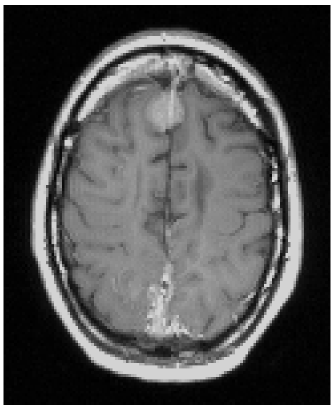
Figure 9. *MR image slice with two meningioma*

The data presented there stem from a brain cancer patient affected by two meningioma. The data were recorded with MRI techniques at a resolution of approximately 1mm in the image plane and 4mm in the z–plane.

The classification result in Figure 10(left) was computed by an artificial neural network that was trained on comparable pathologic data. Due to the limited resolution of the acquisition technique, the underlaying anatomy was mapped with a rather rough resolution. The classification result is a color–coded image where the color–codes represent anatomical classes.

It should be noted that the neural net considered pixels as *undefined* if their computed features did not correlate to any of the anatomical classes. Furthermore, it must be stated that the class *bone* is seldom found in the result images. This stems from the fact that bone has almost no signal in a MRI image and thus was underrepresented in this training set.

It would be beyond the scope of this paper to consider all details of anatomical rules and their transformation into color–coded morphological operators. Nevertheless, with the following table, we want to present some of the basic rules that define the semantic relationships between two color–codes.

color–code		**semantically possible neighbors**
tumor	(red)	{tumor, liquor, grey matter, white matter}
liquor	(blue)	{liquor, tumor, grey matter, (white matter), bone}
white matter	(light green)	{white matter, tumor, (liquor), grey matter}
grey matter	(dark green)	{grey matter, tumor, liquor, white matter}
fat tissue	(orange)	{fat tissue, background, bone}
background	(turquoise)	{background, fat tissue}
bone	(grey)	{bone, liquor, fat tissue}

We briefly sketch the operations that computed the morphologically processed images in Figure 11(right). The operations are defined with either 2D– or 3D–structuring elements. In general the operators are elliptically shaped and embody the above rules. We assume that our classified input image is not affected by noise in the background, which will not be the case in general. The first step is an operation that removes isolated misclassified pixels in the scene of Figure 10(left) as they exist for grey matter, white matter, liquor, tumor, and fat tissue. Figure 10(right) shows the result that was reached with a relative transparent opening. The associated relativity value is chosen with respect to the specific color code and to the sequence of processing. The second step smoothes the contours and closes the regions. We start with a closing on the tumor regions. Then a sensitive closing is performed on each of the following anatomical classes: fat, white matter, grey matter, liquor. The set of protected color values is the set {tumor, fat, white matter, grey matter, liquor} without the color of the actual single sensitive closing step. The output of this smoothing procedure, as it is illustrated in Figure 11(left), contains a number of undefined regions. They origin either in undefined input from the classifier, or they result from the removal of isolated pixels. Thus, we finish with a limited filling of undefined regions. This can be achieved by alternating sensitive dilations on the anatomical classes.

5 Conclusion

The extension of morphological operations to color–coded capability has proved to be a powerful approach for postprocessing analysis. The examples in Section 3 and the medical application in Section 4 demonstrate the general performance of our operations. The optional sensitivity of the operations enables us to embed semantic meaning into the operations. Furthermore the optional color coding of structuring elements offers the opportunity to realize semantically based neighboring relations. Relative operations have shown to be an adequate method to control the matching of structuring elements to a semantic environment.

Future work should be directed to the algebraic aspects of the proposed color–coded extension. At some points, algebraic rules can be just taken over from their binary prototypes, such as the reflection or the intersection of two sets. Others, like the complement of a set, need to be redefined. Furthermore, due to the new definitions, the associative or extensive behavior of operations requires further investigation. Finally, the definite loss of binary advantages like the erosion dilation duality theorem has to be compared to the power of the new operations.

6 Acknowledgement

This research was made possible by the financial support from Deutsche Telekom as a part of the project KAMEDIN. Further thanks to the department of Biophysics and Medical Radiation Physics of the German Cancer Research Center which provided us with the MRI data set that was used in this work.

7 References

1. C. Busch, M. Gross: ”Interactive Neural Network Texture Analysis and Visualization for Surface Reconstruction in Medical Imaging”. Computer Graphics Forum, Vol.12, No.3, pp.C49–C60, (1993)
2. M. Eberle: ”Nachbearbeitung klassifizierter Tomographiedaten des menschlichen Gehirns unter Berücksichtigung der Anatomie” Diplom–thesis, Computer Science Department, TH Darmstadt, (1994)
3. F. Gerritsen, L.G.Aardema: ”Design and use of DIP–1: A fast flexible and dynamically microprogrammable image processor” Pattern Recognition, vol.14, pp.319–330, (1981)
4. C.R. Giardina, E.R. Dougherty: ”Morphological Methods in Image and Signal Processing”. Prentice–Hall, (1988)
5. M. Gross, R. Koch, L. Lippert, A. Dreger: ”Multiscale Image Texture analysis in Wavelet Spaces”. Proceeding of the IEEE – International Conference on Image Processing ’94, pp. 412–416, (1994)
6. M. Gross, F. Seibert: ”Neural network image analysis for environmental protection”. In Grützner (Edts.): Visualisierung von Umweltdaten 1991, GI, Berlin – Heidelberg – New York: Springer (1991)
7. H. Hadwiger: ”Vorlesungen über Inhalt, Oberfläche und Isoperimetrie”, Springer Verlag, (1957)
8. R.M. Haralick, S.R. Sternberg, X. Zhuang: ”Image Analysis using Mathematical Morphology”. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI–9 no. 4, pp. 532–550, (1987)
9. K.H. Höhne, W.A. Hanson: ”Interactive 3D–Segmentation of MRI and CT Volumes Using Morphological Operations”. J. Comput. Assist. Tomogr. 16, pp.285–294,(1992)
10. G. Ravichandran, M.M. Trivedi: ”Texture Segmentation using Circular–Mellin Operators”. Proceeding of the IEEE – International Conference on Image Processing ’94, pp. 636–639, (1994)
11. J.C. Klein, J.Serra: ”The texture analyzer”. J. Microscopy, vol. 95, pp. 349–356, (1977)
12. R.A. Kirsch, et al.: ”Experiments in Processing Pictorial Information with a Digital Computer”. Proceedings of the Eastern Joint Conference, pp. 22–229, (1957)
13. H. Minkowski: ”Volumen und Oberfläche”. Math. Ann, vol.57, pp.447–495, (1903)
14. M. Werman, S. Peleg: ”Min–max operators in texture analysis”. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI–7 no. 4, (1985)
15. J. Serra: ”Introduction to mathematical morphology”. Comput. Vision, Graphics, Image Processing, vol. 35, pp. 333–355, (1986)
16. P. Zamperoni: ”Methoden der digitalen Bildsignalverarbeitung” pp. 216–243, Vieweg–Verlag, Braunschweig, (1991)