



Universidad de Concepción
Dirección de Postgrado
Facultad de Ingeniería
Magíster en Ciencias de la Computación



INTERPOLACIÓN MORFOLÓGICA DE IMÁGENES EN
COLOR
(MORPHOLOGICAL INTERPOLATION OF COLOR
IMAGES)

ADRIÁN ADOLFO DEL PINO NAVARRETE
CONCEPCIÓN, CHILE
2014

Profesor Guía: Dr. Javier Vidal Valenzuela
Dpto. de Ing. Informática y Cs. de la Computación
Facultad de Ingeniería
Universidad de Concepción



Acknowledgments

I wish to thank Dr. Javier Vidal for his advice, Dr. John Atkinson for revising the first version of this document, and my family for its support.



To my parents



Abstract

This research aimed to build a morphological method for color inter-frame interpolation with better picture quality than other methods when the input images contain complex connected components.

In theory, it is possible to build new color interpolation methods by using different color morphological operators. However, none of these operators has emerged as a standard. Consequently, another approach was taken to interpolate color images: interpolating the shapes of the objects in the color images, and then coloring the interpolated shapes.

The new color interpolation methods were compared to the linear, and the color median of Iwanowski and Serra methods. Some images were interpolated with the aforementioned methods, and the interpolated images were compared against the ground truth. The new methods frequently obtained better interpolated images.

As a conclusion, it is possible to construct better morphologic interpolation methods for color images without using color morphological operators.



Table of Contents

Acknowledgments	iii
Abstract	v
List of Tables	xiii
List of Figures	xv
List of Algorithms	xxi
Chapter 1 Introduction	1
Chapter 2 Literature Review: Morphological Interpolation of Images	5
2.1 Morphological Interpolation for Binary Images	5
2.1.1 Morphology-based Three-dimensional Interpolation	5
2.1.2 Recursive Interpolation Technique based on Morphological Median Sets	6
2.2 Interpolation of Partitions	9
2.2.1 Interpolation of Partitions through Median Sets	10
2.2.2 Generalized Morphological Mosaic Interpolation	11

2.2.3	Segmentation-based Morphological Interpolation of Partition Sequences	11
2.2.4	A Region-based Interpolation Method for Mosaic Images	14
2.3	Morphological Interpolation for Grey-scale Images	18
2.3.1	Median of Images	19
2.4	Color Median of Images	20
Chapter 3	New Morphological Interpolation Methods for Color Images	23
3.1	Segmenting the Input Images to Obtain Color Mosaics	25
3.2	Matching the Regions of the Color Mosaics	29
3.3	Converting the Color Mosaics with Matched Regions into Grey-level Mosaics	30
3.4	Interpolating the Grey-level Mosaics	32
3.5	Coloring the Interpolated Mosaic	34
3.5.1	Overlapping	35
3.5.2	Deforming	39
3.6	The Algorithm for Compression and Expansion	42
3.6.1	Compression and Expansion between the Borders and the Nucleus	43
3.6.2	Birthplaces	51
Chapter 4	Comparisons of Color Interpolation Methods	55
4.1	Selecting Sequences of Images	55
4.2	Statistical Analysis	62
4.3	Results and Discussion	63
4.4	Descriptive Analysis	75
Chapter 5	Conclusions	79
Appendix A	Images	81
A.1	Properties of Color	81

A.1.1	Color Constancy	82
A.2	Digital Images	83
A.2.1	Binary Images	83
A.2.2	Grey-level Images	83
A.2.3	Color Images	84
A.2.4	Mosaic Images	84
A.3	Color Models and Color Spaces	86
A.3.1	RGB Color Model	87
A.3.2	HSV Color Model	88
A.3.3	CIELAB	88
A.3.4	YIQ Color Space	89
A.4	Basic Concepts: Neighborhood, Neighbors, Connected Pixels, and Connected Component	90
A.5	Convolution	91
A.6	Image Gradient	92
A.6.1	Scharr Operator	95
A.6.2	Kroon Operator	96
A.7	Distance	96
A.7.1	Geodesic Distance	97
A.7.2	Hausdorff Distance	98
A.8	Homotopy	98
A.9	Image Segmentation	99
A.9.1	Interactive Segmentation	104
A.10	Image Interpolation	105
A.10.1	Grey-level Image Interpolation	106
A.10.2	Shape-based Interpolation	110
A.11	Image Quality Assessment	110
A.11.1	FSIMc	111

Appendix B	Mathematical Morphology	115
B.1	Mathematical Basis	115
B.2	Structuring Element	116
B.3	Binary Operators	118
B.4	Ultimate Eroded Set	120
B.5	Hit and Miss Transform	121
B.6	Thinning	122
B.7	Shrinking	123
B.8	Skeletonization	123
B.9	Pruning	128
B.10	MSP	129
B.11	Interpolation of Sets	129
B.11.1	Median Set	129
B.11.2	Sequence of Interpolations through Median Sets	132
B.11.3	Interpolation Function	132
B.11.4	Interpolations based on Hausdorff Distance	133
Appendix C	Mathematical Morphology for Images	135
C.1	Grey-level Mathematical Morphology	135
C.1.1	Grey-level Watershed Transform	135
C.2	Color Mathematical Morphology	138
C.2.1	Color Watershed	139
Appendix D	Implementing Existent Interpolation Methods for Color Images	141
D.1	Implementing Linear Interpolation for Color Images	141
D.2	Implementing Color Median of Images	142
Appendix E	Prewitt Operator	145
Appendix F	Sobel Operator	147

Appendix G	Mean Squared Error and Peak Signal to Noise Ratio	149
G.1	Mean Squared Error	149
G.2	Peak Signal to Noise Ratio	149
Appendix H	Statistics	151
H.1	p-value	151
H.2	One- and Two-tailed Tests	151
H.3	Significance Level	152
H.4	Statistical Power	153
H.5	Skewness	153
H.6	Normality	155
H.6.1	Normal Probability Plot	156
H.6.2	Shapiro-Wilk Test	157
H.7	Paired Difference Tests	157
H.7.1	t-Test for Correlated Samples	157
H.7.2	Wilcoxon Signed-rank Test	158
H.7.3	Sign Test	160
H.8	Rank Correlation	161
H.8.1	Correlation Coefficient of Kendall	162
Appendix I	Choosing an Interactive Segmentation Method	163
I.1	Measurement of Interactive Segmentation Methods	163
I.1.1	Accuracy	163
I.1.2	Repeatability	165
I.1.3	Efficiency	165
I.2	Datasets for Evaluating Interactive Segmentation Methods	166
I.3	Comparison of Interactive Segmentation Methods	167
Appendix J	Compression and Expansion between the Borders and the Birthplace	169

J.1 Compression and Expansion between the Outer Border and a Set of Points 172

J.2 Compression and Expansion between the Borders and Rings 177

J.3 Compression and Expansion between the Borders and an Artificial Connected Component 182

Bibliography **183**



List of Tables

4.1	Methods applied according to its name	70
4.2	Types of deforming applied to an interpolation according to its name	70
4.3	FSIM _C calculated for several interpolation methods (part I) . .	71
4.4	FSIM _C calculated for several interpolation methods (part II) .	73
4.5	Descriptive statistics of FSIM _C for several interpolation methods	75
A.1	Categories of image segmentation methods	101



List of Figures

2.1	First matching criterion	7
2.2	Interpolation by using the inclusion relationship property . . .	8
2.3	MSP points to calculate the point where X and Y are going to be translated	9
2.4	Partitions with a one-to-one correspondence between cells . . .	10
2.5	A moving ball	14
2.6	Interpolation of Fig. 2.5 after the dead leave step	14
2.7	Interpolation of a sequence	15
2.8	A mosaic divided into regions	16
2.9	Type 1 region and its artificial connected component	17
2.10	Type 2 region and its artificial connected component	17
2.11	Type 4 region and its artificial connected component	17
2.12	Type 6 region and its artificial connected component	18
2.13	Type 7 region and its artificial connected component	18
2.14	Type 8 region and its artificial connected component	18
2.15	Color median image generated from a couple of images	21
3.1	New interpolation method	26
3.2	C1 Segmenting the input images	27

3.3	Initial and final images of dogdance	27
3.4	Initial and final segmented images from dogdance	27
3.5	Initial image of minicooper, and its segmented image	28
3.6	C2 Matching the regions of color mosaics	29
3.7	Initial and final matched segmented images from dogdance	30
3.8	C3 Converting the color mosaics into grey-level mosaics	30
3.9	Initial and final grey-level mosaics from dogdance	32
3.10	C4 Interpolating the grey-level mosaics	32
3.11	Interpolated grey-level mosaic from dogdance	33
3.12	C5 Coloring the interpolated mosaic	34
3.13	Images before and after overlapping	38
3.14	Image interpolated with overlapping	38
3.15	A big and a small regions	42
3.16	The small region overlaps the big one while the big region covers the small one	42
3.17	Interpolated image of the sequence called compact disc by using skeletons and birthplaces	44
3.18	A leg before and after tracing segments	46
3.19	A leg with adjacent and intersected segments, and with seg- ments that leave empty areas	52
4.1	Initial and final images of the sequence called walkcircle	56
4.2	Initial and final images of the sequence called walkstraight	57
4.3	Initial and final images of the sequence called walkstraight_error	57
4.4	Initial and final images of the sequence called army	57
4.5	Initial and final images of the sequence called basketball	58
4.6	Initial and final images of the sequence called beanbags	58
4.7	Initial and final images of the sequence called dogdance	58
4.8	Initial and final images of the sequence called dumptruck	59
4.9	Initial and final images of the sequence called hydrangea	59

4.10	Initial and final images of the sequence called minicooper . . .	59
4.11	Initial and final images of the sequence called wooden	60
4.12	Initial and final images of the sequence called ceramic	60
4.13	Initial and final images of the sequence called compact disc . .	61
4.14	Initial and final images of the sequence called Santa	61
4.15	Initial and final images of the sequence called walkingVGA . .	61
4.16	Initial and final images of the sequence called walking5M . . .	62
4.17	Interpolation considering the dumptruck as a region	64
4.18	Interpolation considering the car, its shade and the enclosed area as separate regions	65
4.19	Interpolation considering only the car as a region	66
4.20	Interpolation considering the car, its shade and the area as a region	67
4.21	Interpolation considering the car and its shade, the enclosed area, and the dumptruck as three separate regions	68
4.22	Comparison of grey-level interpolated mosaics	68
4.23	Table segmented in the wooden sequence	68
4.24	Table in wooden interpolated with overlapping and deforming	69
4.25	Average FSIMc per Interpolation Method	76
4.26	Segmentation of an image	77
4.27	New interpolation method applying only deforming by using birthplaces	78
A.1	Binary image	84
A.2	Grey-level image	84
A.3	Color image	85
A.4	Grey-level mosaic image	85
A.5	Color mosaic image	86
A.6	Common matrix to convert $R'G'B'$ into YIQ	89
A.7	Neighbors of a pixel	90

A.8	Example of convolution	92
A.9	3 x 3 window	93
A.10	Horizontal derivative kernel	93
A.11	Vertical derivative kernel	93
A.12	Geodesic distance	97
A.13	Hausdorff distance sample	98
A.14	Homotopy tree of a set	100
A.15	Ways to signal the ROI	104
A.16	Typical artifacts of linear interpolation methods	106
A.17	Photograph section zoomed with nearest-neighbor interpolation method	107
A.18	Photograph section zoomed with bilinear interpolation method	108
A.19	Linear interpolation of the images	109
B.1	3x3 cross or diamond	117
B.2	3x3 square	117
B.3	5 x 5 cross	117
B.4	5 x 5 diamond	117
B.5	Dilation of a pixel	118
B.6	Dilation of a rectangle	119
B.7	Example of erosion	120
B.8	Erosion of a rectangle	120
B.9	Computing ultimate eroded set with the 3x3 cross	121
B.10	Computing ultimate eroded set with the 3x3 square	121
B.11	Composite structuring element for detecting upper left corners	122
B.12	Comparing shrinking with thinning	124
B.13	Overlapped disks and its skeleton	125
B.14	Different skeletons of a set	128
B.15	Computing median set	130
B.16	Median set from two input sets	131

B.17	Example of first Hausdorff geodesic interpolation	134
C.1	Watershed of an image	136
D.1	Linear color interpolation program	142
D.2	Color median image generation program	143
D.3	calcularInfimoLuminosidad program	143
H.1	Negative and positive skew diagrams	153
J.1	Detail of a man in the grey-level mosaic obtained segmenting the image basketball12.png	171
J.2	Detail of a man in the interpolation of basketball12.png and basketball14.png by using birthplaces	171
J.3	expanding_or_compressing_Image_by_using_birthplace	175
J.4	Segment between the outer border and the MSP	176
J.5	Segment between the outer border and the center	181
J.6	Matched segmentation of the initial and final images of the sequence called beanbags	181
J.7	Interpolated image of the sequence called beanbags by using birthplaces and the linear interpolation	182



List of Algorithms

1	Overlapping algorithm	37
2	Deforming algorithm	41
3	expanding_or_compressing_Image_by_using_skeleton algorithm	48
4	expanding_or_compressing_Image_by_using_skeleton algorithm (continued)	49
5	trace_segments algorithm	53





Introduction

Mathematical morphology is considered a powerful and useful framework to analyze and process images [9, 68, 184]. It has been applied to many areas, such as image interpolation.

Image interpolation has been classified as *one-* and *two-image interpolation* depending on the number of images used as input (one and two images, respectively). The latter interpolation is also called *inter-image interpolation* and *inter-frame interpolation*.

Intermediate pictures are obtained by using two-image interpolation in applications such as X-ray computed tomography and magnetic resonance imaging. In these examples, equally spaced images are taken by the machines (their mechanisms cannot move in very small steps and, additionally, in the tomographs, to avoid excessive radiation exposure) and the missing pictures are constructed by interpolation.

The techniques and methods for two-image interpolation can be classified into grey-level and object-based [21].

Grey-level methods compute the value of an interpolated pixel by using the values of the corresponding neighbor pixels in the original images [35, 215]. Other names

for these methods are scene-based and intensity-based. Unfortunately, none of these names describes these methods accurately. “Scene-based” conveys the idea of methods operating on whole images, but they can operate on parts of them. “Grey-level” and “intensity-based” convey the idea of methods operating only on grey-level images, but they can operate on binary and color images.

Some grey-level interpolation methods are nearest-neighbor, linear, splines and polynomial [25]. The nearest-neighbor is the simplest, and the linear is the most used. To compute the interpolated image between two images, the nearest-neighbor method copies the input image that is near to the place of the interpolated image into the interpolated image. The linear method (another name is cross-dissolve) obtains new images by using weighted combinations of corresponding pixels [196]. In general, they are fast and easy [215]. However, these methods suppose that a smooth curve can model image data, so they make artifacts and blurred edges [35, 215].

Object-based methods use information of the objects in the images to guide the interpolation process [21]. A crucial characteristic of objects is the shape; methods that use it are called shape-based ones [25]. Some shape-based methods employ mathematical morphology since it provides a coherent framework to develop effective algorithms (Serra, 1982, cited in [5]). For example, three methods that influenced this work are described in these papers: “A region-based approach to interpolate images” [29], “A region-based interpolation method for mosaic images” [210], and “Morphological interpolation and color images” [85].

In theory, new color interpolation methods can be built by using different color morphological operators. However, although many approaches have been proposed to apply mathematical morphology to colors, none of them has emerged as a standard [9]. Consequently, an approach was proposed to interpolate color images that does not need color morphological operators.

The *general goal* was to construct a color morphological interpolation method with

better picture quality than other methods when the input images contain complex connected components.

The specific goals were to find an adequate color segmentation method for images that contain complex connected components, to construct a new method for color morphological interpolation that uses the segmented images as input, and to compare it (with other methods).

The methodology considered the following steps: Implement an adequate segmentation method for color images, construct a new method for color morphological interpolation by using the segmented images and the mosaic interpolation algorithm of Vidal et alii [210], and determine the picture quality of the interpolated images.

The rest of this thesis is organized as follows. Chapter 2 reviews the relevant literature about morphological interpolation of images. Chapter 3 describes the proposed color interpolation methods. Chapter 4 compares the new methods against the linear, and color median of Iwanowski and Serra ones. Finally, chapter 5 shows the conclusions.



Literature Review: Morphological Interpolation of Images

Mathematical morphology provides a coherent framework to develop effective shape-based interpolation algorithms (Serra, 1982, cited in [5]). Consequently, it has been applied to binary, grey-level, and color images.

2.1 Morphological Interpolation for Binary Images

Although it is possible to interpolate binary images by extending the interpolation of sets, the disadvantages of these methods are clear (see Sec. B.11). In practice, binary morphological interpolation methods processed the images by considering their components. For example, the *Morphology-based three-dimensional interpolation* (see Sec. 2.1.1) and the *Recursive interpolation technique based on morphological median sets* (see Sec. 2.1.2).

2.1.1 Morphology-based Three-dimensional Interpolation

This method [111] distinguishes two types of components: objects and holes. It computes the interpolated picture by subtracting the interpolated holes from the interpolated objects. It matches the same class of components between images, and

if some component in an image cannot be paired, it creates the same class of component in the other image. These components, with the size of a pixel, are located, for objects, at the centroid of the objects and, for holes, at a point computed with a simple formula that should avoid that a hole be located out of its corresponding object. Finally, it aligns and interpolates the matched components.

2.1.2 Recursive Interpolation Technique based on Morphological Median Sets

Another approach [209] considers the treatment of connected components (CC) belonging to each of the original slices. It distinguishes between CC belonging to the foreground (*grains*) and those belonging to the background (*holes*). A component that surrounds other components is called the *outer* component; and the surrounded components are called the *inner* components. A *filled* component is a component whose holes have been filled. This method considers a filled outer component as a set and its filled inner components as subsets. This recursive method has three steps: (1) extract the outer connected components, (2) match the filled connected components between slices, and (3) interpolate.

The extraction step takes each foreground component that either overlaps (or covers) the border of the image or has a path to the image border along the background.

The matching step determines which foreground components are going to be interpolated in the next step. This is accomplished by computing a *proximity zone* around each filled component. To do this, it is computed the radius λ of each filled component (see Fig. 2.1 (a)). The proximity zone is computed dilating λ the filled component. If this proximity zone overlaps (or cover) a component in the other image (see Fig. 2.1 (c)), the Euclidean distance between their MSPs is computed and stored. Finally, pairs of components whose distance is minimal are stored as matched filled components.

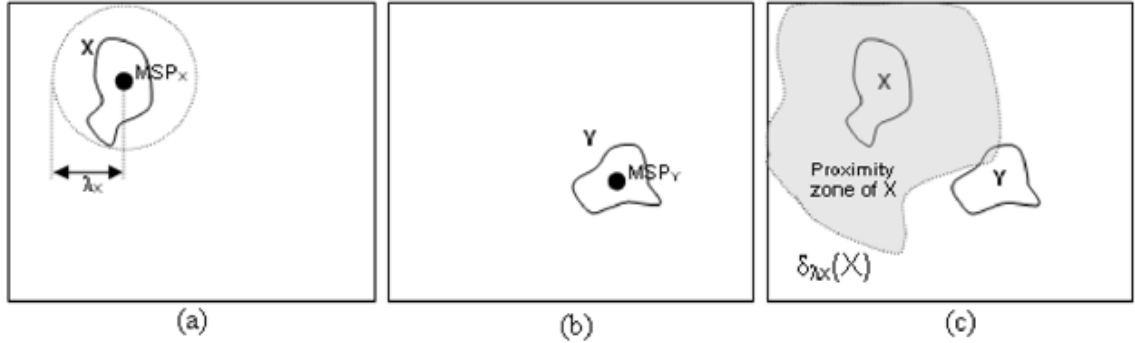


Figure 2.1: First matching criterion: In (a) there is a slice with one CC X indicating its MSP and its radius λ_x ; in (b) there is a second slice with a CC Y ; and in (c) it is shown the proximity zone of X . Since the proximity zone of X overlaps Y , then X matches Y .

The *inclusion relationship property* suggests that the interpolations should satisfy this condition [209, 207]

$$\text{Interpolat}(A_1 \setminus B_1, A_2 \setminus B_2) = \text{Interpolat}(A_1, A_2) \setminus \text{Interpolat}(B_1, B_2),$$

where “Interpolat” is some interpolation method; the sets A_1 and B_1 belong to a slice, with $B_1 \subseteq A_1$; the sets A_2 and B_2 belong to another slice, with $B_2 \subseteq A_2$. It has been shown that the application of this property improves the interpolation results [208]. An example that uses this property is shown in Fig. 2.2.

The interpolating step interpolates each pair of matched components and adds the interpolated component to the interpolated slice. This step fills the matched components, moves the matched filled components to a central position by using their MSPs (see Fig. 2.3, taken from [209]), and interpolates them by using median sets. Then, it computes the holes inside the matched components. If there are holes, it recursively calls this algorithm with these holes, and computes the interpolated component –by using the inclusion relationship property– as the median set “minus” the interpolated holes, otherwise the median set is the interpolated component.

In some unusual cases, this method puts the inner component out of its container.

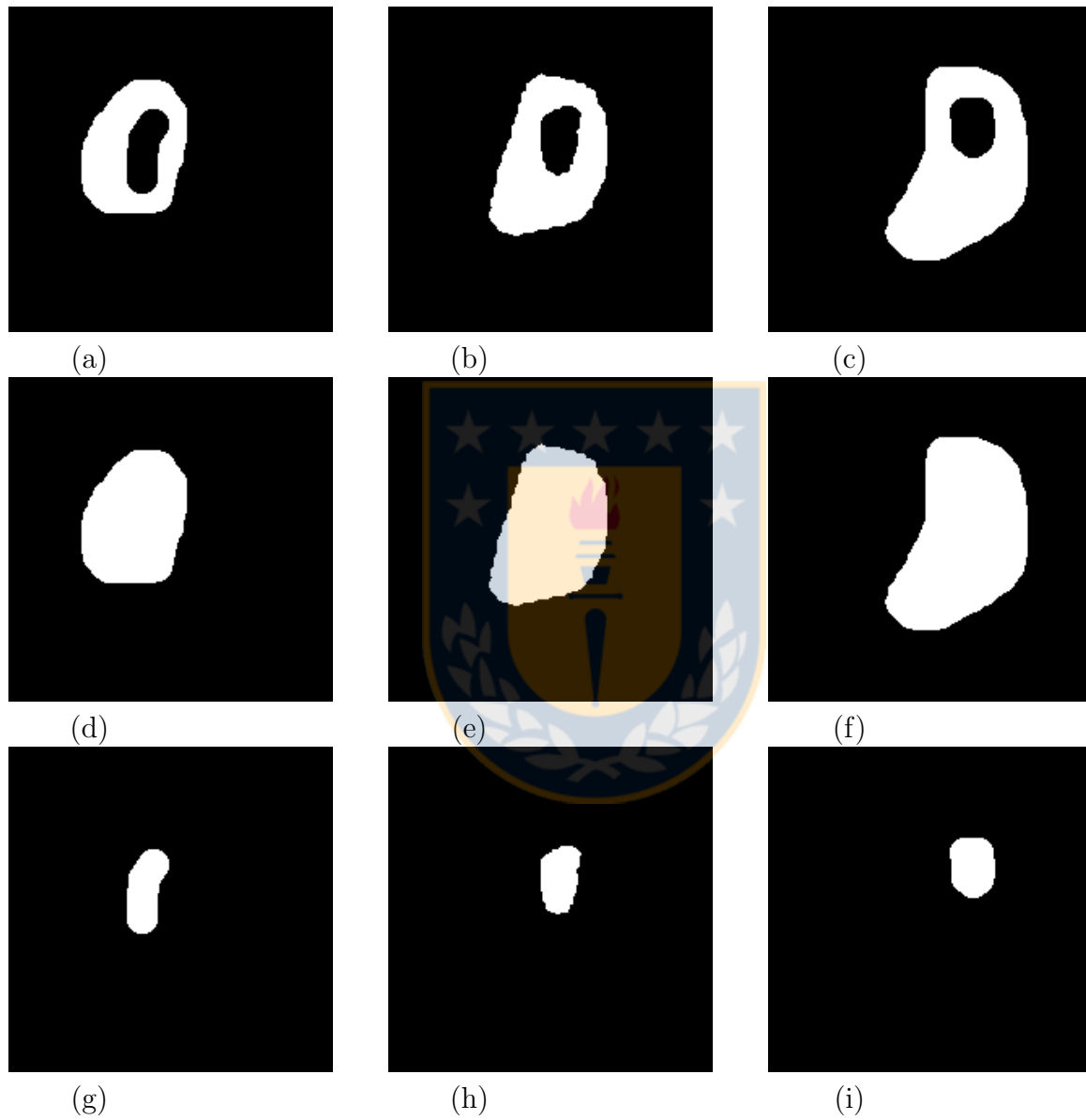


Figure 2.2: Interpolation by using the inclusion relationship property: (a) $A_1 \setminus B_1$, (b) $\text{Interpolat}(A_1 \setminus B_1, A_2 \setminus B_2)$, (c) $A_2 \setminus B_2$, (d) A_1 , (e) $\text{Interpolat}(A_1, A_2)$, (f) A_2 , (g) B_1 , (h) $\text{Interpolat}(B_1, B_2)$, (i) B_2 (taken from [207]).

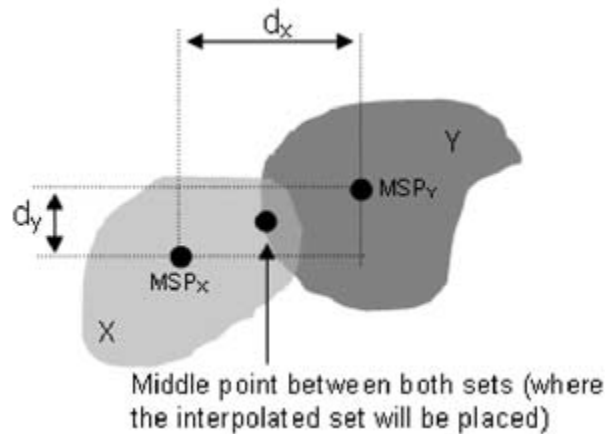


Figure 2.3: MSP points to calculate the point where X and Y are going to be translated

To avoid this violation of the homotopy, the interpolated inner component is relocated and, if necessary, eroded [207].

2.2 Interpolation of Partitions

The extension of morphological interpolation from binary images to grey-level images has led to mosaic interpolation [20, 207]. In other words, the interpolation of partitions has been applied to interpolate grey-level images [20]. In mosaic interpolation, to process the shapes of regions, the regions are usually transformed into binary images [210].

Four approaches are explained: the *interpolation of partitions through median sets*, the *generalized morphological mosaic interpolation*, the *segmentation-based morphological interpolation of partition sequences*, and a *region-based interpolation method for mosaic images*. The first one was indirectly applied in this work as part of the the fourth one; the second one interpolates using geodesic distances; the third one gives an overview of the problems and solutions found interpolating partitions, and the fourth was applied in this work.

2.2.1 Interpolation of Partitions through Median Sets

Beucher [20] proposed interpolating partitions by using median sets. There is a short explanation of this idea below.

Let T and T' be partitions of two same-sized images:

$$T = \{C_i\}, T' = \{C'_i\}$$

There must be a correspondence one-to-one between the cells of these partitions, i.e.

$$\forall C_i \in T, \exists C'_i \in T' : C_i \cap C'_i \neq \phi$$

Let $W = \{C_i \cap C'_i\}$ be the set of all the components $C_i \cap C'_i$. A median partition $M(T, T')$ can then be defined as

$$M(T, T') = \{IZ_W(C_i \cap C'_i)\}$$

In Fig. 2.4 there are two partitions: one is delineated in black; the other, in red. The intersection between cells, W , is painted in yellow.

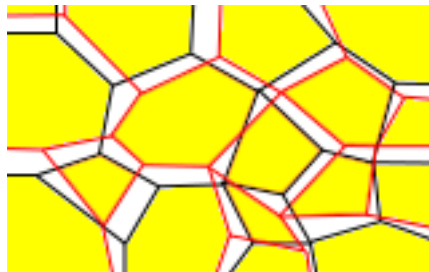


Figure 2.4: Partitions with a one-to-one correspondence between cells: Black lines delimit a partition T ; red lines, another T' .

Two algorithms have been proposed to compute this median: One uses a skeleton by influence zones (SKIZ) and gives a partition with one pixel borders, the other gives a labeled partition without boundaries between the cells [20].

2.2.2 Generalized Morphological Mosaic Interpolation

Meyer [130] proposed an interpolation function between two intersecting sets by using geodesic distances (see Sec. B.11.3); and extended it to mosaics whose regions have a non-empty intersection.

Iwanowski [86] extended the mosaic method from Meyer. In effect, the improved method can treat non-matching regions (regions without a counterpart—a region with the same label—in the other image) and non-intersecting matching regions (regions with the same label and empty intersection). In the first case, a counterpart is created, i.e. a new pixel is obtained by eroding the region—and if necessary, by thinning it— or by calculating its center of gravity—and if this pixel is out of the region, the closest pixel belonging to the region is chosen. In the second case, it applies affine transformations (translation, rotation, and scaling) before the interpolation (described in [87]).

2.2.3 Segmentation-based Morphological Interpolation of Partition Sequences

Brémont and Marqués proposed a method with this name¹ to interpolate sequences of partitions [29].

Let I_t and I_{t+p} be two partitions corresponding to the segmentation of two images at t and $t + p$ respectively. The goal is to construct the partitions I_{t+1} , I_{t+2} , \dots , I_{t+p-1} by using only I_t and I_{t+p} .

The method of Brémont and Marqués has four steps: region parametrization, region ordering, region interpolation, and partition creation.

¹This name is clearly redundant as the partitions are already segmented. Therefore, a better title would be “Morphological Interpolation of Partition Sequences”. However, in the results section, they say “...a region-based approach for image sequences interpolation”. Therefore better titles for this article would be “Segmentation-based Morphological Interpolation of Image Sequences” or “Region-based Morphological Interpolation of Image Sequences”. These titles refers to both the method and its application area.

Region Parametrization

Let $R_t(i)$ and $R_{t+p}(i)$ be the regions with label i in I_t and I_{t+p} respectively. The evolution of a region from $R_t(i)$ to $R_{t+p}(i)$ can be divided into regular motion and shape deformation. If adjacent regions with the same type have a similar regular motion, they are merged into the same meta-region.

Regular Motion Estimation

This method considers two types of regular motions: translation and zooming. The translation of the center of mass $G(i)$ gives an approximation of the region movement $\vec{T}(i)$. However, it might not be useful for regions that fuse or split. The zoom factor $Z(i)$, due to the apparent enlarging or decreasing of a region caused by the movement of the camera or the region, can be computed using the surface ratio between the initial and final regions. However, it might give wrong values when there exist occlusions.

It was proposed this classification for the possible situations for regions:

1. Foreground regions. Use both translation and zooming.
2. Background regions. Use only translation.
3. Merged or split regions. Stay motionless.

To classify a region, it estimates a motion error by performing each alternative on $R_t(i)$, obtaining $R'_{t+p}(i)$; compares each of them with $R_{t+p}(i)$ ² by computing a function cost on the contours of the regions, and chooses the alternative with the smallest error. The chosen alternative is the motion *type* of the region.

²In the article says $R_t(i)$, but it should be compared with $R_{t+p}(i)$. Otherwise, the best alternative would always be stay motionless as in this case $R'_{t+p}(i)$ is equal to $R_t(i)$. So their contours are the same. Therefore, their distance is zero. In other words, instead of computing the cost between $R_t(i)$ and $R'_{t+p}(i)$, it should be computed the cost between $R_{t+p}(i)$ and $R'_{t+p}(i)$.

Merging on Meta-regions

Some regions belong to the same object. It is important to detect them because this information is useful in the region ordering step.

It is considered that adjacent regions with the same motion type and a similar motion ($\|\vec{T}(i) - \vec{T}(j)\| < \lambda$) belong to the same macro-region.

Shape Deformation

As the regular motion model cannot consider any motion different from translation and zooming, it is necessary another model. Using the regular model to compute $R'_{t+k}(i)$ from $R_t(i)$ and $R''_{t+k}(i)$ from $R_{t+p}(i)$ can give two different regions. The idea is to compute the geodesic distance from $R_t(i)$ to $R_{t+p}(i)$ –after translation and zooming, if required–, and threshold the difference. Two methods were created, one separately computes the deformation for each region and the other simultaneously compute the deformation for all the regions within each macro-region.

Region Ordering

The motion error estimation gives information about the physical depth of the regions. While the estimated error is higher, the region is deeper. For example, in Fig. 2.5 [29], a ball has no error since it has not been hidden by other objects. Therefore, it is the shallowest. The same happens with some squares. However, the ball overlaps some others, so their estimated errors are higher. Consequently, they are deeper.

Region Interpolation

Regions are interpolated by using the parameters computed previously.

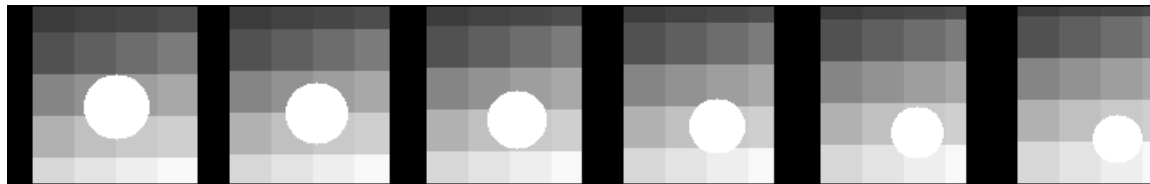


Figure 2.5: A moving ball

Partition Creation

The interpolated regions are placed down in the interpolated partition by following a dead leave model. First, it lays down the deepest regions (the background), then the upper ones, up to the shallowest. For example, the dead leave step was applied to the interpolation of the regions of the initial and final images from Fig. 2.5. Its result can be seen in Fig. 2.6 [29]. Note that some pixels were not covered.



Figure 2.6: Interpolation of Fig. 2.5 after the dead leave step

If the previous step leaves some pixels uncovered, the propagation step fills them with their surrounding labels. This step considers the estimated propagation error: it does not propagate regions that were interpolated correctly. The propagation error is computed as the Hausdorff distance between $R'_{t+k}(i)$ and $R''_{t+k}(i)$.

For example, the interpolation of the initial and final images from Fig. 2.5 can be seen in Fig. 2.7 [29].

2.2.4 A Region-based Interpolation Method for Mosaic Images

Vidal et al. [210] extended their previous work for binary images to mosaic images. A three-step algorithm was proposed whose steps differ from those used to interpolate

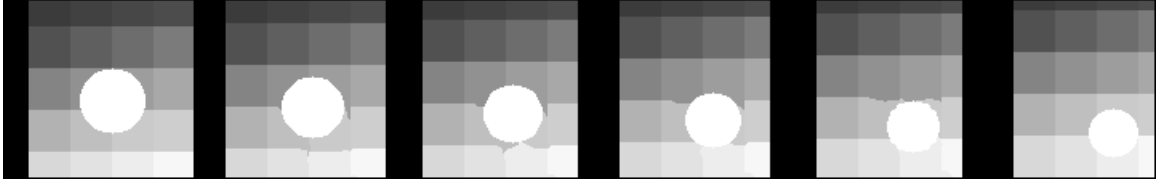


Figure 2.7: Interpolation of a sequence

binary images:

1. Separation of regions in each slice.
2. Matching and interpolation between regions.
3. Final adjustment.

In the first step, each region j belonging to a slice S_i is stored as a binary image (there is the same number of binary images as the number of regions in the slice S_i) in the vector element RS_i^j ; in addition, the grey-level value of each region is stored in a tree structure to preserve the inclusion relationship among regions [61]. The first level of this tree stores all the regions directly and indirectly adjacent to the image border; the next level stores all the regions inside, and directly or indirectly adjacent to the regions in the previous level, and so on. An example is shown in Fig. 2.8: an input mosaic contained in a slice (a) is separated (b), the relationship between its regions, and the grey level of each of them is stored in (c), and their regions are stored as binary images in (d), (e), (f), (g) and (h).

In the second step, the hierarchical level, the grey-level, and the proximity test (see proximity zone in Sec. 2.1.2) are used to decide which regions from one slice match regions from the other. First, regions with the same hierarchical level, grey level and that pass the proximity test are matched. Second, regions belonging to consecutive levels with the same grey level and that pass the proximity test are matched. After this computation, some regions might match various regions. In this case, only the matched region with the minimal Euclidean distance between their MSP points is kept. Finally, the (remaining) matched regions are interpolated by using median sets. It is also possible that some regions do not match any other (called *isolated*

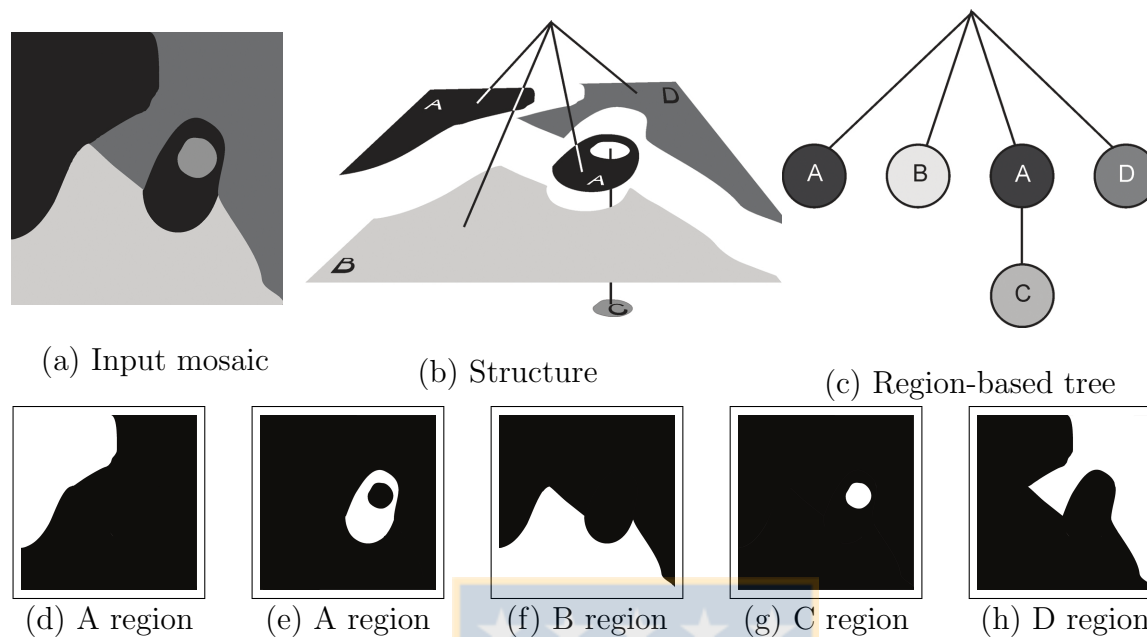


Figure 2.8: A mosaic divided into regions

regions). In this case, for each isolated region, an artificial region is created in the other slice. Usually, this artificial region is a point, but a new approach is used if this region overlaps the border: The artificial region takes different shapes (see Sec. 2.2.4). Finally, isolated regions are interpolated with their artificial regions by using median sets.

The set of interpolated regions has two problems: (1) Their union does not necessarily cover all the interpolated slice; and (2) some regions overlap. In the third step, each point of the interpolated slice that belongs to no region or to various regions is assigned to a region. To do this, it is computed a slice with all the interpolated regions minus their overlapped parts, and then this image is flooded by using the watershed transform.

Finally, the interpolated regions are labeled with the grey values of their original regions.

Classification of Border Regions and their Artificial Connected Components

The border regions were classified in several types (1, 2, 4, 6, 7, and 8). When these border regions are not paired, artificial components are created in the other image. The shapes of these regions and their artificial components are shown below.

If a (non-paired) component overlaps a border, it is interpolated with a segment in that border (see Fig. 2.9).



Figure 2.9: Type 1 region and its artificial connected component

If a component overlaps two borders without interruption, it is interpolated with a point in the corner (see Fig. 2.10).



Figure 2.10: Type 2 region and its artificial connected component

If a component extends over one, two or three borders, it is interpolated with artificial lines in these borders (see Figs. 2.11, 2.12 and 2.13).



Figure 2.11: Type 4 region and its artificial connected component

If a component overlaps various borders with a gap in the middle, it is interpolated with the thinning of the component (see Fig. 2.14).



Figure 2.12: Type 6 region and its artificial connected component



Figure 2.13: Type 7 region and its artificial connected component

2.3 Morphological Interpolation for Grey-scale Images

Brémond and Marqués [29] propose segmenting the images and interpolating their partitions. In this approach, each original image is segmented into a set of regions and, at the same time, the corresponding regions are labeled. Each region is characterized by its contours and texture (the texture can be regular, such as polka dots; stochastic, such as pebbles on a beach; or anywhere in between, such as tiger stripes [118]). For each pair of regions, their contours and textures are interpolated separately. On one hand, they proposed a general scheme to interpolate the contours by using partitions (see Sec. 2.2.3) that is independent of the segmentation method used. On the other hand, they did not explain how to interpolate the textures.

Another image interpolation method [20] uses the same approach. First, the watershed transform is applied to the images, then the interpolation of partitions through median sets (see Sec. 2.2.1) is applied to the resulting mosaics. Unfortunately, the

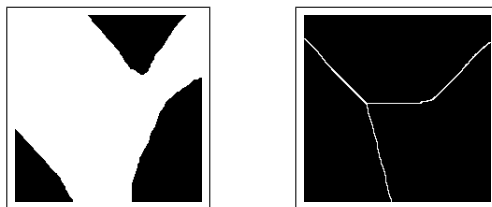


Figure 2.14: Type 8 region and its artificial connected component

description of this method has left out some points about its functioning. For example, are the images smoothed before the watershed transform, is the watershed transform applied to the original images or to the image gradients, and how is the image obtained after the mosaic interpolation.

There is also a method that interpolates images without segmenting them. It is explained below.

2.3.1 Median of Images

Iwanowski and Serra [85] formulated the median of images, an extension of median of sets (see Sec. B.11.1), as:

$$m(f, g) = \sup\{\forall \lambda : \inf[\delta^\lambda(\inf(f, g)), \epsilon^\lambda(\sup(f, g))]\}, \quad (2.1)$$

where f and g are images, $\lambda = 1, 2, \dots, K$ integer values, \inf and \sup are infimum and supremum, respectively, and δ^λ and ϵ^λ are dilation and erosion of size λ , respectively, performed with a non-flat structuring element (see Sec. B.2).

This equation can be applied to grey-level and color images. The only difference are the \inf and \sup operators. For grey-level images, numbers are compared; the greatest number is the \sup and the smaller is the \inf . The comparison of colors is explained in Sec. 2.4.

An algorithm to compute median of images, based on Eq. 2.1, taken from [85], is described below.

Initially, three auxiliary images are defined:

$$\begin{aligned} z_0 &= m_0 = \inf(f, g) \\ w_0 &= \sup(f, g) \end{aligned}$$

Then, new values are computed iteratively until idempotency, i.e. $m_{i+1} = m_i$:

$$\begin{aligned}
z_i &= \delta(z_{i-1}) \\
w_i &= \epsilon(w_{i-1}) \\
m_i &= \sup[\inf(z_i, w_i), m_{i-1}]
\end{aligned}$$

Consequently, the median of images f and g is

$$m(f, g) = m_\infty = m_i.$$

2.4 Color Median of Images

Iwanowski and Serra [85] were pioneers in morphological, color interpolation. First, they propose a method that uses the median of images (see Sec. 2.3.1). This is an automatic method useful when at least one image is textured and there is no need to transform some object into another. The median image of color images can be obtained, and the differences with the median image of grey-level images are the infimum and supremum operators used. In addition, they suggest that constructing these operators with the usual lexicographical ordering (conditional ordering) does not consider the importance of the color components. Therefore, they propose using that ordering in a *comparative vector space*. They suggest that this space should be used only to compare vectors and that it can be created by using linear combinations of the original space, such as the [luma] value. An example of the median image of two color images is shown in Fig. 2.15. Second, they propose applying warping to avoid that some important elements disappear before applying the interpolation. This method is useful in general but requires human intervention to define some control points.



(a) Image

(b) Median image

(c) Image

Figure 2.15: Color median image (b) generated from the images (a) and (c)



New Morphological Interpolation Methods for Color Images

The *general goal* of this work is to *construct*¹ a morphological interpolation method with better picture quality than other methods when the input images contain complex connected components, i.e. components with internal ones².

In this work, it would have been very easy to interpolate using any color morphological operator. However, none of them has general acceptance [10]. In addition, none of them was considered adequate. For example, component-wise operators applied to RGB images might change a color (hue) for another that does not exist in the image [40]. Consequently, the component-wise approach on the RGB color model was discarded because it does not comply with color constancy (see Sec. A.1.1).

Therefore, an approach that does not use color morphology was explored, i.e. an approach that avoids using color morphological operators. This has been done before. For example, if the colors of an image are ordered (they represent elevation on a map, some biological characteristic on a medical image, or something similar), the image

¹In the proposal of this thesis says “to design and program”.

²Vidal [207] considers that complex shapes includes those with inclusion relationship (“...formas complejas (por ejemplo, en las que exista inclusión de formas)”).

can indeed be treated with the grey-scale methods. Also, Comer and Delp [40] suggest processing two-color images with binary morphological operators. One color is taken as the foreground, and the other, as the background. The binary morphological operator is applied only to the foreground.

In this work, it is proposed to apply binary morphological operators to the shapes in color images. For example, if an image shows —seen from above— a red ball and a yellow box, on a brown table, to apply binary morphological operators to a disk, a square, and a rectangle; not to apply color morphological operators to red, yellow, and brown areas.

As it is mandatory to apply segmentation to obtain the shapes, the aforementioned idea leads to mosaics (each shape should be contained in a region of a mosaic). However, there are regions everywhere in a mosaic image. Therefore, if a morphological operation is applied to a region, it might have the opposite effect on others. For example, the binary dilation of a region implies the erosion of its adjacent regions.

To avoid these undesirable consequences, it is possible to apply constraints to these operators (dilation and erosion). These constraints can be provided by a sequence of two mosaics. On one hand, if a region in the first mosaic is larger in the second one, apply dilation to the initial region but without surpassing the final region. On the other hand, if a region in the first mosaic is smaller in the second one, apply erosion to the initial mosaic but without eroding the final region. If these constraints are replaced by a line midway both regions, and the operators are applied until stability, it is the same as mosaic interpolation.

The question is how to apply mosaic interpolation to full-color natural and artificial images. An answer could be extending a method for grey-level image interpolation to color images. In particular, the new methods proposed resemble an adaptation of the Brémond and Marqués proposal (see Sec. 2.3). Their method includes these steps approximately (some steps were split; others were merged): segmenting the images,

matching the regions, interpolating the regions, and finally, combining the interpolated regions to form the interpolated image.

The interpolation methods developed in this work have these stages (see Fig. 3.1):

C1 Segmenting the input images to obtain color mosaics.

C2 Matching the regions of the color mosaics.

C3 Converting the color mosaics with matched regions into grey-level mosaics.

C4 Interpolating the grey-level mosaics.

C5 Coloring the interpolated mosaic.

3.1 Segmenting the Input Images to Obtain Color Mosaics

This process, *C1 Segmenting images* in the figure, receives as input the data flow *DF0 Couple_of_color_images* = $\{O_1, O_2\}$, the original images (see Fig. 3.2). This process gives as output the data flow *DF2 Couple_of_color_mosaics* = $\{M_1, M_2\}$, the color mosaics that represent the partitions of the original images.

The objective of this stage is to partition the original color images O_1 and O_2 into $\bigcup_{j=1}^n RO_1^j$ and $\bigcup_{j=1}^m RO_2^j$, where RO_i^j represents the color region j of a partition of the image O_i . Although these partitions would be useful in the last step, *C5 Coloring interpolated mosaic*, only the color mosaics, M_1 and M_2 , that represents the areas covered by the regions of these partitions are passed to the next stage. For example, the images shown in Fig. 3.3 were segmented. Their color mosaics are shown in Fig. 3.4. Each color in these mosaics represents a region in the original color images. Therefore, the orange regions represent part of the background and the big green regions represent the dog.

It is expected that these color mosaics can be easily and correctly processed in the following stages. So, it is ideal that the number of regions be as small as possible

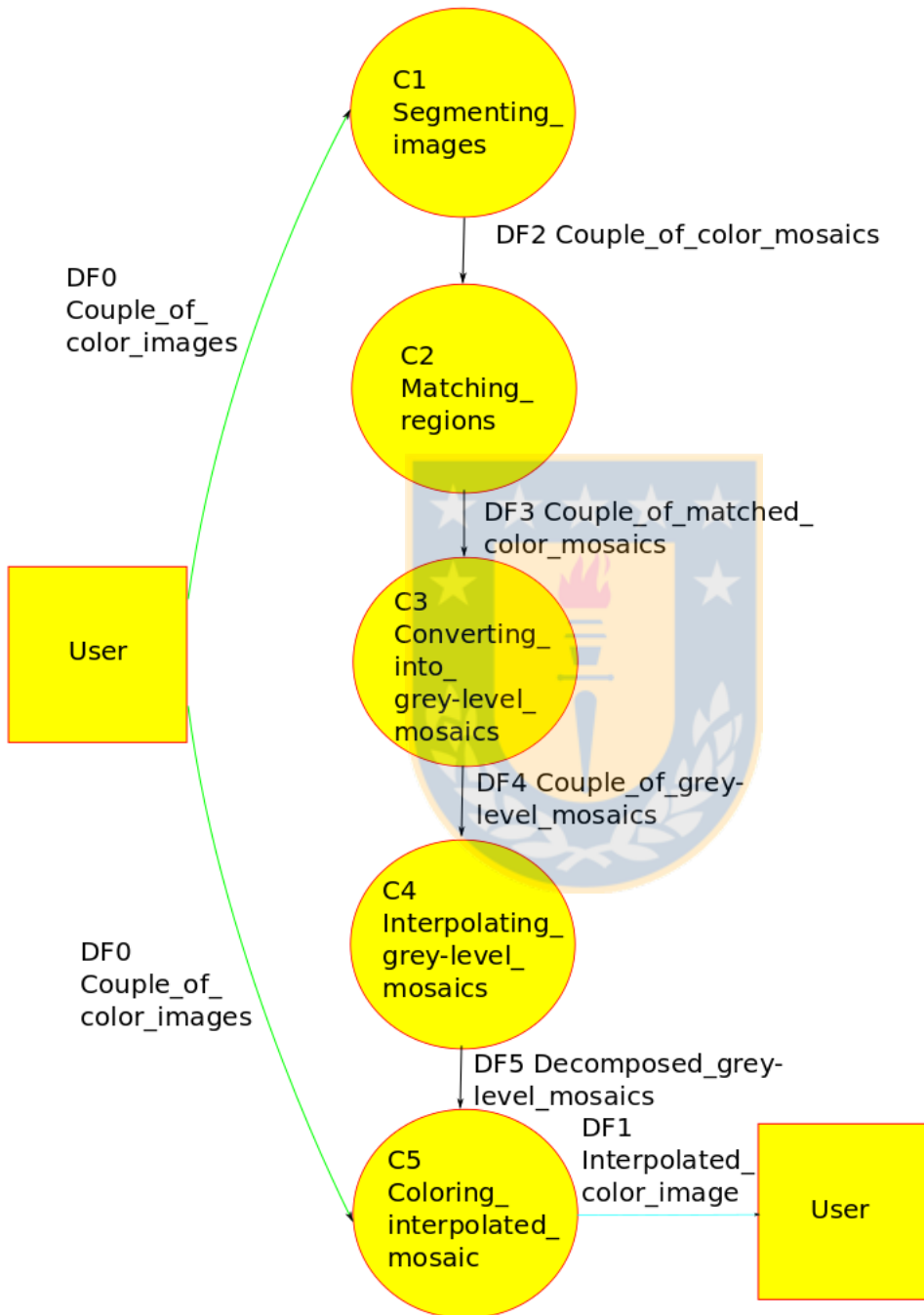


Figure 3.1: New interpolation method

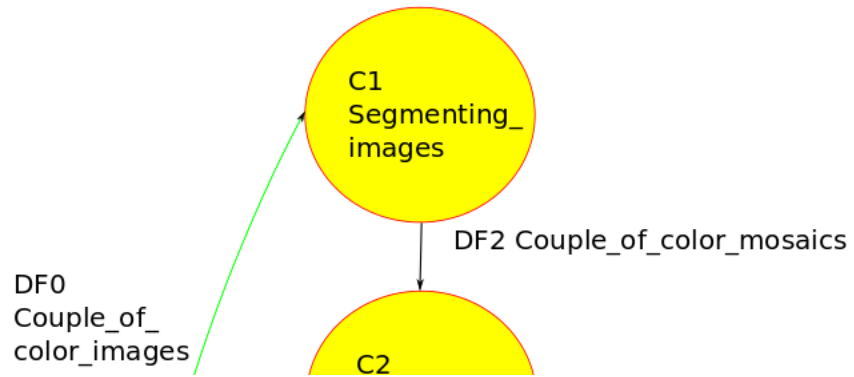


Figure 3.2: C1 Segmenting the input images



Figure 3.3: Initial (leftwards) and final (rightwards) images of dogdance.



Figure 3.4: Initial (leftwards) and final (rightwards) segmented images of dogdance.

but without including objects with different movements (translation, rotation, etc.) or magnitude of movement in the same region. Hence, each region should include an object or a group of objects moving conjointly and independently from others. For example, a car or the parts of a car (tires, doors, windows, etc.) should be considered as a region. If this movement does not only include translation or the occlusions make impossible to compute the center correctly, segment this group into occluded and non-occluded regions. For example, an older child overlaps a younger one in the leftward image in Fig. 3.3. As it is impossible to compute the center of the younger child, the part of her that was occluded in the initial image was segmented in the initial segmented image (see Fig. 3.4).

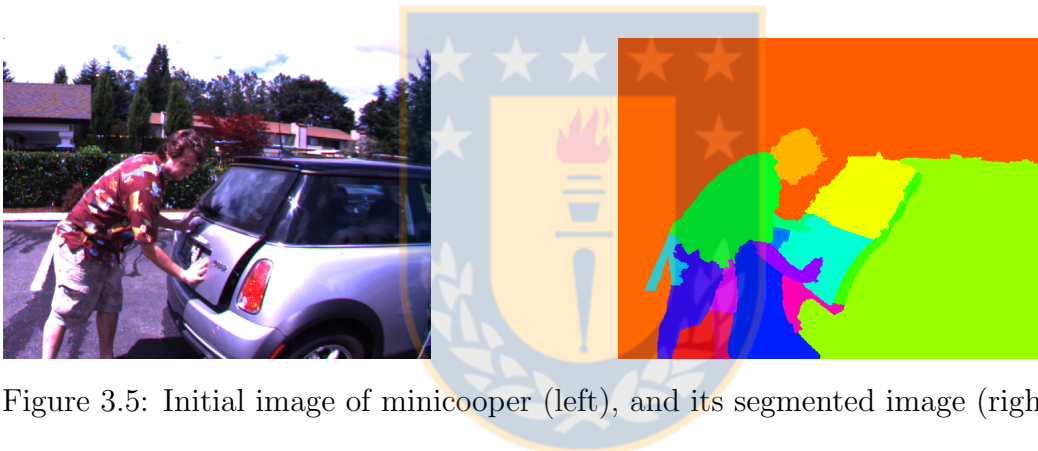


Figure 3.5: Initial image of minicooper (left), and its segmented image (right)

Ideally, this process should segment each input image separately and automatically. In practice, a user segments each original image interactively by using *SegmentIt* (see Sec. C.2.1), and corrects the mosaic(s) with some flaws by comparing both mosaics and both original images. The user can repeat this cycle several times. In spite of this, sometimes it is impossible to separate some objects. For example, segmenting the image (a) gives (b) (see Fig. 3.5). In this case, the inclined white line in the ground and part of the shorts of the man are in the same region.

At the same time, sometimes it is tricky to segment an image. For example, in dumptruck the vehicles move while the background is still. The vehicles have shades, and there are areas between the vehicles and their shades. These shades and these areas are in the background. However, the best interpolation results were obtained

segmenting the Mercedes and its shape in a region, the enclosed area in another, and the rest of the image in another (see the results and discussion chapter for details).

3.2 Matching the Regions of the Color Mosaics

This stage, *C2 Matching_regions* in Fig. 3.6, has as input the data flow *DF2 Couple_of_color_mosaics* with $M_1 = \bigcup_{i=1}^n RM_1^i$ and $M_2 = \bigcup_{j=1}^m RM_2^j$, where RM_1^i and RM_2^j are the regions belonging to M_1 and M_2 , respectively. This process gives as result the color mosaics Q_1 and Q_2 that conform the output data flow *DF3 Couple_of_matched_color_mosaics*.

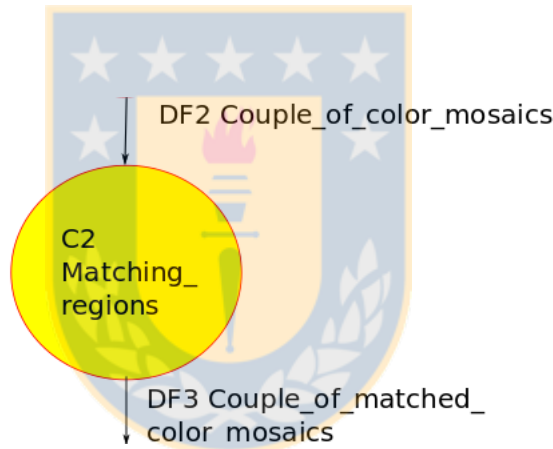


Figure 3.6: C2 Matching the regions of color mosaics

Since the colors of the regions RM_1^i and RM_2^j corresponding to the same (part of a) group of objects can be different, it is necessary to match them. As this problem is beyond the scope of this research, the user puts identical color to all the regions belonging to the same (part of a) group of objects in both mosaics (this process is called matching). As a result, a region in M_1 (M_2) can match one or more regions in M_2 (M_1). It is also possible that a region in M_1 (M_2) does not match any region in M_2 (M_1).

For example, the mosaics in Fig. 3.4 were matched as it can be seen in Fig. 3.7. All the regions of each mosaic have a different color. This helps to avoid interpolating

non-paired regions.



Figure 3.7: Initial and final matched segmented images from dogdance: Leftwards and rightwards, respectively

3.3 Converting the Color Mosaics with Matched Regions into Grey-level Mosaics

This process, *C3 Converting_into_grey-level_mosaics* in the figure, receives as input the data flow *DF3 Couple_of_matched_color_mosaics* = $\{Q_1, Q_2\}$ (see Fig. 3.8). As the interpolation algorithm uses the grey-level to match regions, this stage converts the color mosaics Q_1 and Q_2 into the grey-level ones S_1 and S_2 , respectively. The latter conform the output data flow *DF4 Couple_of_grey-level_mosaics*.

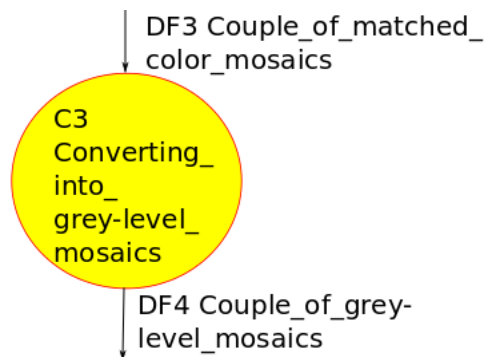


Figure 3.8: C3 Converting the color mosaics into grey-level mosaics

The color pixels were converted by computing their luma (Y') according to the

Rec. 601 [83]

$${}^{601}Y' = 0.299R' + 0.587G' + 0.114B',$$

where R' , G' and B' are the components of the RGB color model.

It is unimportant the formula used to compute luma as the values themselves are not processed. For example, images that use the Rec. 709 [84] can be processed; their luma is

$${}^{709}Y' = 0.2126R' + 0.7152G' + 0.0722B'.$$

As there are more colors than grey-levels, any chosen method to compute luma might convert pixels with different colors into pixels with the same grey-level (*collisions*). This diminution in the number of (grey-level) regions could produce errors in the next stage (*C4 Interpolating_grey-level_mosaics*).

An error is going to occur when two adjacent color regions are merged into one grey-level region. In addition, an error is going to occur when two non-adjacent color regions finish with the same grey-level and one of these regions finishes near the other region in the other slice. In this case, two distinct objects are interpolated. For example, imagine a person that walks so that his right foot in one image overlaps his left foot in the other image. If these feet have the same grey-level, the next step will interpolate them.

These errors can be detected through visual comparison of the grey-level mosaics S_1 and S_2 with the color mosaics Q_1 and Q_2 , respectively. If there are some errors, it is necessary to change the colors of some of the color regions that collided. To do so, repeat the stage that matches regions (see Sec. 3.2).

For example, the images in Fig. 3.7 were converted into Fig. 3.9. The feet of the little child have the same grey-level in spite of their different colors in the color mosaics. Fortunately, this collision does not cause errors. Although these feet seem identical to the left foot of the older girl, they are different (the feet value is 76 and

the foot value, 75).



Figure 3.9: Initial and final grey-level mosaics from dogdance: Leftwards and rightwards, respectively

3.4 Interpolating the Grey-level Mosaics

This process, *C4 Interpolating_grey – level_mosaics* in the figure, takes the input data flow *DF4 Couple_of_grey – level_mosaics* = $\{S_1, S_2\}$, and obtains the output data flow *DF5 Decomposed_grey – level_mosaics* (see Fig. 3.10).

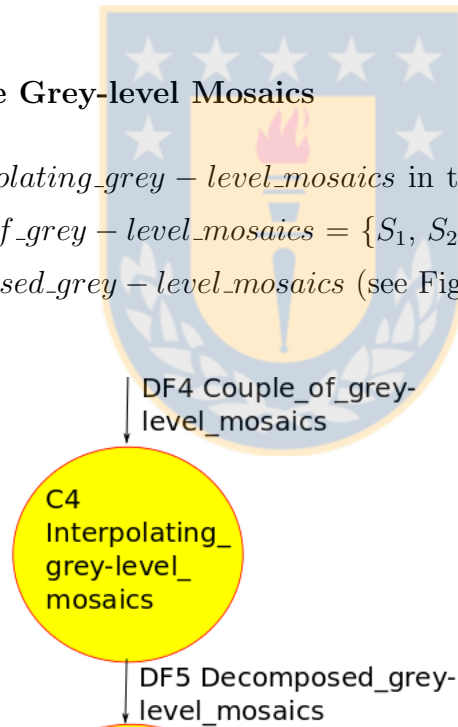


Figure 3.10: C4 Interpolating the grey-level mosaics

It was implemented by using a grey-level mosaic interpolation algorithm from Vidal et al. [210] (this algorithm is explained in detail in Sec. 2.2.4). This algorithm uses vectors of binary images RS_i , where RS_i^j represent each grey-level region j of the slice S_i with “1” and the remainder of this image is filled with “0”. It interpolates the matched regions from S_1 and S_2 and its results are stored in RS_3^k . As it is possible

that some of these regions overlap or that they leave some empty areas, they are adjusted to avoid both conditions. The vectors RS_1 , RS_2 , and the adjusted vector RS_3 conform the output data flow *DF5 Decomposed_grey – level_mosaics*.

Although the interpolated image S_3 ($S_3 = \bigcup_{k=1}^n RS_3^k$) is not necessary for the next step, it is very useful to discover where some errors originate in the interpolated color image. For example, the interpolation of the images in Fig. 3.9 gives Fig. 3.11. Note some conspicuous errors: the feet of the dog disappeared, and the head and the body of the little girl seem to overlap the body of the older girl. It is not strange that the feet of the dog disappeared in the interpolated color image.



Figure 3.11: Interpolated grey-level mosaic from dogdance

As this work bases so much in this algorithm, described in Vidal et al. works that use mathematical morphology and consider structural aspects of binary [209] and mosaic [210] images, this work might be considered its extension to color images.

3.5 Coloring the Interpolated Mosaic

This stage, *C5 Coloring_interpolated_mosaic* in the figure, constructs an interpolated color image F by using the data flows *DF0 Couple_of_color_images* = $\{O_1, O_2\}$ and *DF5 Decomposed_grey-level_mosaics* = $\{RS_1, RS_2, RS_3\}$ (see Fig. 3.12).

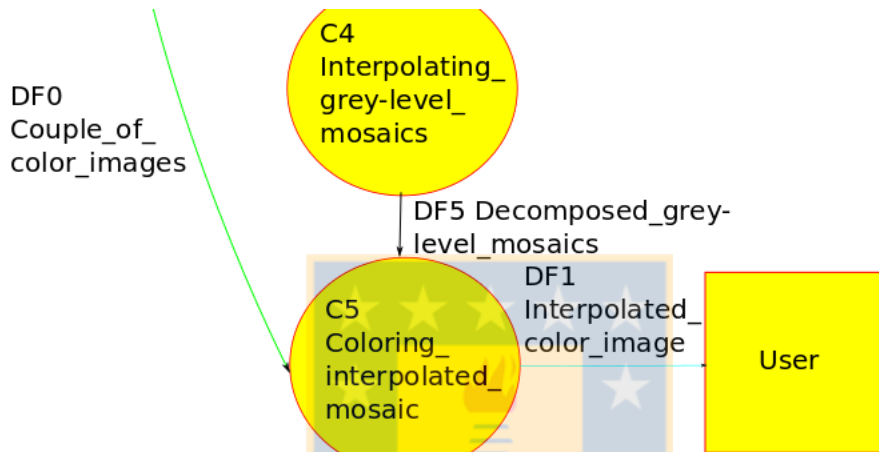


Figure 3.12: C5 Coloring the interpolated mosaic

Initially, it was proposed an algorithm that uses the regions in RS_1^r and RS_2^r to interpolate the associated color regions in RO_1 and RO_2 . A short description is shown below (see Sec. 3.5.1 for details).

The regions in RS_1^r , RS_2^r , RO_1 and RO_2 are copied appropriately into RSM_1 , RSM_2 , ROM_1 and ROM_2 , respectively.

The regions in ROM_1 and ROM_2 are interpolated linearly where the three regions in RSM_1 , RSM_2 , and RS_3^r overlap. In this case, it is possible to apply linear interpolation as there are points in both ROM_1 and ROM_2 to interpolate each point. At the same time, the nearest-neighbor should not be used as the linear interpolation is a first-order approximation while the nearest-neighbor is a zero-order one[97].

The region in ROM_1 (ROM_2) is interpolated with the nearest-neighbor method

where the regions in RS_3^r and RSM_1 (RS_3^r and RSM_2) overlap but without overlapping RSM_2 (RSM_1). In this case, using nearest-neighbor is the only option as there is only one point in either ROM_1 or ROM_2 to interpolate each point.

Later, it was proposed an algorithm similar to the previous one but has a difference: it computes the deformation that the binary region in RSM_1 (RSM_2) must have to match its interpolated binary region in RS_3^r , and applies the same deformation to the color region in ROM_1 (ROM_2). If both regions exist, it applies either the linear interpolation or the median image generation; otherwise the nearest-neighbor one. It was named *deforming* (see Sec. 3.5.2).

Although these algorithms use slices as input, it is possible and advisable to modify them to use regions instead. Thus, it is possible to execute one or another algorithm on different matched regions of the same slices. If the objects change between the matched regions, the *deforming* algorithm should be executed, else the *overlapping* one. In fact, it was constructed an interpolation program following this approach.

3.5.1 Overlapping

The *overlapping* algorithm, as it is explained here, can replace the *C5 Coloring the interpolated mosaic* process (see Fig. 3.12). In this algorithm (see Alg. 1), for each interpolated binary image RS_3^r :

1. If RS_1^r (RS_2^r) exists, cut the original color region from O_1 (O_2) by using RS_1^r (RS_2^r) as a mold and store it in RO_1 (RO_2). If RS_1^r (RS_2^r) exists, copy the original regions in RS_1^r and RO_1 (RS_2^r and RO_2) into RSM_1 and ROM_1 (RSM_2 and ROM_2) so that the MSP of the region in RSM_1 (RSM_2) coincide with the MSP of the region in RS_3^r . This step includes these sub-steps:
 - (a) The original binary image RS_1^r (RS_2^r) is used to obtain the corresponding color image RO_1 (RO_2). The region in the binary image RS_1^r (RS_2^r) is represented with “1” and the remainder of this slice, with “0”. Then, it was

made a point-to-point multiplication of (RS_1^r, RS_1^r, RS_1^r) $[(RS_2^r, RS_2^r, RS_2^r)]$ with the color slice O_1 (O_2). Hence, the color image obtained RO_1 (RO_2) includes the color region and the remainder of this image finishes black $(0, 0, 0)$.

- (b) The regions in RS_1^r and RO_1 are copied into RSM_1 and ROM_1 , respectively, so that the MSP of the region in RSM_1 coincide with the MSP of the region in RS_3^r . It was made moving a squared section that contains the regions in RS_1^r and RO_1 in the same magnitude as the distances between the MSPs of RS_1^r and RS_3^r . The regions in RS_2^r and RO_2 are copied into RSM_2 and ROM_2 in the same way as RS_1^r and RO_1 were copied.

2. The interpolation is performed using two different methods. The linear interpolation method interpolates the part of the color regions in ROM_1 and ROM_2 where the interpolated region in RS_3^r covers the regions in RSM_1 and RSM_2 ; and the nearest-neighbor interpolation method interpolates the part of the color region in ROM_1 (ROM_2) where the interpolated binary region in RS_3^r covers only the region in RSM_1 (RSM_2). For example, there is a red square in ROM_1 and a green circle in ROM_2 . Only to facilitate this explanation, these objects were put in an image (the square was put first and then, the circle) which is shown at the left in Fig. 3.13. The interpolated region F is shown in the image at the right in Fig. 3.13. F is completely included within the union of the square and the circle. The points in which the circle and the square overlap were interpolated linearly so they are brown. The points interpolated with nearest neighbor were taken from the part of the circle outside of the square (in green) and from the part of the square outside of the circle (in red).

An example of interpolation with overlapping considers O_1 , O_2 , RS_1 , RS_2 , and RS_3 shown in Figs. 3.3, 3.7, and 3.11, respectively³. The interpolated image F is shown in Fig. 3.14. The background is blurred because part of it is occluded at the

³This algorithm receives the binary regions in RS_1 , RS_2 , and RS_3 that conform the slices S_1 , S_2 , and S_3 , respectively.

Algorithm 1 Overlapping algorithm

```

function OVERLAPPING( $O_1, O_2$ :colorSlice;  $RS_1, RS_2, RS_3$ :array of binaryImage)
   $F$ :colorSlice
  for each image  $RS_3^r$  do
    if  $RS_1^r$  exists then
       $RO_1 \leftarrow (RS_1^r, RS_1^r, RS_1^r) * O_1$ ; // (1a) Cut the original color region.
      // (1b) Move the original regions
       $displacement\_MSP1 \leftarrow$  MSP of the region in  $RS_3^r$  – MSP of the region
      in  $RS_1^r$ ;
       $square\_region \leftarrow$  square region that includes the region in  $RS_1^r$ ;
       $RSM_1 \leftarrow square\_region$  displaced by  $displacement\_MSP1$ ;
       $square\_region \leftarrow$  square region that includes the region in  $RO_1$ ;
       $ROM_1 \leftarrow square\_region$  displaced by  $displacement\_MSP1$ ;
    end if
    if  $RS_2^r$  exists then
       $RO_2 \leftarrow (RS_2^r, RS_2^r, RS_2^r) * O_2$ ; // (1a) Cut the original color region
      // (1b) Move the original regions
       $displacement\_MSP2 \leftarrow$  MSP of the region in  $RS_3^r$  – MSP of the region
      in  $RS_2^r$ ;
       $square\_region \leftarrow$  square region that includes the region in  $RS_2^r$ ;
       $RSM_2 \leftarrow square\_region$  displaced by  $displacement\_MSP2$ ;
       $square\_region \leftarrow$  square region that includes the region in  $RO_2$ ;
       $ROM_2 \leftarrow square\_region$  displaced by  $displacement\_MSP2$ ;
    end if
    // (2) Interpolating the corresponding pixel(s) in  $ROM_1$  and/or  $ROM_2$ 
    for each position  $p$  in the region in  $RS_3^r$  do
      if position  $p$  is in the regions in  $RSM_1$  and  $RSM_2$  then
         $pixel1 \leftarrow$  value of the position  $p$  of  $ROM_1$ ;
         $pixel2 \leftarrow$  value of the position  $p$  of  $ROM_2$ ;
        position  $p$  of  $F \leftarrow$  linear_interpolation ( $pixel1, pixel2$ ) //  $pixel1$  and
        //  $pixel2$  must be translated into a linear color model.
      else if position  $p$  is in the region in  $RSM_1$  then
        position  $p$  of  $F \leftarrow$  value of the position  $p$  in  $ROM_1$ 
      else if position  $p$  is in the region in  $RSM_2$  then
        position  $p$  of  $F \leftarrow$  value of the position  $p$  in  $ROM_2$ 
      else
        error
      end if
    end for
  end for
  return  $F$ 
end function

```

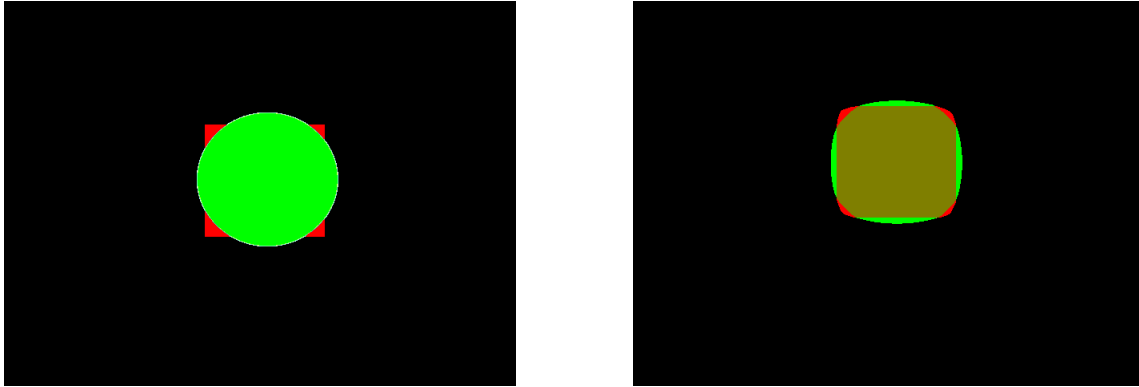


Figure 3.13: Images before and after overlapping: Leftwards and rightwards, respectively

right and left sides of the first and second images, respectively. Although it could be applied the same solution as in the occlusions caused by objects, it is very difficult to “segment”⁴ these areas.



Figure 3.14: Image interpolated with overlapping

⁴In fact, this is not a segmentation problem but an image registration one.

3.5.2 Deforming

The *deforming* algorithm replaces *C5 Coloring_interpolated_mosaic* (see Fig. 3.12). In this algorithm (see Alg. 2), for each interpolated binary image RS_3^r :

1. If RS_1^r (RS_2^r) exists, cut the original color region from O_1 (O_2) by using RS_1^r (RS_2^r) as a mold and store it in RO_1 (RO_2). If RS_1^r (RS_2^r) exists, copy the original regions in RS_1^r and RO_1 (RS_2^r and RO_2) into RSM_1 and ROM_1 (RSM_2 and ROM_2) so that the MSP of the region in RSM_1 (RSM_2) coincide with the MSP of the region in RS_3^r . This step includes these sub-steps:
 - (a) The original binary image RS_1^r (RS_2^r) is used to obtain the corresponding color image RO_1 (RO_2). The region in the binary image RS_1^r (RS_2^r) is represented with “1” and the remainder of this slice, with “0”. Then, it was made a point-to-point multiplication of (RS_1^r, RS_1^r, RS_1^r) [(RS_2^r, RS_2^r, RS_2^r)] with the color slice O_1 (O_2). Hence, the color image obtained RO_1 (RO_2) includes the color region and the remainder of this image finishes black (0, 0, 0).
 - (b) The regions in RS_1^r and RO_1 are copied into RSM_1 and ROM_1 , respectively, so that the MSP of RSM_1 coincide with the MSP of RS_3^r . It was made moving a squared section that contains the regions in RS_1^r and RO_1 in the same magnitude as the distances between the MSPs of RS_1^r and RS_3^r . The regions in RS_2^r and RO_2 are copied into RSM_2 and ROM_2 in the same way as RS_1^r and RO_1 were copied.
2. If RS_1^r (RS_2^r) exists, deform the region in ROM_1 (ROM_2) to match the shape of the region in RS_3^r , and store the result in DS_1 (DS_2). This step requires these sub-steps:
 - (a) It is computed the compression that equals the translated binary region in RSM_1 (RSM_2) to its intersection with the interpolated binary region in RS_3^r —i.e. the region in $RSM_1 \cap$ the region in RS_3^r (the region in $RSM_2 \cap$ the region in RS_3^r)—, and this compression is applied to the corresponding color region in ROM_1 (ROM_2); the result is stored in ES_1 (ES_2).

- (b) It is computed the expansion that equals the intersection of the regions in RS_1 and RS_3^r (RS_2 and RS_3^r) to the region in RS_3^r , and the same expansion is applied to the color region in ES_1 (ES_2); the result is stored in DS_1 (DS_2). The compression reduce the number of pixels; the expansion multiplies the pixels. So, applying compression and then expansion creates an overrepresentation of some pixels. Therefore, it makes sense to change the order of application of these algorithms. In this case, the intersection is replaced by the union.
3. Finally, if RS_1^r and RS_2^r exist, the color regions in DS_1 and DS_2 are congruent, so they can be interpolated with any color interpolation method, such as the linear and median methods explained in Secs. A.10.1 and 2.4, respectively; otherwise the interpolated color region is equal to the unique resulting color region. The result of the previous operation is added to F .

A central point in this method is the algorithm for compression and expansion (*expanding_or_compressing_Image*). This algorithm is used in the steps 2 (a) and 2 (b). It is based on the big region and the small region concepts. Suppose two paired regions (these regions are in different slices). It is supposed that they are put in two new slices. A region is put in a slice and the other region is put over it. In the other slice, the regions are put changing places. The *big region* is the region that covers the other region in a slice; and the *small region* is the region that overlaps the other region without covering it in the other slice. For example, there is a magenta region and a green one in Fig. 3.15. If these matched regions are put one over the other, either the small region overlaps —then, does not cover— the big one or the big region covers —then, does not overlap— the small one (see Fig. 3.16).

Although in the following sections only color regions are mentioned, the borders of these regions are taken from the corresponding binary regions, as it is impossible to know where color regions without labels finish. Accordingly, pixels are taken from and put into color regions.

Algorithm 2 Deforming algorithm

```

function DEFORMING( $O_1, O_2$ :colorSlice;  $RS_1, RS_2, RS_3$ :array of binaryImage)
   $F$ :colorSlice;  $RSM_1, RSM_2$ :binaryImage
   $DS_1, DS_2, ES_1, ES_2, RO_1, RO_2, ROM_1, ROM_2$ :colorImage
  for each image  $RS_3^r$  do
    if  $RS_1^r$  exists then // (1)
       $RO_1 \leftarrow (RS_1^r, RS_1^r, RS_1^r) * O_1$ ; // (1a) Cut the original color region
      // (1b) Move the original regions
       $displacement\_MSP1 \leftarrow$  MSP of the region in  $RS_3^r$  – MSP of the region
      in  $RS_1^r$ ;
       $square\_region \leftarrow$  square region that includes the region in  $RS_1^r$ ;
       $RSM_1 \leftarrow square\_region$  displaced by  $displacement\_MSP1$ ;
       $square\_region \leftarrow$  square region that includes the region in  $RO_1$ ;
       $ROM_1 \leftarrow square\_region$  displaced by  $displacement\_MSP1$ ;
    end if
    if  $RS_2^r$  exists then
       $RO_2 \leftarrow (RS_2^r, RS_2^r, RS_2^r) * O_2$ ; // (1a) Cut the original color region
      // (1b) Move the original regions
       $displacement\_MSP2 \leftarrow$  MSP of the region in  $RS_3^r$  – MSP of the region
      in  $RS_2^r$ ;
       $square\_region \leftarrow$  square region that includes the region in  $RS_2^r$ ;
       $RSM_2 \leftarrow square\_region$  displaced by  $displacement\_MSP2$ ;
       $square\_region \leftarrow$  square region that includes the region in  $RO_2$ ;
       $ROM_2 \leftarrow square\_region$  displaced by  $displacement\_MSP2$ ;
    end if
    if  $RS_1^r$  exists then // (2)
       $ES_1 \leftarrow$  expanding_or_compressingImage ( $ROM_1, RSM_1, RSM_1 \cap RS_3^r$ ,
      “compress”);
       $DS_1 \leftarrow$  expanding_or_compressingImage ( $ES_1, RSM_1 \cap RS_3^r, RS_3^r$ ,
      “expand”);
    end if
    if  $RS_2^r$  exists then
       $ES_2 \leftarrow$  expanding_or_compressingImage ( $ROM_2, RSM_2, RSM_2 \cap RS_3^r$ ,
      “compress”);
       $DS_2 \leftarrow$  expanding_or_compressingImage ( $ES_2, RSM_2 \cap RS_3^r, RS_3^r$ ,
      “expand”);
    end if
    if  $RS_1^r$  and  $RS_2^r$  exist then // (3)
       $F \leftarrow F \cup$  anyColorInterpolationMethod ( $DS_1, DS_2$ )
    else if  $RS_1^r$  exists then
       $F \leftarrow F \cup DS_1$ 
    else if  $RS_2^r$  exists then  $F \leftarrow F \cup DS_2$ 
    end if
  end for
  return  $F$ 
end function

```



Figure 3.15: A big and a small regions: Leftwards and rightwards, respectively



Figure 3.16: The small region overlaps the big one (leftward) while the big region covers the small one (rightward)

3.6 The Algorithm for Compression and Expansion

This algorithm is able to compress and expand regions. In the compression, it distributes (copies) the pixels in the big region within the small one; while, in the expansion, it stretches the pixels in the latter within the former⁵.

A region can be seen as a set of *line segments*. The fundamental idea to compress and expand regions is to distribute pixels along line segments. Thus, in the compression, it distributes the pixels of each line segment belonging to the big region within the part of this segment within the small one; while, in the expansion, it stretches the latter pixels within the former ones.

The challenge is to define adequate line segments. As a big region encloses a small one, it is clear that each segment have to begin in the border of the former. It is also

⁵The expansion, if the small region has only a pixel, creates a big region painted with this pixel. Therefore, it seems fair to give more importance to the paired region, but this possibility was not explored.

clear that each segment should finish within the small region. The question is where. The approach used in this work suppose that the line segments finish in the *nucleus* —the part of the skeleton of the big region inside the small one. However, under some circumstances the nucleus can be replaced by a connected component called *birthplace* (see Sec. 3.6.2).

3.6.1 Compression and Expansion between the Borders and the Nucleus

Every region has a skeleton. In this work, it was calculated in Matlab with `bwmorph` operation “`skel`”. This operation uses lookup tables to remove iteratively pixels from the borders but without allowing that the regions break apart⁶.

As it is deduced from Sec. B.8, there is an associated skeleton point along the (interior) normal from each point in the border of a region. Consequently, there is a line segment from each point in the borders to the skeleton.

Therefore, the line segments beginning in the big region and finishing in the skeleton could be used to compress and expand. In the compression, each (line) segment is compacted in the part of it within the small region. In the expansion, the part of each segment within the small region is extended in the whole segment. However, some parts of the big region skeleton extend outside the small region. Thus, the segments that finish in these parts might not pass through the small region. Consequently, there are no pixels to stretch in the expansion, or place to distribute the pixels in the compression. To solve this problem, it was proposed that these segments finish at the nearest point inside the small region belonging to the skeleton, i.e. the nucleus.

In addition, several adjustments were needed since the borders could be imperfect in discrete images. This has two consequences: the skeletons and the line segments could be incorrect. In the case of the skeletons, spurious branches could start in the

⁶<http://www.mathworks.com/help/images/ref/bwmorph.html> and internal documentation from Matlab2012a.

borders. In the case of the line segments, their gradient could not be normal to the real border so the line segments do not finish in the right point of the skeleton. An option to solve this problem is to use some skeletonization method that gives for each point in the border the corresponding point in the skeleton.

In particular, the method based on skeletons get bad results with circles and annuli because their skeletons are neither points nor circumferences, respectively, and the segments swing significantly around the correct value —some segments have too much gradient; some, too low. Hence better results are obtained on circles by using *Compression and Expansion between the Outer Border and a Set of Points* (see Sec. J.1) and on annulus by using *Compression and Expansion between the Borders and Rings* (see Sec. J.2). For example, there are ring-shaped areas and a circle in the compact disc images (see Fig. 4.13). Their interpolations using skeletons and birthplaces are shown in Fig. 3.17.

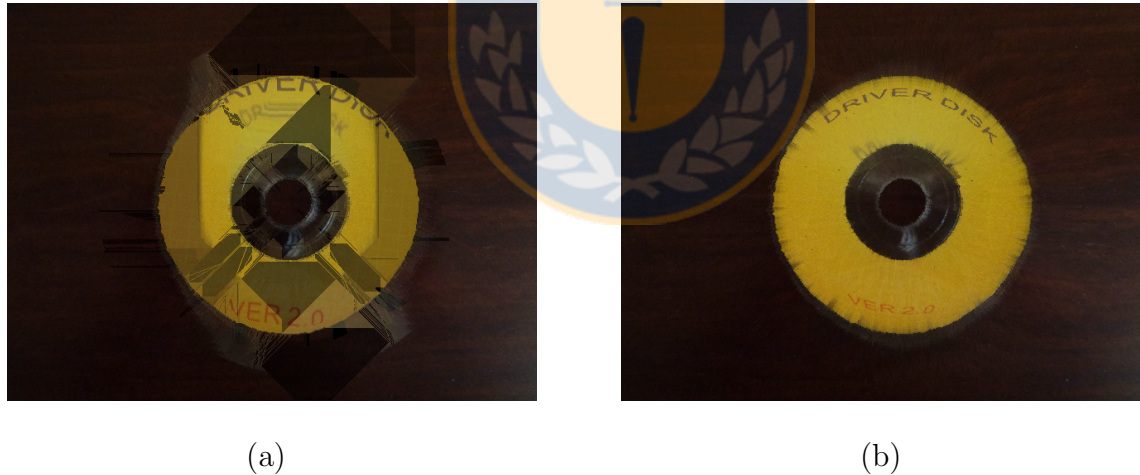


Figure 3.17: Interpolated image of the sequence called compact disc by using (a) skeletons and (b) birthplaces

This approach sometimes fails when the big region overlaps the borders. Here, the color interpolation fails because it is impossible to compute an “adequate” skeleton for the big region because the region represents an object that extends outside the image. On one hand, there is a part of the skeleton outside the region. On the other hand, “artificial” branches are added to the skeleton from the points where the region

intersects the image border. If this region has a hole, it is possible to get better results by using *Compression and Expansion between the Borders and Rings* (see Sec. J.2) since this method eliminates the artificial branches. However, it could also eliminate correct branches. Another possibility—that was not proved—is erasing the artificial branches.

In particular, this approach is unable to manage non-paired border regions without holes adequately since the grey-level interpolation algorithm might process these regions incorrectly. In addition, if it could process them correctly, it would give an interpolated region whose outer border overlaps the big region nucleus. In this case, the compression reduces the exterior region (the part of the big region that is outside the small region) to the outer border, and it copies the rest of the big region in the small one, what distorts the interpolated region. This problem can be avoided processing the border regions with other methods, such as the *Compression and Expansion between the Borders and an Artificial Connected Component* method (see Sec. J.3).

This approach also can fail when a rough or turning border change much the gradient, so that two consecutive (line) segments could left an empty area (see Fig. 3.18). Particularly, where the border is shorter than the nucleus (each point in the border originates only one segment towards the nucleus). For this reason, when two consecutive segments end in the nucleus, new segments are drawn between them (see Fig. 3.18). If one or both segments finish elsewhere, no action is taken.

Following the aforementioned ideas, it was constructed the *expanding_or_compressing-Image* algorithm (see below, Alg. 3). This algorithm can compress or expand an image. It has these steps:

- (1) The compression and expansion are similar, so it is possible to unify them in the same algorithm. This step follows this goal by filling the small and big regions. Notice that in “expand” the final region is the big one, and in “compress” the final region is the small one.



Figure 3.18: A leg before (leftward) and after (rightward) tracing segments

(2) It computes the portion of the skeleton of the big region inside the small region. This is the nucleus.

(3) It computes the external borders from the big region and from its holes, and store them in *bordes*.

(4) First, it calculates the gradient of the big region by using convolution with the 5x5 kernel proposed by Kroon (see Sec. A.6.2) —the convolution fills with zeros the borders of the image that contains this region. Second, where the border of the big region covers the border of the image, it replaces the previous gradient with the normal to the image border.

(5) Color tracing between the borders of the big region and the nucleus.

(5.1) Only the borders of the region are useful. However, *bordes* contains borders from the region and from its holes. Consequently, when a border belongs to a hole, it finds the adjacent border of the region.

(5.2) For each point in the border of a region, it finds the segment to the skeleton or the opposite border, and stores it in *camino*. It also finds the segment from the next point in the border to the skeleton or the opposite border, and stores it in *camino_siguiente*. Although line segments have two ends, it was considered that the end on the first border is the beginning and that the other end is the end. If *camino* ends in the nucleus,

(5.2.1) the pixels along *camino* are copied along the part of it inside the small region in the compression and the pixels along *camino* inside the small region are stretched along *camino* in the expansion. In addition, if *camino_siguiente* ends in the nucleus, it calls the *trace_segments* procedure.

(5.2.2) otherwise, it finds the path within the big region to the nearest point

belonging to the nucleus, called *camino_al_nucleo*, and it does the compression and expansion detailed in the previous step by using this auxiliary path. In this case, it does not call the *trace_segments* procedure since two consecutive paths usually do not leave empty areas as they should finish in the same point of the nucleus.

Algorithm 3 expanding_or_compressing_Image_by_using_skeleton algorithm

```

function EXPANDING_OR_COMPRESSING_IMAGE(Rcolorinicial: colorImage;
Rinicial, Rfinal: binaryImage, operacion: string)
    Rcolorfinal: colorRegion
    Rbig, Rsmall, borde: binaryImage
    bordes: array // See the description of bordes in the description of B in
    // http://www.mathworks.com/help/images/ref/bwboundaries.html
    camino, camino_siguiente: line segment

    // (1) These sentences allow that the expansion and the compression work in
    // the same algorithm. The next instruction belongs to Matlab.
    if strcmp(operacion, "expand") == 1) then
        Rsmall = Rinicial
        Rbig = Rfinal
    else
        Rsmall = Rfinal
        Rbig = Rinicial
    end if

    // (2) Finding the nucleus (area of Rbig skeleton inside of Rsmall).
    esqueleto ← skeleton (Rbig)
    nucleo ← esqueleto ∩ Rsmall

    // (3) Finding the border(s) of the region
    borde = bwperim (Rbig); // Matlab sentence that obtains the borders from
    // the region.
    bordes = bwboundaries (Rbig, 4, 'holes'); // Matlab sentence that obtains
    // the external borders from the region and from its holes.

    // (4) Calculating the vertical and horizontal components of the gradient of
    // Rbig
    [Gv, Gh] = GRADIENTE (Rbig, borde);

```

Algorithm 4 expanding_or_compressing_Image_by_using_skeleton algorithm (continued)

```

// (5) Color tracing between the borders of Rbig or Rsmall and the nucleus.
for each boundary P in bordes do

    // (5.1)
    if P belongs to the foreground then
        circunvalacion ← P
    else // Obtaining the internal boundary adjacent to P.
        [x_pixel_del_borde_de_un_agujero, y_pixel_del_borde_de_un_agujero] ←
        any pixel q adjacent to P and belonging to Rbig
        circunvalacion = bwtraceboundary (borde, [x_pixel_del_borde_de_un_
        agujero, y_pixel_del_borde_de_un_agujero], 'N', 4); // Matlab sentence
        // that obtains the internal border from Rbig that includes the pixel q.
    end if

    // (5.2) Tracing of segments
    for each pixel p in circunvalacion do
        camino = buscarSegmento_al_Esqueleto_o_al_Borde_Opuesto (p, Rbig,
        esqueleto, Gv, Gh);
        camino_siguiete = buscarSegmento_al_Esqueleto_o_al_Borde_Opuesto
        (pixel after p in circunvalacion);
        if camino ends in the nucleo then // (5.2.1) Trace camino
            Rcolorfinal = ponerPixeles_en_la_forma_small_o_big (camino,
            Rcolorinicial, Rcolorfinal, Rsmall, Rbig, operacion);
            if camino_siguiete ends in the nucleo then // Color tracing from
            // the circunvalacion to the nucleo between camino and
            // camino_siguiete.
                trace_segments (camino, camino_siguiete, Rcolorinicial,
                Rcolorfinal, Rbig, Rsmall);
            else // (5.2.2) Find the path within Rbig to the nearest point
            // belonging to the nucleo, and draw this path.
                camino_al_nucleo = Buscar_camino_al_nucleo_de_la_imagen (p,
                nucleo);
                Rcolorfinal = ponerPixeles_en_la_forma_small_o_big (camino_al_
                nucleo, Rcolorinicial, Rcolorfinal, Rsmall, Rbig, operacion);
            end if
        end if
    end for
end for
return Rcolorfinal
end function

```

Drawing Line Segments between Consecutive Ones

This algorithm traces (line) segments from a border to a nucleus between *camino* and *camino_siguiente*. However, tracing segments when it is unnecessary degrades the performance and the image quality. For this reason, this algorithm (see below, Alg. 5) only computes new segments after checking some conditions (the first two steps). In spite of this, at times it redraws some pixels.

The steps of this algorithm are:

1. Check that the ends of the segments are neither adjacent nor finish in the same point. As *camino* and *camino_siguiente* begin in the border and are consecutive, the (1-norm) distance between their beginnings is one. Consequently, if their ends finish in the nucleus at a distance lesser or equal to one, they cannot leave empty areas between them. Therefore, only if the ends distance is greater than one, they could left empty areas between them. For example, the colored segments in the image (a) in Fig. 3.19 are adjacent. Therefore it is impossible to trace segments between them. The distance between the ends of the blue segment that is adjacent to the green one is 2; this shows that a distance between the ends greater than one is not sufficient to assure that two consecutive segments are not adjacent.
2. Check that these segments do not intersect⁷. For example, in the image (b) in Fig. 3.19 the green segment was traced first and the red one later.
3. Sometimes consecutive segments leave empty areas between them. For example, the green and the red, and the red and the blue ones in the image (c) in Fig. 3.19. As each segment is a part of a line, it tries to find the intersection point, *Interseccion*, of the lines corresponding to these consecutive segments.
4. Find the shortest path with 1-norm distance between *camino* and *camino_siguiente* along the nucleus; store it in *recorrido_nucleo* and its length in *indice*.

⁷Migeon uses the external normal direction on the border of the internal figure to get rid of these intersections [132].

As the nucleus is a subset of a skeleton, it is possible that the nucleus is not connected. In this case, there is no path along the nucleus between the segments.

5. If *recorrido_nucleo* exists, for each pixel belonging to *recorrido_nucleo* except the first:
 - (a) Find the segment *camino_entre_medio* whose ends are *Interseccion* and the pixel.
 - (b) In the compression, the pixels between the outer border of the big region and the nucleus along *camino_entre_medio* are put between the outer border of the small region and the nucleus along *camino_entre_medio*. In the expansion, the pixels along *camino_entre_medio* between the outer border of the small region and the nucleus are put into *camino_entre_medio* between the outer border of the big region and the nucleus.

3.6.2 Birthplaces

Sometimes a skeleton does not represent the “ideal” shape correctly. For example, this happens to circles and annuli. In the case of a circle, the skeleton should be a point at the center. In the case of an annulus, the skeleton should be a ring midway between the borders of the ring. However, spurious branches start from the borders. Sometimes, a skeleton does not represent the shape because part of the shape lies outside of the image.

In these cases, a birthplace could take the place of the skeleton. A birthplace can be a point, a line segment, two or three consecutive line segments, a ring, or something more complex. However, only points and rings were implemented in this work.

The idea is to replace the nucleus with a birthplace. The nucleus takes the part of the skeleton of a big region that lies in a small one. Consequently, the birthplace is

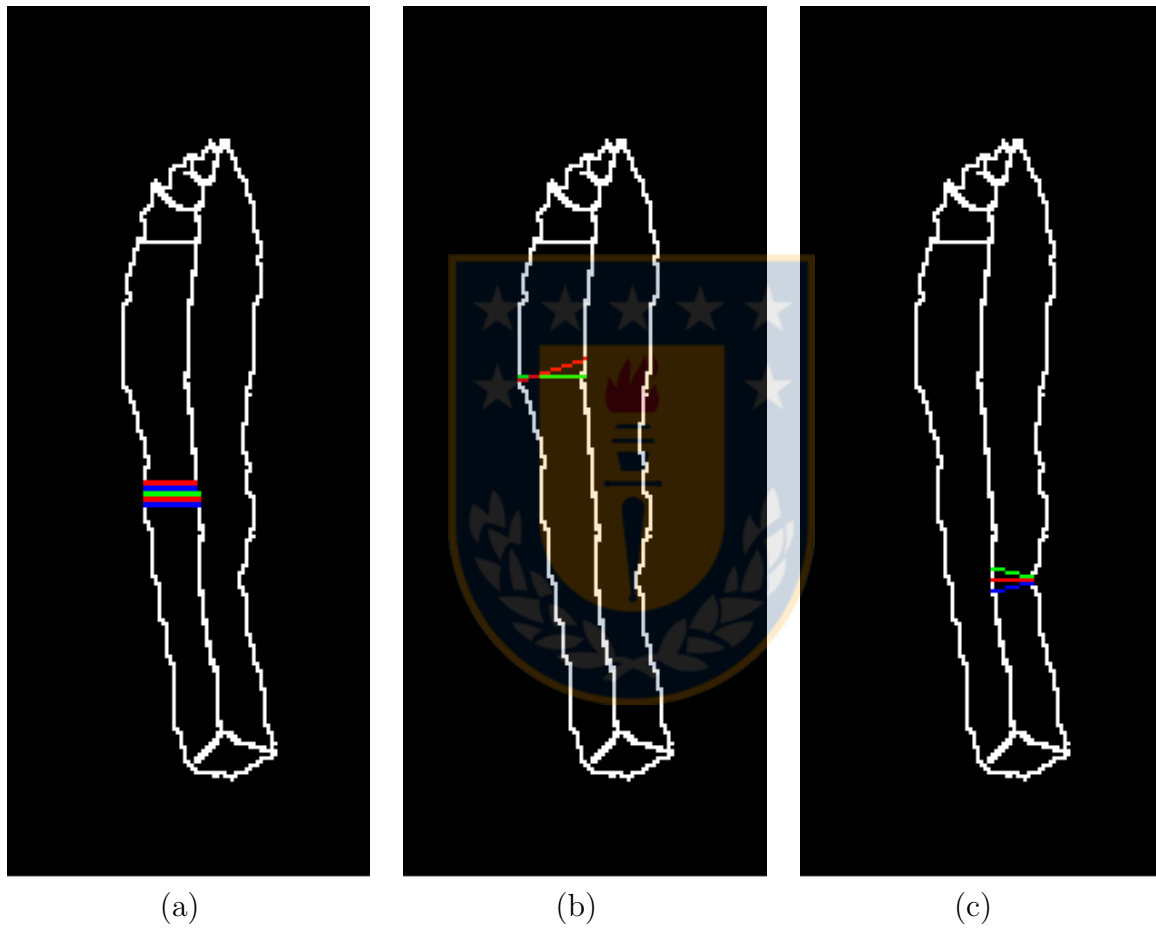


Figure 3.19: A leg with adjacent segments (a), with intersected segments (b), and with segments that leave empty areas (c)

Algorithm 5 trace_segments algorithm

```

function TRACE_SEGMENTS(camino, camino_siguiente: line segment,
  Rcolorinicial: colorRegion; Rbig, Rsmall, nucleo: binaryImage, operacion:
  string)
  Rcolorfinal: colorRegion
  camino_entre_medio: line segment
  // (1) to avoid unnecessary complex calculus, it checks that their distance
  // (1-norm) in the nucleus is greater than one.
  if block distance between the ends of camino and camino_siguiente > 1 then
    // (2) to avoid redrawing some pixels, it verifies that the segments do not
    // intersect.
    if camino and camino_siguiente do not intersect then
      // (3) it tries to intersect the lines that contain these segments.
      Find the intersection point (Interseccion) between the lines to which
      camino and camino_siguiente belong to.
      if this intersection point exists then
        // (4) it computes a path in the nucleus between the ends.
        Find the shortest path with 1-norm distance between camino and
        camino_siguiente along nucleo; store it in recorrido_nucleo and its
        length in indice.
        // (5) If this path exists, it computes new segments from the
        // intersection to the points in the path and traces the part of these
        // segments from the border to the path.
        if this path exists then
          // For each pixel in recorrido_nucleo except the first
          for i = 2:indice do
            // Find the segment between the intersection and recorrido_
            // nucleo(i).
            camino_entre_medio = Buscar_camino (Interseccion,
            recorrido_nucleo(i))
            // Trace segment along camino_entre_medio between the outer
            // border of the small or big region and the nucleus.
            Rcolorfinal = ponerPixeles_en_la_forma_small_o_big
            (camino_entre_medio, Rcolorinicial, Rcolorfinal, Rbig,
            Rsmall, operacion);
          end for
        end if
      end if
    end if
  end if
  return Rcolorfinal
end function

```

computed on the small region as computing it on the big region could leave it outside the small region.

Considering that shrinking applied on a region without holes gives a point (MSP) at or near the geometric center of the region, this method was chosen to compute the birthplace. The *compression and expansion between the outer border and a set of points* is explained in Sec. J.1. If a region has holes, shrinkage gives a ring. The *compression and expansion between the borders and rings* is explained in Sec. J.2.



Comparisons of Color Interpolation Methods

Finally, this work compared the interpolation methods programmed before, i.e. the new interpolation methods (see Chapter 3) and the existent ones (see Appendix D). To do this comparison, the image quality of the interpolated images was assessed. Although this comparison can be done subjectively or objectively [112], this work used the latter approach since the former is more burdensome and expensive [180](see more detail in Sec. A.11).

A method belonging to the full-reference class was used: the feature similarity color index (FSIM_C). It was chosen since it achieved the best overall performance in an evaluation of algorithms belonging to its class [233]. The methods belonging to this class compare a distorted image with a distortion-free one. The FSIM_C method evaluated the interpolated images by considering the second of three-consecutive images as the distortion-free image, and the different interpolations of the first and the third, as the distorted one [70].

4.1 Selecting Sequences of Images

Image interpolation has two application areas: temporal and spatial (see Sec. A.10). However, some sequences of images do not belong to any of them. Consequently, these

images should not be interpolated¹. For example, the images taken from a moving car belonging to the KITTI Vision Benchmark Suite [62]. Here, the images are taken both at different times and at different places.

In this work, the images were selected only for convenience (*convenience sampling*). Some sequences of three color images were taken from the Internet: walkcircle (frame0001.tif, frame0002.tif, frame0003.tif; processed as PNG images), walkstraight (frame0045.tif, frame0048.tif, frame0051.tif; processed as PNG images), and walkstraight_error (frame0040.tif, frame0045.tif, frame0050.tif; processed as PNG images) from [186]; army (frame07.png, frame10.png, frame13.png), basketball (frame12.png, frame13.png, frame14.png), beanbags (frame10.png, frame11.png, frame12.png), dog-dance (frame07.png, frame10.png, frame13.png), dumptruck (frame07.png, frame10.png, frame13.png), hydrangea (frame07.png, frame10.png, frame13.png), minicooper (frame07.png, frame10.png, frame13.png), and wooden (frame07.png, frame10.png, frame13.png) from a database for optical flow [13]. The initial and final images belonging to each sequence of images mentioned in this paragraph are shown below.



Figure 4.1: Initial (leftwards) and final (rightwards) images of the sequence called walkcircle

¹This kind of sequences of images should be processed by spatiotemporal interpolation methods.



Figure 4.2: Initial and final images of the sequence called walkstraight: Leftwards and rightwards, respectively



Figure 4.3: Initial and final images of the sequence called walkstraight_error: Leftwards and rightwards, respectively. This sequence is inappropriate for interpolation as there is too much change between the images. In the first image, the right leg and shoe are almost completely overlapped by the left ones; in the last one, the former are visible.

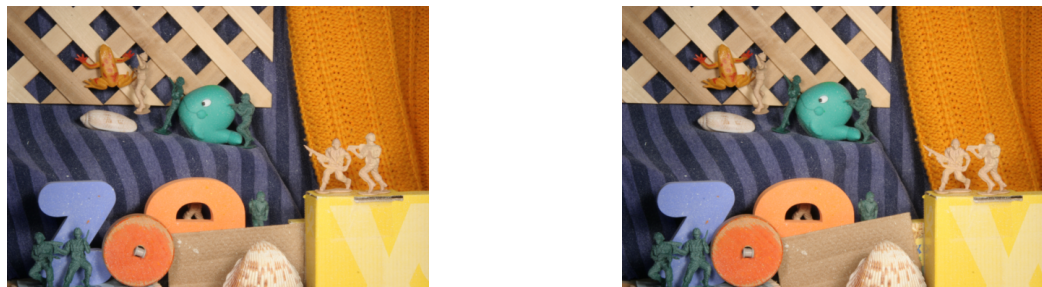


Figure 4.4: Initial and final images of the sequence called army: Leftwards and rightwards, respectively



Figure 4.5: Initial and final images of the sequence called basketball: Leftwards and rightwards, respectively

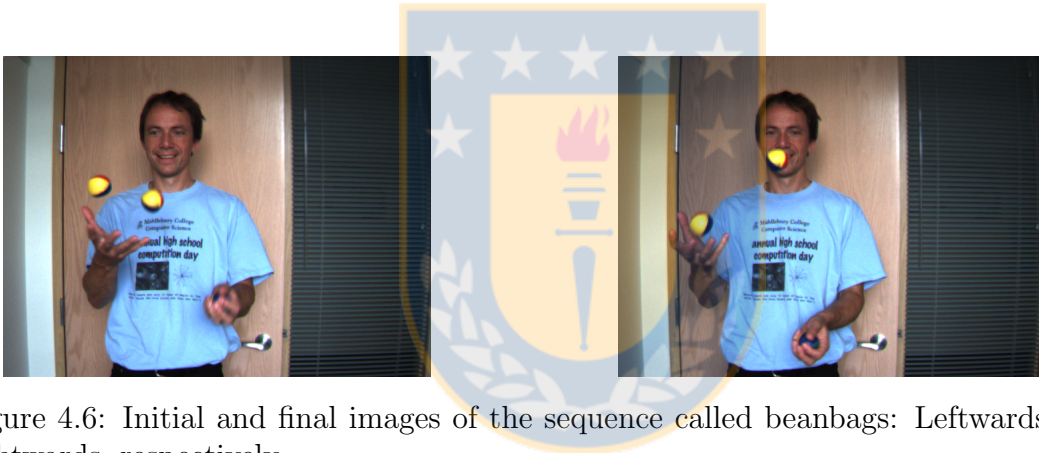


Figure 4.6: Initial and final images of the sequence called beanbags: Leftwards and rightwards, respectively

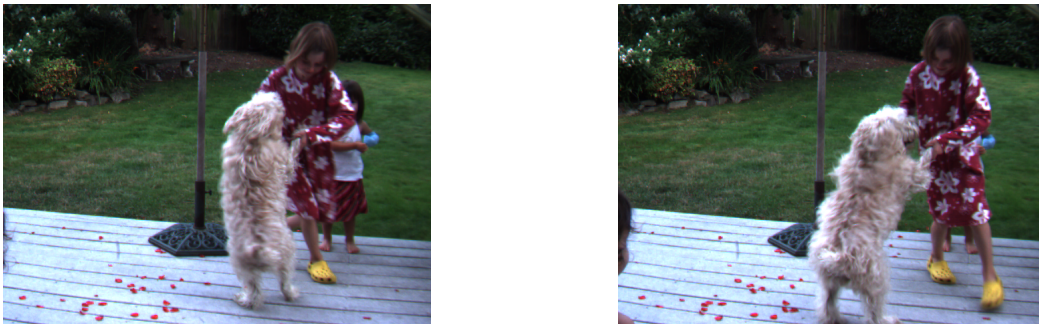


Figure 4.7: Initial and final images of the sequence called dogdance: Leftwards and rightwards, respectively



Figure 4.8: Initial and final images of the sequence called dumptruck: Leftwards and rightwards, respectively

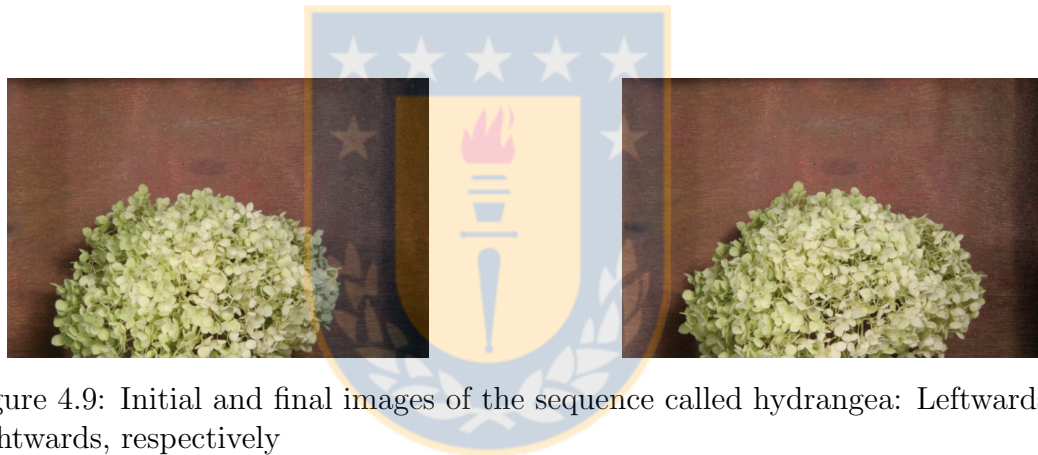


Figure 4.9: Initial and final images of the sequence called hydrangea: Leftwards and rightwards, respectively



Figure 4.10: Initial and final images of the sequence called minicooper: Leftwards and rightwards, respectively



Figure 4.11: Initial and final images of the sequence called wooden: Leftwards and rightwards, respectively

Other sequences of images were homemade: ceramic (ceramic00188.jpg, ceramic00189.jpg, ceramic00190.jpg), compact disc (CD169.jpg, CD170.jpg, CD171.jpg), Santa (santa505.jpg, santa506.jpg, santa507.jpg), walkingVGA (walking00124.jpg, walking00125.jpg, walking00126.jpg), and walking5M (walking5M411.png, walking5M412.png, walking5M413.png). The initial and final images belonging to each sequence of images mentioned in this paragraph are shown below.

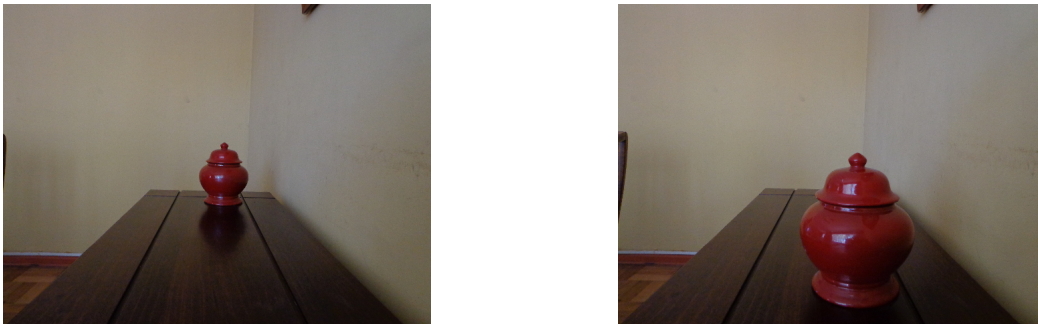
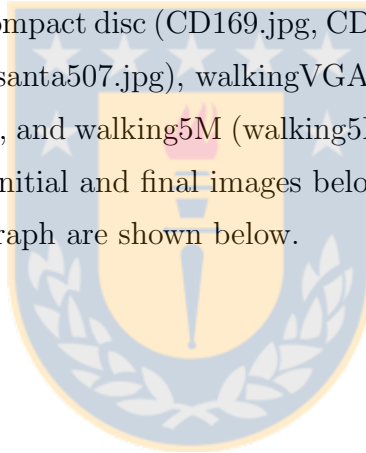


Figure 4.12: Initial and final images of the sequence called ceramic: Leftwards and rightwards, respectively

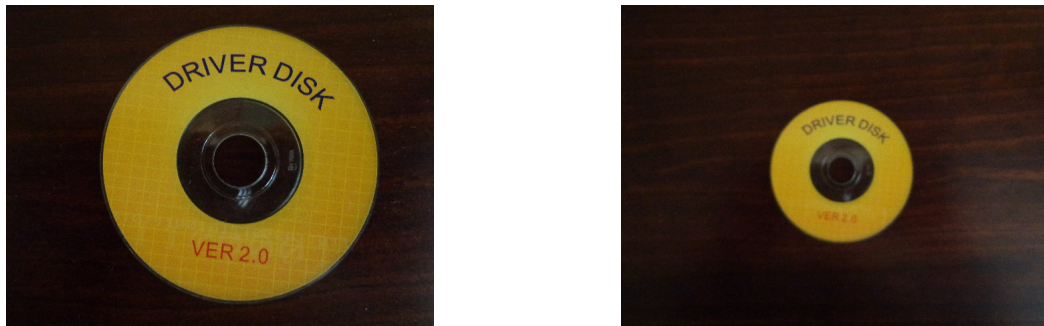


Figure 4.13: Initial and final images of the sequence called compact disc: Leftwards and rightwards, respectively

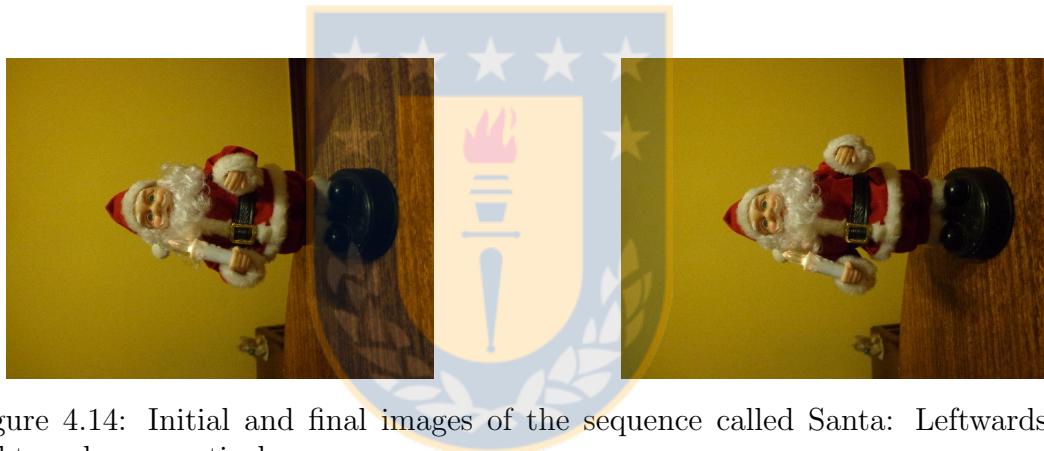


Figure 4.14: Initial and final images of the sequence called Santa: Leftwards and rightwards, respectively



Figure 4.15: Initial and final images of the sequence called walkingVGA: Leftwards and rightwards, respectively



Figure 4.16: Initial and final images of the sequence called walking5M: Leftwards and rightwards, respectively

4.2 Statistical Analysis

The objective is to compare the previous methods (linear, and median) with the new ones. Comparing interpolation methods is a little complex since some methods function with some images better than with others. Therefore, it is necessary to apply statistics to draw right conclusions. Specifically, to find if the new methods are better than the older ones. In other words, testing if the means of the new methods are greater than the means of the old ones.

It is necessary to choose a statistical test. This is a critical decision as it can change the interpretation of data [100]. If several tests are available, it should be chosen the test with the greatest power. Usually, parametric tests are more powerful than non-parametric ones but also have stronger assumptions. If these assumptions are not satisfied, it is better to use non-parametric tests [100].

As, in this work, the same images undergo different treatments, the samples can

be paired. In other words, two methods are applied to the same set of images, and the (two) results for each image are paired. For paired samples, a *paired difference test* (see Sec. H.7) should be applied. If a sample of differences comes from a normal population with standard deviation unknown, a t-test for correlated samples is appropriate [66]. Sometimes it is necessary to check normality (see Sec. H.6).

If there is no normality, it is compulsory to apply a non-parametric test, such as the sign or the Wilcoxon signed-rank test. The former test has no assumptions; the latter requires a symmetrical sample, i.e. a sample without skewness (see Sec. H.5).

All these tests require random samples. As the samples obtained are non-random, these test cannot be applied. Even worse, there is no statistically justified method for probability analysis and inference about the quality of the results obtained from convenience samples [7]. For these reasons, the statistical analysis was discarded.

4.3 Results and Discussion

The interpolation methods are not suitable to interpolate the sets of images taken when there is a forward or backward camera movement or a zooming. When a camera moves forward or zooms, it seems as expanding the first images and taking away picture-frame-like areas from the border of the next images. When a camera moves backward, it seems as compressing the first images and adding picture-frame-like areas around the next images (see Fig. 4.13). These camera movements also produce parallax errors so they should be tackled by methods designed to compensate for parallax. For this reason, the compact disc sequence of images was excluded from the analysis.

The same parallax problem, but to a lesser extent, occurs when a camera moves up, down, right or left. In this case, it is expected that after “segmenting”² the areas that are added or took away, the interpolation methods could be applied without

²In fact, this is not a segmentation problem but an image registration one.

changes. Although this phenomenon affects the army and dogdance sequences (they show a little horizontal movement), they were included in the analysis.

In the overlapping method, usually there were seams where the part interpolated linearly met the part interpolated with nearest-neighbor. For example, when only the dumptruck was interpolated (see Fig. 4.17), the interpolation of its rear wheels showed a seam where the part interpolated linearly –all the truck except the lower part of the rear wheels– met the part interpolated with nearest-neighbor –the lower part of the rear wheels, over the blurred area.

In this method, when an object included in a region was occluded, sometimes the interpolation was blurred. For example, in the chute that belongs to the dumptruck (see Fig. 4.17). It is owed to the fact that the grey-level interpolation algorithm uses the MSP to center the regions, but as part of the rear wheels are not visible in the first image, the computed MSP did not coincide with the real MSP. Then, the moved regions were not aligned.



Figure 4.17: Interpolation considering the dumptruck as a region

The interpolated images from the new methods depend on the segmentation of the images. For example, several interpolations for the dumptruck images were made. Only the Mercedes, its shade and the area enclosed by them were segmented. The Mercedes always was part of a region. The shade and the area were part of the same region as the Mercedes, or isolated regions, or part of the background. The rest of the image was always considered background. Consequently, four partitions were obtained.

The first partition considered the car, its shade and the enclosed area as separate regions. This interpolation gave the worst result. It was caused by the inaccurate segmentation of the car and the shade (this was inevitable as both areas are dark where they met). This impeded to compute the center of these regions correctly, so they were not well overlapped (see Fig. 4.18).



Figure 4.18: Interpolation considering the car, its shade and the enclosed area as separate regions

The second partition considered only the car as a region. This interpolation gave a much better result, but it was still bad. It was caused by the inaccurate segmentation

of the car and its shade. This impeded to compute the center of the car correctly, so it was not well overlapped (see Fig. 4.19).



Figure 4.19: Interpolation considering only the car as a region

The third partition considered the car, its shade and the enclosed area as a region. This interpolation got a better result. In this case, the center of the region was well computed, so it was moved correctly (see Fig. 4.20).

The last partition considered the car and its shade as a region and the enclosed area as another. This interpolation got the best result. The car and the shade had the same interpolation as in the previous case. The area was better interpolated (the strong contrast between the area and the other object allowed its accurate segmentation). It was possible to appreciate the difference between the areas in both interpolated images by applying zooming to them, but it was impossible to detect the improvement by doing so (it is necessary the ground truth image or the original images). Therefore, this interpolation is not shown.

It was tried to interpolate more regions of the dumptruck image with the new



Figure 4.20: Interpolation considering the car, its shade and the area as a region

methods. For example, in addition to the regions segmented in the last partition, it was added the dumptruck. The result was worse than those of the two previous cases (see the black areas around the regions in Fig. 4.21) because the grey-level interpolation considered all these regions as a complex connected component when they are not (the objects that they represent are independent). Consequently, it computed the interpolation of this component, what assigned some parts of the dumptruck, the Mercedes and its shade to the background and vice versa (see Fig. 4.22).

A table was segmented in the wooden sequence (see Fig. 4.23). Its interpolation with the overlapping and deforming methods is shown in Fig. 4.24. The interpolation of this table showed that the overlapping method did not consider the rotation of the objects (the shade within the table had the same orientation of the shades below the original tables). The deforming method was able to do a much better interpolation. Although it also did not consider the rotation of the objects, the texture of the table hid this.



Figure 4.21: Interpolation considering the car and its shade, the enclosed area, and the dumptruck as three separate regions



Figure 4.22: Grey-level interpolated mosaics belonging to the the car, its shade, and the enclosed area; to the dumptruck, and to all of them, respectively.

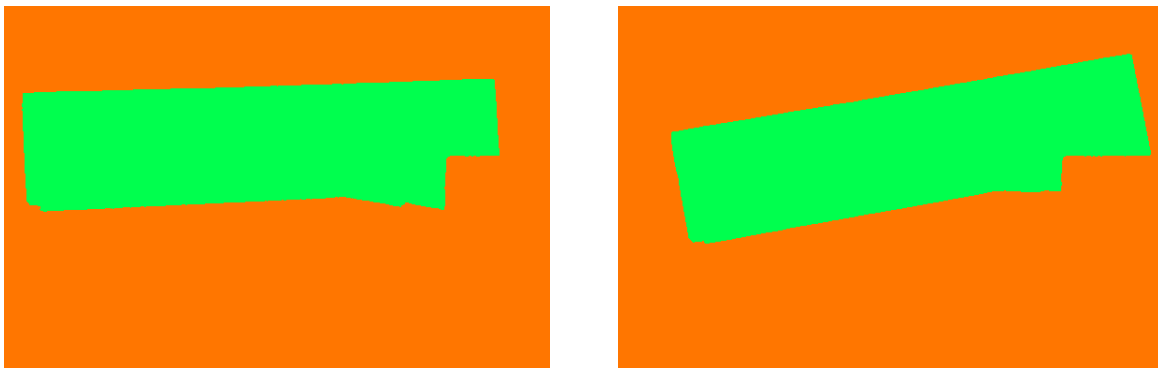


Figure 4.23: Table segmented in the wooden sequence



Figure 4.24: Table in wooden interpolated with overlapping and deforming; the rest of the image was interpolated with overlapping

The results obtained from the different interpolations are summarized in a couple of tables. When a sequence was segmented several times, only the best result was included. These are the identifications (Id.), names, and descriptions of the different interpolation methods used:

- (a) linear: Linear interpolation applied to the original images.
- (b) median: Median image generation for color images of Iwanowski and Serra applied to the original images.

From (c) to (s), each couple of paired regions can be interpolated with either *overlapping* or *deforming*. Moreover, there are three independent options to perform the latter (eight combinations). Therefore, to interpolate two images, there are seventeen possible combinations. The substrings in the name of the file describe which combinations were applied to the original images (see tables 4.1 and 4.2).

Although it is possible and sometimes recommendable to apply “Birthplace” to some regions and, at the same time, “Skeleton” to others, this variation was not implemented (the same happens with “CE” and “EC”, and “linear” and “median”).

The results of $FSIM_C$ computed on the different sequences of images are shown

Table 4.1: Methods applied according to its name

Substring	Methods applied
overlapping	<i>Overlapping</i> processed at least a couple of paired regions.
Birthplace	<i>Deforming</i> processed at least a couple of paired regions by deforming between the borders and the birthplace.
Skeleton	<i>Deforming</i> processed at least a couple of paired regions by deforming between the borders and the skeleton.

Table 4.2: Types of deforming applied to an interpolation according to its name

Substring	Type of deforming applied
CE	It was applied compression, then expansion.
EC	It was applied expansion, then compression.
linear	The linear interpolation method was applied to paired deformed regions.
median	The color median of Iwanowski and Serra was applied to paired deformed regions.

in tables 4.3 and 4.4³. Sometimes the best result between the new methods was obtained using exclusively either *overlapping*, such as in *dumtruck*, or *deforming*, such as in *army* and *wooden*. In this case, the results showed from l, \dots, s correspond either to c , or to d, \dots, k .

³It is almost sure that other segmentations would improve some of these results, but it is burdensome to test them (see Sec. 3.1). For example, by segmenting the shades of *walking5M* and *walkingVGA*.

Table 4.3: FSIM_C calculated for several interpolation methods (part I)

Id.	Interpolation method	army	basketball	beanbags	ceramic	dogdance	dumptruck	hydrangea
a	linear	0.878	0.924	0.935	0.927	0.809	0.930	0.780
b	median	0.931	0.944	0.941	0.923	0.802	0.936	0.783
c	overlapping	0.913	0.942	0.937	0.934	0.811	0.936	0.791
d	BirthplaceCElinear	0.928	0.916	0.889	0.924	0.780	0.927	0.788
e	BirthplaceCEmedian	0.929	0.929	0.896	0.916	0.771	0.926	0.781
f	BirthplaceECllinear	0.928	0.915	0.878	0.920	0.789	0.927	0.788
g	BirthplaceECmedian	0.929	0.929	0.881	0.913	0.778	0.924	0.781
h	SkeletonCElinear	0.918	0.914	0.913	0.896	0.785	0.928	0.786
I	SkeletonCEmedian	0.929	0.944	0.914	0.914	0.789	0.926	0.786
j	SkeletonECllinear	0.923	0.909	0.91	0.897	0.785	0.928	0.785
k	SkeletonECmedian	0.933	0.944	0.911	0.915	0.791	0.927	0.786
l	overlappingBirthplaceCElinear	0.928	0.945	0.938	0.935	0.816	0.936	0.792
m	overlappingBirthplaceCEmedian	0.929	0.951	0.937	0.935	0.811	0.936	0.792
n	overlappingBirthplaceECllinear	0.928	0.945	0.938	0.935	0.816	0.936	0.792

Continued on next page

Table 4.3 – Continued from previous page

Id.	Interpolation method	army	basketball	beabags	ceramic	dogdance	dumptruck	hydrangea
o	overlappingBirthplaceECmedian	0.929	0.951	0.936	0.935	0.810	0.936	0.792
p	overlappingSkeletonCElinear	0.918	0.946	0.936	0.934	0.817	0.936	0.792
q	overlappingSkeletonCEmedian	0.929	0.952	0.936	0.932	0.812	0.936	0.793
r	overlappingSkeletonCElinear	0.923	0.944	0.936	0.934	0.817	0.936	0.791
s	overlappingSkeletonECmedian	0.933	0.952	0.936	0.932	0.812	0.936	0.793



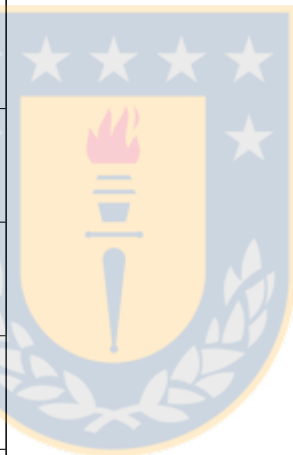
Table 4.4: FSIM_C calculated for several interpolation methods (part II)

Id.	Interpolation method	minicooper	Santa	walkcircle	walking5M	walkingVGA	walkstraight	walkstraight error	wooden
a	linear	0.882	0.918	0.945	0.758	0.820	0.902	0.895	0.782
b	median	0.892	0.919	0.952	0.753	0.827	0.893	0.880	0.843
c	overlapping	0.905	0.921	0.969	0.788	0.829	0.938	0.889	0.855
d	BirthplaceCElinear	0.889	0.915	0.923	0.731	0.789	0.871	0.846	0.861
e	BirthplaceCEmedian	0.891	0.913	0.924	0.726	0.789	0.861	0.838	0.860
f	BirthplaceEClinear	0.889	0.914	0.926	0.732	0.792	0.869	0.846	0.861
g	BirthplaceECmedian	0.891	0.912	0.925	0.730	0.792	0.858	0.839	0.860
h	SkeletonCElinear	0.896	0.899	0.941	0.747	0.804	0.89	0.878	0.852
I	SkeletonCEmedian	0.900	0.912	0.941	0.747	0.813	0.881	0.863	0.866
j	SkeletonEClinear	0.896	0.900	0.944	0.747	0.817	0.887	0.876	0.853
k	SkeletonECmedian	0.901	0.916	0.942	0.749	0.821	0.877	0.873	0.867
l	overlappingBirthplaceCElinear	0.909	0.920	0.969	0.792	0.829	0.938	0.921	0.861
m	overlappingBirthplaceCEmedian	0.909	0.919	0.968	0.791	0.829	0.938	0.921	0.860
n	overlappingBirthplaceEClinear	0.909	0.920	0.969	0.791	0.829	0.938	0.921	0.861

Continued on next page

Table 4.4 – Continued from previous page

Id.	Interpolation method	minicooper	Santa	walkcircle	walking5M	walkingVGA	walkstraight	walkstraight error	wooden
o	overlappingBirthplaceECmedian	0.909	0.919	0.968	0.791	0.829	0.937	0.922	0.860
p	overlappingSkeletonCElinear	0.910	0.920	0.969	0.788	0.829	0.938	0.921	0.852
q	overlappingSkeletonCEmedian	0.910	0.920	0.970	0.788	0.829	0.938	0.921	0.866
r	overlappingSkeletonEClinear	0.909	0.920	0.970	0.787	0.829	0.938	0.920	0.853
s	overlappingSkeletonECmedian	0.910	0.921	0.971	0.788	0.829	0.939	0.921	0.867



4.4 Descriptive Analysis

The average and standard deviation from the interpolation methods were computed using tables 4.3 and 4.4. The results are shown in table 4.5.

Table 4.5: Descriptive statistics of $FSIM_C$ for several interpolation methods

	Interpolation method	Average	Standard deviation
a	linear	0.872	0.065
b	median	0.881	0.064
c	overlapping	0.891	0.060
d	BirthplaceCElinear	0.865	0.064
e	BirthplaceCEmedian	0.863	0.067
f	BirthplaceECllinear	0.865	0.063
g	BirthplaceECmedian	0.863	0.065
h	SkeletonCElinear	0.870	0.060
I	SkeletonCEmedian	0.875	0.063
j	SkeletonECllinear	0.870	0.060
k	SkeletonECmedian	0.877	0.062
l	overlappingBirthplaceCElinear	0.895	0.060
m	overlappingBirthplaceCEmedian	0.895	0.061
n	overlappingBirthplaceECllinear	0.895	0.060
o	overlappingBirthplaceECmedian	0.895	0.061
p	overlappingSkeletonCElinear	0.894	0.060
q	overlappingSkeletonCEmedian	0.895	0.061
r	overlappingSkeletonECllinear	0.894	0.061
s	overlappingSkeletonECmedian	0.896	0.061

The average $FSIM_C$ value has been plotted in Fig. 4.25.

Fig. 4.25 shows that the overlapping method is better than the methods of reference (linear and median). At the same time, the median is better than the linear.

This graph also shows that the new interpolation methods with *deforming* by using the Birthplace and the Skeleton families of methods obtained worse results than the overlapping method. The reasons that explain these poor results are the distortions in static areas while trying to interpolate moving objects, and the grey and black areas where this interpolation method was not able to do so. This is not

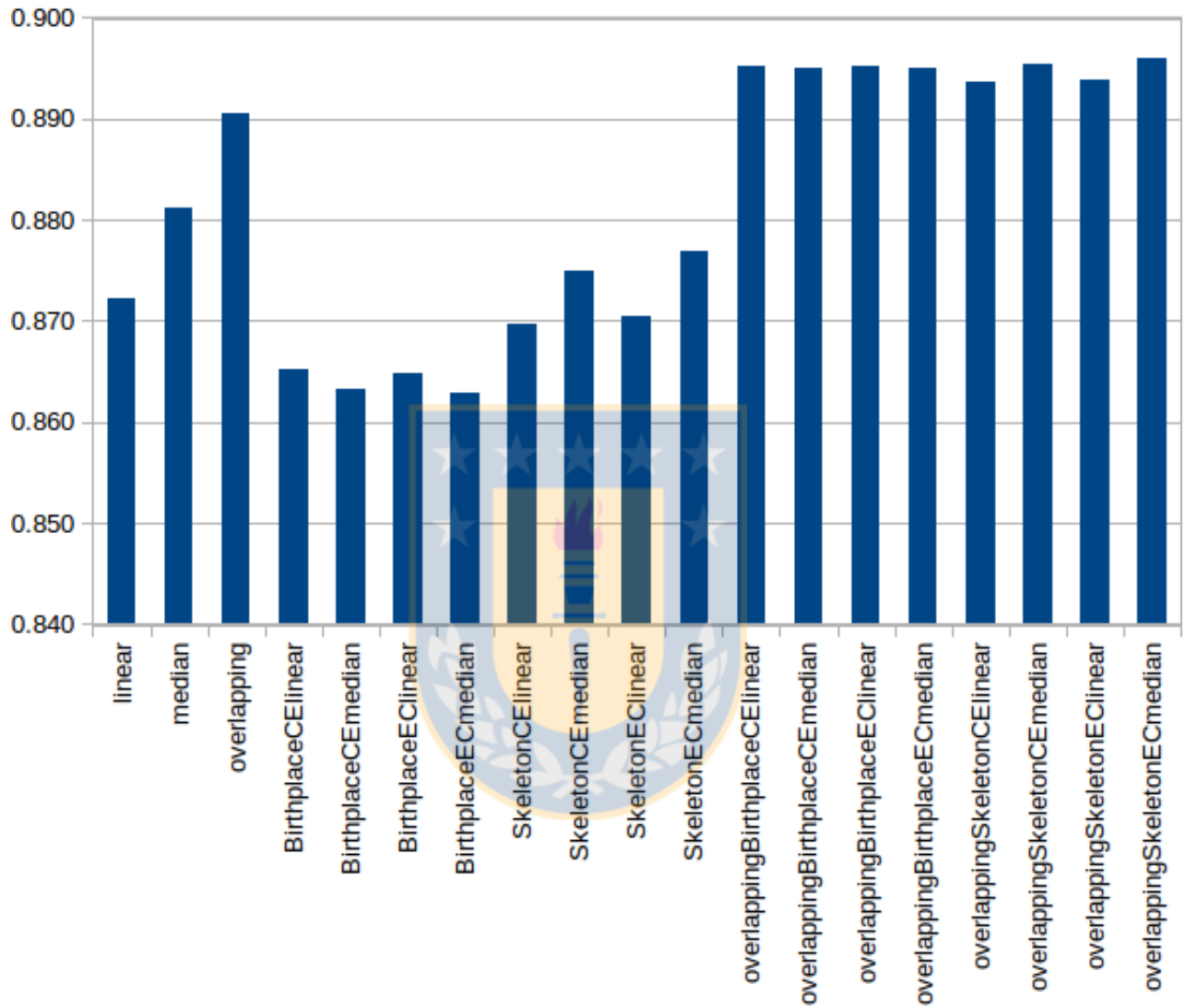


Figure 4.25: Average FSIMc per Interpolation Method

strange because the *deforming* algorithm should be applied to regions that change; not to regions that are static or only are translated. For example, in Fig. 4.27 all the regions were interpolated with deforming by using birthplaces and the segmentation of Fig. 4.26. As an example of distortion, the area near the ball —especially in the window— is deformed in the direction in which the ball moves (up and right). As an example of grey areas, see over the hands of the person waiting for the ball.



Figure 4.26: Segmentation of an image

Other facts that can be extracted from the tables 4.3 and 4.4 are shown below:

First, the new interpolation method with *overlapping* usually got better results than the linear one, and frequently outperformed the median one.

Second, the methods that can use both *overlapping* and *deforming* simultaneously always outperformed the linear one, and generally outperformed the median one.

Third, in these methods, computing segments between the borders of the big region and the birthplace gave similar results to computing segments between the



Figure 4.27: New interpolation method applying only deforming by using birthplaces borders of the big region and the nucleus. In particular, the methods that use birthplaces were better for the ceramic, and walking5M images. In ceramic, there was a ceramic jar that was moved forward; in walking5M there was a person that moved backward. It was clear that these methods have to zoom the aforementioned shapes, and the birthplace methods did it better.

For this reason, it is better to use other methods to process them. For example, image registration can detect that two shapes are related and compute how much zoom is needed to transform one into the other.

Fourth, in these methods, compressing and then expanding gave similar results to expanding and then compressing.

Conclusions

As the most important conclusion, it was proved that it is possible to construct better morphological interpolation methods for color images without using color morphological operators. In fact, a general method was constructed with two optional methods: overlapping and deforming.

The *overlapping* method is best suited for immobile matching regions (a set of corresponding regions whose MSP does not move between the slices) that show objects that do not change in shape, place, orientation or size between the slices (the regions can have different shapes; the objects can have different color attributes). For example, when the corresponding immobile regions show a stationary background. Additionally, it is usable when the mobile matching regions have the same shape, orientation, and size, i.e. their shapes, orientations, and sizes do not change, and they show the same static objects. For example, when the regions show a car moving horizontally.

The *deforming* method is best suited when the object within the matching regions changes its shape between the slices owed to rotation, enlarging, decreasing, or zooming. It was expected that this method could manage object deformations, but the set of images did not allow to prove this.

As the set of images was selected only for convenience, it is biased what impeded to apply statistical analysis. As this set is infinite, a very profound understanding of it is required to select a random sample. Alternatively, an “interpolation benchmark” should exist. Consequently, the results obtained comparing the interpolation methods cannot be extrapolated to all the images.

This set of natural and artificial images allowed to prove the different methods under different situations. However, the lack of complex connected components impeded to evaluate them in this case. In fact, this characteristic inherited from the grey-level interpolation algorithm sometimes was detrimental when this algorithm processed the images considering this kind of components when they did not exist. This result suggests that the grey-level interpolation algorithm must fit the kind of images to be processed.

The new interpolation method with overlapping overcame the linear, and median image generation ones. The reason is obvious: this new interpolation method can move objects while the other ones cannot.

In the methods that can use both overlapping and deforming simultaneously, compressing and then expanding gave similar results to expanding and then compressing. This result suggests to change these consecutive steps for another one that makes both operations simultaneously. This should reduce the time used by these operations by a half.

The methods that use birthplaces zoomed objects better than those methods using nuclei. Therefore, the objects that change size should be zoomed before applying the latter methods. After this improvement, it is expected that using nuclei would obtain better results than using birthplaces.

This chapter explains several concepts about images: properties, types, representations, and operations.

A.1 Properties of Color

The rays, to speak properly, are not colored; in them there is nothing else than a certain power and disposition to stir up a sensation of this or that color. (Newton, 1704, cited in [181])

As Newton suggested, light has no color but has some properties that human beings can perceive. Color vision is a complex phenomenon that even nowadays is not perfectly understood [8].

Colors can be described by hue, saturation, and lightness or brightness attributes. Sharma [181] defined them:

Hue. Attribute of a visual sensation according to which an area appears to be similar to one of the perceived colors: red, yellow, green, and blue, or to a combination of two of them.

Brightness. Attribute of a visual sensation according to which an area appears to emit more or less light.

Lightness. The brightness of an area judged relative to the brightness of a similarly illuminated area that appears to be white or highly transmitting.

Saturation. Colorfulness of an area judged in proportion to its brightness.

A.1.1 Color Constancy

The color of an object is the same although the amount of illumination varies. This effect is known as color constancy [181]. For example, a car has the same color in the sun and shade sides, and on sunny or cloudy days. Technically, hue and saturation are constants (hue is also invariant to viewing direction, object geometry, and highlights [63]).

On one hand, this property permits matching objects between images. If an object in an image has the same hue than an object in another image, it is possible that these objects are the same.

On the other hand, this property forbids some operations on images. For example, some morphological ones on RGB images.

Finally, this property gives a reason to compute some operations on color models different to RGB. For example, the linear interpolation since it is possible that interpolating two RGB pixels with the same hue gives another hue.

A.2 Digital Images

A digital image represents an image by using a set of bits. There are two main categories of digital images: vector and raster images [136].

A vector image represents an image by using mathematical equations such as lines, circles, polygons, and others [136]. The computer draws these shapes on an output device [30].

A raster image represents an image by using pixels [136]. This representation is discrete both in its spatial coordinates and in its values [184]. The spatial coordinates of an image are usually represented as a matrix with M rows and N columns [156, 139]. An *image element*, *picture element*, *pixel*, or *pel* is an element in this matrix [65]; the value it takes is called *image value* [103]. The computer draws these pixels on an output device.

As only raster images were used in this work, hereinafter only this kind of images are treated. The (image) values allow to classify the images in binary, grey-level and color.

A.2.1 Binary Images

A binary image is a digital image in which the points can have only two values (0 and 1). Usually, in a white support, these values are displayed as black and white, respectively [189]. An example of binary images is shown in Fig. A.1.

A.2.2 Grey-level Images

A grey-scale image is a digital image in which the points can have values between 0 and $2^n - 1$, where n is an integer [139] equal to the number of bits that store each value. In a white support, the points whose value is zero are displayed in black, the

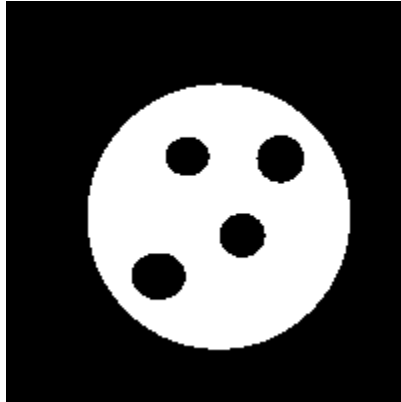


Figure A.1: Binary image

points whose value is the highest, in white, and the points whose values are in-between, in intermediate grey tones [189]. An example of grey-level images is shown in Fig. A.2.



Figure A.2: Grey-level image

A.2.3 Color Images

A color image is an image whose points map on a color space [52]. This representation is explained in Sec. A.3. An example of color images is shown in Fig. A.3.

A.2.4 Mosaic Images

A *partition* is a division of a set into non-empty disjoint subsets [189]. A *mosaic image* is a partition of an image [20]. Each part in a mosaic is called a facet, a tile, a cell or a particle [20, 86, 130].



Figure A.3: Color image

The values within each part (in a mosaic) can be different; however, if each part has pixels with the same value, there are two classes: grey-level and color mosaics.

Grey-level Mosaics

In grey-level mosaics, each tile has pixels with the same grey-value [210]. An example of grey-level mosaics is shown in Fig. A.4.

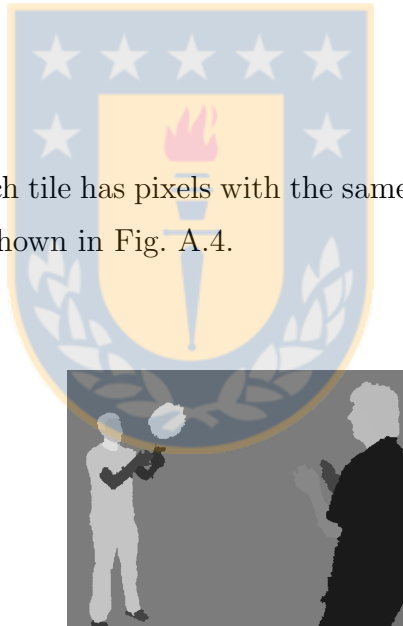


Figure A.4: Grey-level mosaic image

Color Mosaics

In color mosaics, each tile has pixels with the same color. An example of color mosaics is shown in Fig. A.5.



Figure A.5: Color mosaic image

A.3 Color Models and Color Spaces

A *color model* is an abstract mathematical model that permits representing colors by using a tuple of numbers [135]. Most color models represent colors by using three components; for example, RGB uses *Red*, *Green* and *Blue* components. A few color models use four or more components; for example, CMYK [165].

Color models cannot represent a defined color because they are abstractions. They need *references* in the real world, such as red, green and blue primary colors (hues) in the RGB color model. They also need a *scale*. For example, it is not enough to define the hues in the RGB color model; it is necessary to define their saturation. Appropriate quantities of these primary colors are added to form a color (this characteristic makes RGB an additive color model). A color model with references and scale is a *color space*. Thus, different color spaces can be defined using the same color model. For example, Adobe RGB and sRGB are based on the RGB color model [165].

It is impossible to reproduce some colors by mixing red, blue, and green colors. The chosen primaries determine which colors can or cannot be reproduced. *Gamut* is the colors that can be reproduced within a color space or by a particular device [165].

Consequently, color models are distinct from color spaces. In spite of this, some authors use these terms indistinctly¹.

¹In this work, these concepts are used distinctly.

In *uniform color spaces*, same distances between coordinates gave similar perceived color differences [103]. In other words, in this case the color space has perceptual uniformity. This is an important property in color spaces since it permits accurate linear image interpolation.

The color models and color spaces used directly and indirectly in this work are described below.

A.3.1 RGB Color Model

The tristimulus theory of vision states that any color can be matched by an additive combination of the three primary colors [143]. The RGB color model is based on this theory [72]. Thus, each color is a mixture of red, green, and blue components [103].

The RGB color model is well suited for hardware such as monitors and cameras [65]. In the case of capturing devices, it is usual that three distinct filters are used what determines a color space specific for each device [165]. In the case of CRTs, the chromaticities of their phosphors determine their gamuts. Thus, two CRTs with distinct phosphors cover different color spaces [55].

Unfortunately, the values captured by capturing devices are linear RGB intensities. These values are perceptually non-uniform. Therefore, they are transformed into non-linear RGB values by using gamma correction. The new values are denoted $R'G'B'$ [154]. Therefore, the RGB images in a computer are in fact $R'G'B'$ ones².

Although the image processing literature does not discriminate between RGB and $R'G'B'$ [154], this work does a distinction. When a properly weighted value of linear RGB components is computed, relative luminance, Y , is obtained; when a properly weighted value of $R'G'B'$ components is computed, luma, Y' , is obtained [156]³. In

²In this work, linear RGB is mentioned only in this section and in Sec. A.3.4.

³To avoid confusions, the prime should always be present to denote luma [156].

this work, all RGB images are, implicitly, $R'G'B'$ ones. Hence, luma is used.

A.3.2 HSV Color Model

As humans describe colors by hue, saturation, and brightness, the RGB color model is not appropriate to describe colors to them [65]. Therefore, the HSI, HSL, and HSV color models were introduced to describe colors in a time when the colors were specified numerically instead of visually [88, 205]. In general, these color models should not be used nowadays⁴.

All these models are obtained from the RGB color model by coordinate conversion [154]. These models take hue, saturation, and “brightness” values to obtain RGB values or vice versa.

The HSV (also called HSB) coordinates are hue, saturation, and value (brightness). It is very intuitive and simple [103].

A.3.3 CIELAB

The *International Commission on Illumination* (*Commission internationale de l'éclairage*) recommended the CIELAB and CIELUV as approximately uniform color spaces [103].

CIELAB coordinates are (L^*, a^*, b^*) ; CIELUV coordinates are (L^*, u^*, v^*) . In both spaces, L^* corresponds to lightness.

In this work, CIELAB was utilized to interpolate pixels.

⁴The descriptions and formulae of the HSI, HSL, and HSV models disregard the principles of color science [156]. In addition, HSL and HSV are not useful for quantitative image analysis [74].

A.3.4 YIQ Color Space

The *YIQ color space*, also known as *NTSC color space* [2], is a device-dependent color space developed for analog television and video systems [154, 125, 238, 121]. It keeps up compatibility with previous monochromatic systems and saves transmission bandwidth [78, 154, 121].

It has three components: *luma*, Y' ; *in-phase*, I ; and *quadrature*, Q . Y' represents the achromatic information: it approximates the lightness, L^* , but is different. Only this component is used in monochrome systems. I and Q represent the chromatic information: I is an orange-cyan axis, and Q is a magenta-green axis [154, 156].

Gamma-corrected RGB , $R'G'B'$, is converted to YIQ by using the formula [157]:

$$\begin{bmatrix} Y' \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.29889531 & 0.58662247 & 0.11448223 \\ 0.59597799 & -0.27417610 & -0.32180189 \\ 0.21147017 & -0.52261711 & 0.31114694 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \quad (\text{A.1})$$

A common matrix that is almost equal to the matrix showed in Eq. A.1 but rounded to three digits (the element in (3,3) does not follow this rule) is shown in Fig. A.6 [?, 229, 79, 103, 45]⁵. FSIM_C (see Sec. A.11.1) is an algorithm for comparing two images; it uses this matrix to process each component (Y' , I , Q) separately.

$$\begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix}$$

Figure A.6: Common matrix to convert $R'G'B'$ into YIQ

In addition, there exist other matrices to convert $R'G'B'$ into YIQ [162, 55, 94, 78] [48, 26, 125, 82]. They are almost equal to the matrix showed in Fig. A.6, but

⁵This is the same matrix used by `rgb2ntsc` in Matlab [121].

the last digit of some elements is different.

The conversion from YIQ to $R'G'B'$ is done by using the inverse of the matrix used to convert $R'G'B'$ into YIQ [55, 78].

A.4 Basic Concepts: Neighborhood, Neighbors, Connected Pixels, and Connected Component

The neighborhood of a pixel is the region of pixels within a given *radius* around that pixel [102].

A pixel p at coordinates (x,y) has two vertical and two horizontal neighbors whose coordinates are $(x+1,y)$, $(x-1,y)$, $(x,y+1)$, and $(x,y-1)$. This set of neighbors is called *4-neighbors* or *four pixel neighborhood* of p . Also, it has four diagonal neighbors whose coordinates are $(x+1,y+1)$, $(x+1,y-1)$, $(x-1,y+1)$, and $(x-1,y-1)$. The set of all the neighbors of p is called *8-neighbors* or *eight pixel neighborhood* of p [148, 65]. For example, the pixel labeled p has as 4-neighbors, the pixels labeled c , and it has as 8-neighbors, the pixels labeled C (see Fig. A.7).



Figure A.7: Neighbors of a pixel: (a) 4-neighbor and (b) 8-neighbor pixels of p

Two pixels x_0, x_n with the same value are *4-(8)connected* if and only if there exists a sequence of pixels x_0, x_1, \dots, x_n , such each x_{i+1} is in the 4-(8)-neighbors of x_i and x_i have the same value as x_0 and x_n , $\forall i \in 0, 1, \dots, n-1$ [198].

A *connected component* is a set of all the pixels connected to each other [198]. A

formal definition can be seen in [120].

A.5 Convolution

Convolution is a mathematical operation that can be applied to continuous (integral convolution) or discrete systems (discrete convolution) [115, 92].

The two-dimensional convolution is fundamental to extract information from images [145, 92]. It can be used at least in image filtering, image enhancement, image restoration, and feature extraction. In this work, discrete convolution is used to compute the image gradient (see Sec. A.6). The operation depends on a set of weighting coefficients that can be stored in a matrix called *convolution matrix*, *filter mask*, or *convolution kernel* [51, 92, 145].

Given a convolution kernel K , the procedure to compute 2D convolution on an image I is [145]:

For each pixel P on the image I :

- place the center of K on P ,
- multiply each value of K with the value of the pixels under it,
- add all the products, and
- place the result into position P of the output image.

An example taken from [36]⁶ that shows this procedure to compute one pixel is shown in Fig. A.8. The center of the kernel K is at (2,2). It is placed on the black pixel in I .

It is possible to compute the convolution by putting the center of K out of the image I what creates an output image greater than I [92]. In this work, this situation

⁶http://www.fch.vutbr.cz/lectures/imagesci/includes/harfa_screenshots_overview.inc.php

39	33	35	36	31
35	34	36	33	34
34	33	36	34	32
32	36	35	36	35
33	31	34	31	32

 \otimes

1	2	1
2	4	2
1	2	1

 $= \begin{cases} 1 \cdot 34 & 2 \cdot 36 & 1 \cdot 33 \\ 2 \cdot 33 & 4 \cdot 36 & 2 \cdot 34 \\ 1 \cdot 36 & 2 \cdot 35 & 1 \cdot 36 \end{cases} = \{139 + 278 + 142\} = 559$

I **K**

Figure A.8: Example of convolution

does not happen so it has been overlooked.

In addition, there is a problem to compute the convolution at, and near, the borders of I because part of the kernel lies out of I . Although, there is no perfect method to solve this problem —except avoiding that objects of interest lie near the borders, the solution is adding pixels next to the borders of I . These extra pixels can be obtained copying the border pixels, copying the pixels of the opposite border, or writing zeros there [89].

A.6 Image Gradient

For a two-variable function f , the gradient of f , denoted by ∇ , is defined as $\nabla = (\partial f / \partial x, \partial f / \partial y)$ [192]. This definition shows that the gradient for a two-variable function gives a two-dimension vector whose components are its partial derivatives.

Image gradients are often used for edge detection [157]. In this work, they are used by $FSIM_C$ to compute the gradient magnitude (see Sec. A.11.1) and by *SegmentIt* to segment (see Sec. C.2.1). Besides, in this work, image gradients are used to compute perpendiculars (normals) to some edges in binary images.

The image gradient cannot be computed analytically, but the (discrete) derivatives can be computed by convolution [82]. There are several kernels to compute the convolution. A kernel construction method is explained that follows Hunt ideas [82].

Suppose a small 3 x 3 window within a picture, as shown in Fig. A.9. In this

case, the picture is represented as an array: the numbering of rows grows from top to bottom, and the numbering of columns, from left to right. The value of the pixel at (i, j) is $a_{i,j}$. The distance between adjacent pixels is h .

$a_{i-1,j-1}$	$a_{i-1,j}$	$a_{i-1,j+1}$
$a_{i,j-1}$	$a_{i,j}$	$a_{i,j+1}$
$a_{i+1,j-1}$	$a_{i+1,j}$	$a_{i+1,j+1}$

Figure A.9: 3 x 3 window

The objective is to find the derivatives around the center of this window, i.e. $a_{i,j}$. The *horizontal derivative* can be approximated by $\frac{1}{2h}(a_{i,j+1} - a_{i,j-1})$ (also known as *column gradient* [157]) and the *vertical derivative* by $\frac{1}{2h}(a_{i+1,j} - a_{i-1,j})$ (also known as *row gradient* [157]) [18, 82]. The coefficient $\frac{1}{2h}$ is often omitted because it is needed only the gradient direction or the approximate gradient magnitude. The terms order in the difference determines the direction of the derivation: It comes from the subtrahend to the minuend. Here, the horizontal derivative (column gradient) goes rightwards, and the vertical derivative (row gradient), downwards.

The convolution kernels corresponding to the simplified differences are shown in Figs. A.10 and A.11.

$$\begin{vmatrix} -1 & 0 & 1 \end{vmatrix}$$

Figure A.10: Horizontal derivative kernel

$$\begin{vmatrix} 1 \\ 0 \\ -1 \end{vmatrix}$$

Figure A.11: Vertical derivative kernel

This simple derivation considers only two pixels, so a single incorrect pixel changes the derivative. In other words, the horizontal and vertical derivative approximations

are sensitive to noise. To improve the noise immunity, it can be used more pixels within the 3 x 3 window [82].

Usually, image processing textbooks recommend the 3x3 kernel defined by [18]

$$\frac{1}{2h(w+2)} \begin{bmatrix} -1 & 0 & 1 \\ -w & 0 & w \\ -1 & 0 & 1 \end{bmatrix} \quad (\text{A.2})$$

Some methods that consider the other pixels are the Prewitt ($w=1$), Sobel ($w=2$), and Scharr ($w=10/3$) operators [18]. They provide differencing and smoothing simultaneously [26, 96]. Smoothing reduces noise what is an advantage considering that derivatives augment noise [32]. These operators work on binary and grey-scale images, but also on the luma of color images.

Each gradient has a magnitude and an orientation. The magnitude and orientation can be computed with arrays. Accordingly, $G_C(j, k)$ is the *column gradient* and $G_R(j, k)$ is the *row gradient*, at the row j and column k . In this case, rows grow downwards, and columns grow from left to right [157].

The *magnitude* can be computed as

$$|G(j, k)| = \sqrt{G_C(j, k)^2 + G_R(j, k)^2}. \quad (\text{A.3})$$

It also can be computed with l_1 or l_∞ norm [26].

The *orientation* can be computed as the angle θ with respect to the row axis using [157]

$$\theta(j, k) = \arctan\left\{\frac{G_C(j, k)}{G_R(j, k)}\right\}. \quad (\text{A.4})$$

In addition, they can be computed with the usual Cartesian coordinates. Accordingly, $G_x(x, y)$ is the gradient in the x-direction, i.e. the horizontal derivative (it goes rightwards) at the position (x, y) ; and $G_y(x, y)$ is the gradient in the y-direction, i.e.

the vertical derivative (it goes upwards) at the position (x, y) .

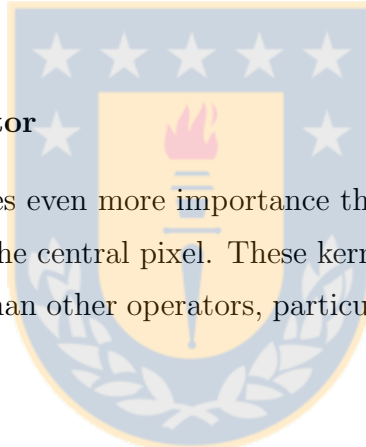
The *magnitude* can be computed as [17]

$$|G(x, y)| = \sqrt{G_x(x, y)^2 + G_y(x, y)^2}. \quad (\text{A.5})$$

The *orientation* angle ϕ can be computed as [17, 176]

$$\phi(x, y) = \arctan\left(\frac{G_y(x, y)}{G_x(x, y)}\right). \quad (\text{A.6})$$

Note that in the same point, $G_x = G_C$ and $G_y = -G_R$.



A.6.1 Scharr Operator

The Scharr operator gives even more importance than the aforementioned operators to the difference across the central pixel. These kernels give more accurate estimates of gradient orientation than other operators, particularly the Sobel operator (see Appendix F) [95].

The Scharr kernels are shown in Figs. A.7 and A.8 [18]. Notice that G_x goes rightwards, and G_y , upwards.

$$G_x = \frac{3}{32} \begin{bmatrix} -1 & 0 & 1 \\ -\frac{10}{3} & 0 & \frac{10}{3} \\ -1 & 0 & 1 \end{bmatrix} \quad (\text{A.7})$$

$$G_y = \frac{3}{32} \begin{bmatrix} 1 & \frac{10}{3} & 1 \\ 0 & 0 & 0 \\ -1 & -\frac{10}{3} & -1 \end{bmatrix} \quad (\text{A.8})$$

For the same reason given for the Prewitt operator, approximate Scharr masks are frequently used.

In this work, the Scharr operator is used by FSIM_C to compute the gradient magnitude (see Sec. A.11.1).

A.6.2 Kroon Operator

Image gradients can also be computed with larger kernels. Larger kernels reduce noise but increment the error in the edge location and the interference of objects in the neighborhood [177]. In this work, gradients are not used to find edges since, in binary images, the edges are the 1 pixels with at least one 0 neighbor; the gradients are used to compute the direction of the normals along binary image borders, so it is very important to choose an operator that gives accurate directions.

Kroon found the 5 x 5 sized derivative kernel with the minimal angle error [105]:

$$G_y = \begin{vmatrix} 0.0007 & 0.0052 & 0.0370 & 0.0052 & 0.0007 \\ 0.0037 & 0.1187 & 0.2589 & 0.1187 & 0.0037 \\ 0 & 0 & 0 & 0 & 0 \\ -0.0037 & -0.1187 & -0.2589 & -0.1187 & -0.0037 \\ -0.0007 & -0.0052 & -0.0370 & -0.0052 & -0.0007 \end{vmatrix} \quad (\text{A.9})$$

Notice that G_y goes upwards. The kernel G_x is obtained rotating this kernel 90° (it goes rightwards).

A.7 Distance

Distance is a measure of the space between two points. The usual notion of distance is the length of the line segment between two points in the space, the *Euclidean distance*. In this work, the *geodesic distance* and the *Hausdorff distance* are also mentioned.

A.7.1 Geodesic Distance

The geodesic distances are constrained to follow the surface of the earth. The distance between cities is given as a geodesic one. This is the relevant distance to travel between them as the Euclidean distance passes through the earth. For example, the red line in Fig. A.12 shows the geodesic distance between Seattle and London.

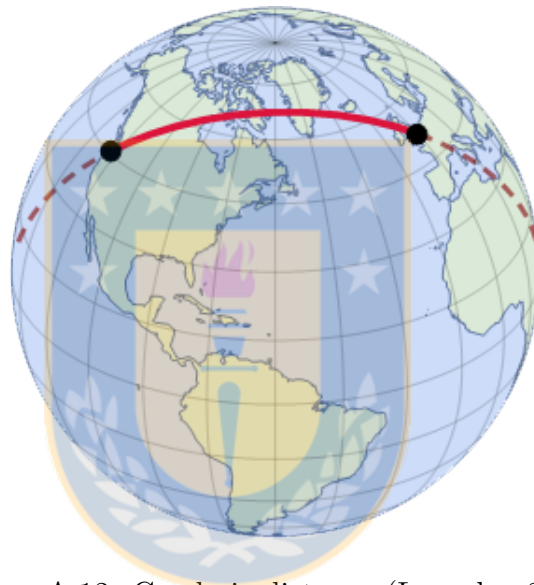


Figure A.12: Geodesic distance (Lysenko, 2012)

The same notion is applied to images. When a path is constrained to remain within a subset of an image, the relevant distance is a geodesic one. Formally, let A a set. The geodesic distance $d_A(p, q)$ between two pixels p and q in A is the shortest path(s) $P = (p_1, p_2, \dots, p_l)$ that joins p and q which is included in A . If L denotes the function that computes the length of a path:

$$d_A(p, q) = \min\{L(P) \mid p_1 = p, p_l = q, \text{ and } P \subseteq A\} \quad [189] \quad (\text{A.10})$$

If there is no path between p and q , the distance can be considered infinite.

A.7.2 Hausdorff Distance

The Euclidean and the geodesic distances are distances between points. However, it is possible to define distances between sets what the Hausdorff distance does. This distance uses the concept of dilation that is explained in Sec. B.3. Let X and Y be two sets. The Hausdorff distance is the minimum of the radius λ of the discs B such that X dilated by B_λ contains Y and Y dilated by B_λ contains X :

$$d_H(X, Y) = \min\{\lambda | X \subseteq \delta_{B_\lambda}(Y), Y \subseteq \delta_{B_\lambda}(X)\} \quad (\text{A.11})$$

where B_λ denotes a disc of radius λ [189]. For example, there two lines X and Y in Fig. A.13 . The dilation of Y with a disc of radius equal to the dotted line shown in the bottom includes X . The dilation of X with a disc of radius equal to the dotted line shown in the top includes Y . The Hausdorff distance is equal to the greater of these radii.

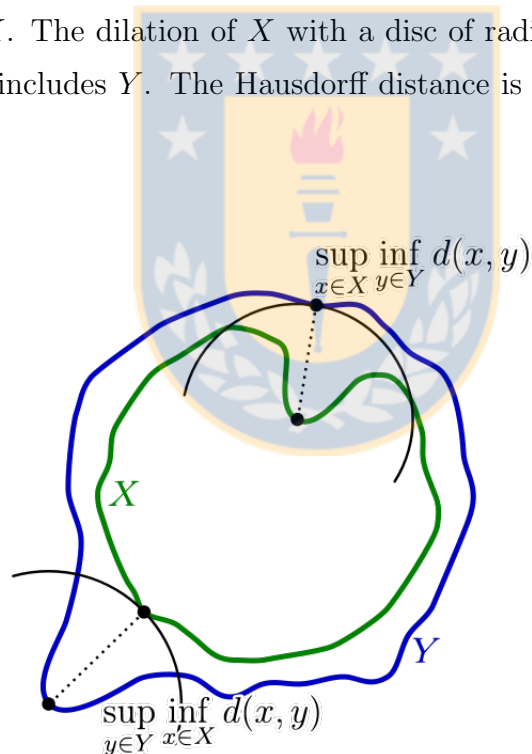


Figure A.13: Hausdorff distance sample (Rocchini, 2007)

A.8 Homotopy

It is easier to understand this concept understanding first the concept of *homotopy tree* —also known as the *adjacency tree*— since two images are homotopic when they

have the same homotopy tree [189]. Although homotopy trees can be applied to bounded sets of the Euclidean plane IR^2 [189], this work only applies them to discrete 2-D binary images.

In a discrete 2-D binary image there are connected components belonging to the foreground and to the background. Each of them is a node of the homotopy tree. In this case, it is assumed that the pixels outside of the image are background pixels. Hence, there exists a connected component, belonging to the background, denoted $S_{outside}$, that embeds (a connected component C_1 is embedded in another connected component C_2 if every 4-path from C_1 to $S_{outside}$ passes through C_2) all the other components. This component is the root of this tree. Each vertex of this tree represent both a relationship of embedding and of 4-adjacency. When a node represents a connected component of the foreground (alternatively, a connected component of the background), its children are connected components belonging to the background (alternatively, to the foreground) that are both embedded and adjacent. Hence, $S_{outside}$ has as children the connected components belonging to the foreground that are adjacent (and embedded) to itself [189].

An example of a set and its homotopy tree is shown in Fig. A.14. The root has as children the connected components A , B , and C . The component A has as children the connected components, belonging to the background, D and E . D has as child the connected component, belonging to the foreground, G [207].

A.9 Image Segmentation

Image segmentation is a process that divides (partitions) an image into a set of non-overlapping, connected image areas, called regions, such each region has similar characteristics [43, 103, 189]. It is also possible to consider segmentation as labeling each pixel of an image [183].

Two-label images have two distinct labels; multi-label images has several or many

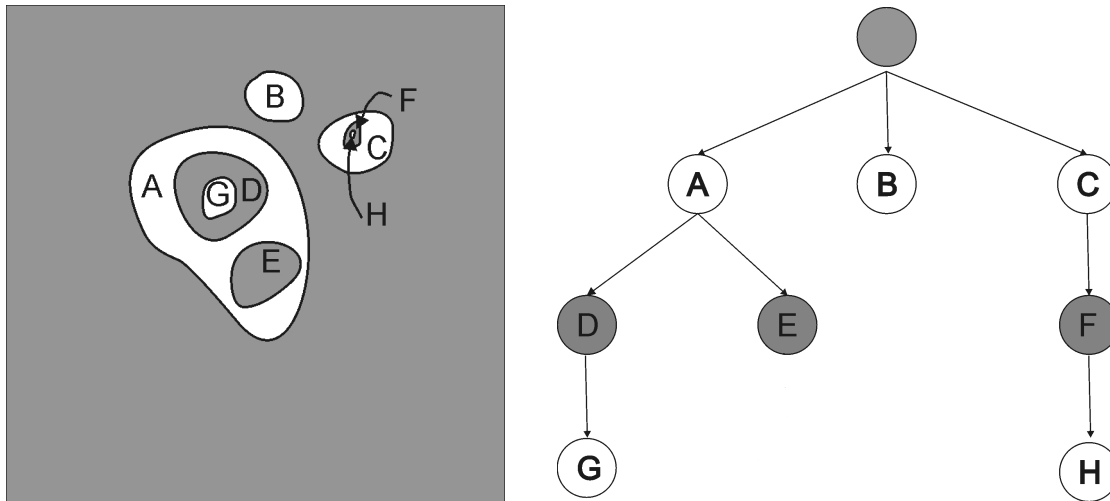


Figure A.14: Homotopy tree of a set: The set (left) and its homotopy tree (right).

distinct labels. Usually, in two-label images, one label represent the foreground and the other the background. In addition, usually, some labels represent objects and one label represent the background what someones called *figure-(back)ground segmentation* [109].

Two or more regions might have the same label in an image. For example, two regions labeled “tree”.

Image segmentation returns regions with and without interest for the user. Each of the former regions is called a *region of interest* (ROI) [80]. Sometimes, a ROI is segmented into many small regions; this phenomenon is called *over-segmentation* [133].

Image segmentation has several goals. For instance, the usual goal is that each of the regions corresponds to a different object [42]. In other words, image segmentation aims to partition an image into its constituent ROIs [232].

While some authors consider that image segmentation, also known as *scene segmentation*, differs from figure-(back)ground segmentation, also known as *object segmentation* [109], others consider that the latter is only a category within the former [219]. In this work, it is followed the second approach: what some called a figure

is a ROI and what they called (back)ground is not.

As there is not a general theory for image segmentation [144, 157, 121], many strategies have been proposed to segment an image [43]. As a consequence, many algorithms and techniques have been proposed [236]. They are so numerous that choosing one has become a laborious and time-consuming task [126].

There are two fundamentally distinct strategies for image segmentation, i.e. edge-based and region-based [12, 153]. *Edge-based* methods try to find the edges of the regions by looking for significant contrast changes (the regions are defined implicitly by the enclosing edges); *region-based* methods try to classify pixels by using some property (the edges are defined implicitly by the borders of the regions) [12]. These strategies can be applied either globally or locally (Bailey, 1991, cited in [12]). Global methods operate on all the pixels of an image; local methods operate in the part being segmented [12]. Local methods start with seeds belonging to either the edge or the region, and extend them [12]. As a result, there are four categories of segmentation methods; they are shown in table A.1.

Table A.1: Categories of image segmentation methods

	Edge-based	Region-based
Global processing	Edge detection and linking	Thresholding
Incremental processing	Boundary tracking	Region growing

Edge detection and linking: Edges are associated with the borders of objects. *Edge points* are points belonging to edges. Unfortunately, edge points —due to noise— seldom form closed connected boundaries. So, a linking process is usually required to connect the edge points [225].

Boundary tracking: Starting from a (border) seed, new border points are searched in the local neighborhood. If several points are available, one is chosen arbitrarily. For example, in a grey-level image, it is possible to compute its gradient. A point with a maximum gradient is chosen as seed. The adjacent pixel with a maximum

gradient is the next point in the border. From the adjacent pixel, the next maximum gradient is sought. This process continues until the seed is found [223].

Thresholding: There are two options [228]:

1. Single level thresholding and
2. Multi level thresholding.

These methods are easier to explain for grey-level images. *Single level thresholding*: If there are a set of light objects on a darker background, it is possible to separate the objects by taking a unique grey value greater than the background values and lesser than the values of the objects. *Multi level thresholding*: If there are objects with distinct grey-levels on a darker background, they can be separated taking a set of grey values [228].

Region growing: Starting from a (region) seed, new pixels are added to the region until a termination condition is met [12]. An algorithm belonging to this category is the watershed segmentation (Serra, 1982, cited in [158]).

As there exists no accepted taxonomy to classify image segmentation methods [121], it was sought a classification useful to select a segmentation method for the original images.

Image segmentation methods have been classified into manual, automatic, and semi-automatic [237]⁷. In manual segmentation, the user traces contours with a pointing device [158]. This is often a tedious, time-consuming and prohibitively expensive process [235, 39, 214]. Manual segmentation is used for creating ground truth to evaluate the other categories of segmentation [197]. In automatic segmentation, the user does not intervene. This segmentation type is unsatisfactory in many real situations [199, 232], in particular for multi-label segmentation [44]. In semi-automatic segmentation, the user guides (directs) this process either by entering parameters or

⁷This paper and this work have different definitions of semi-automatic.

by giving guidelines.

There are three common types of guidelines: 1) Pieces (usually points) belonging to the border (the algorithm completes the border); 2) Approximate contour of an object (the algorithm moves that boundary to fit with the object border), and 3) Seeds (the user labels some pixels belonging to the regions of interest, and the algorithm labels all of them completely) [188].

*Semi-automatic methods using guidelines*⁸ allow to segment what an application needs [188]. In particular, the segmentation can be improved if the user can give further clues that are effectively integrated in the segmentation process (this is an *interactive* segmentation) [232].

In this work, it was sought a method suitable for the images that are going to be segmented. The set of images includes both natural and artificial images but excluding medical ones. These images are in full color, i.e. excluding grey-level ones. Finally, these images contain one or more regions of interest. Each of these regions might be an object, part of an object, or even a set of objects. Sometimes these regions are somewhat arbitrary; partitioning, for example, an object in two regions.

Consequently, it is necessary an interactive, color, multi-region segmentation method.

Since there are many methods for color image segmentation [37, 22], choosing one requires experience and depends on the application [203]. In this work, it was chosen a color watershed method as it is based on mathematical morphology. As color watershed is based on grey-level watershed, the latter is described before.

The grey-level watershed transform was applied in the grey-level mosaic interpolation method. The color watershed (see Sec. C.2.1) segmented the original images.

⁸Notice that this is called “supervised” in the reference.

A.9.1 Interactive Segmentation

Interactive segmentation is an iterative process in which the user signals the ROIs and the system shows a proposed segmentation [77]. This interaction requires an adequate user interface [147]. Some ways to signal the ROI [173] are shown in Fig. A.15. The user indicates the ROI approximately, and the algorithm guesses it. The user can give additional cues to continue improving the segmentation [173]. It is also possible to use several signaling methods on the same image [230].

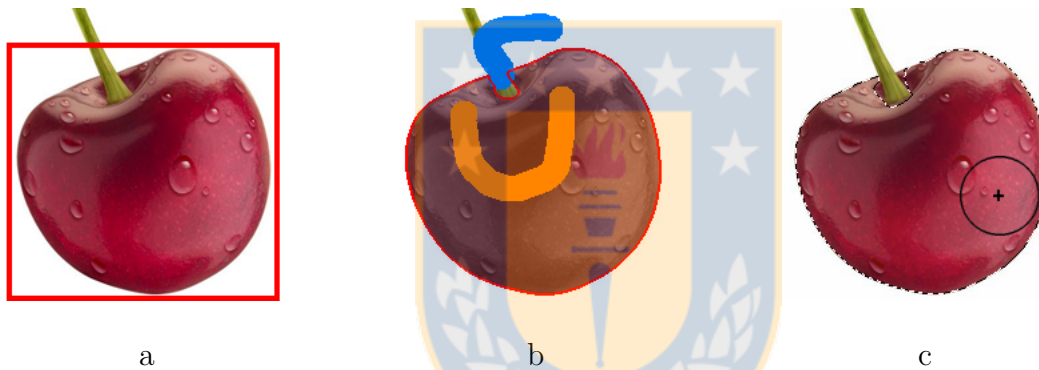


Figure A.15: Ways to signal the ROI: By using (a) a bounding rectangle, (b) scribbles, and (c) *Quick Selection Tool* in *Adobe Photoshop*.

As there are many interactive segmentation methods, it is necessary to choose one of them. It was tried to do this by researching recent evaluations that compare methods objectively against a dataset (see Appendix I). Unfortunately, the program corresponding to the best method was not available in the Internet.

As segmentation is not the main theme of this thesis, programming a method was discarded. Instead, it was searched an interactive color multi-label program in the Internet. At the end, only SegmentIt fulfilled the conditions and, at the same time, can be installed. Thus, it was chosen (see Sec. C.2.1).

A.10 Image Interpolation

In the study of phenomena where is impossible or too expensive to measure some variables as frequently as needed, the solution is to measure some points, and estimate the unknown values. The interpolation fits some curve that passes by the known points and uses this curve to compute the values of the unknown points [128].

The interpolation methods were first applied to astronomy and calendar computation in ancient times. They have been recently extended to signal and image processing [128].

Image interpolation applies interpolation to estimate the value of unknown pixels. Image interpolation has two application areas: temporal and spatial interpolation. Temporal interpolation computes the value of a fixed pixel by using images taken at different times; spatial interpolation computes the value of a pixel by using the values of different locations at the same time [81]. Temporal interpolation usually uses two images (never one) while spatial interpolation uses one or two images.

Applications such as printing and digital photography employ spatial interpolation to obtain a picture larger or smaller. Applications such as X-ray computed tomography and magnetic resonance imaging use spatial interpolation to improve the spatial resolution of a sequence of images.

An example of temporal interpolation is to add images in an old silent film to match the modern frame rate of 24 frames per second.

The image interpolation methods can be classified into two categories: adaptive and non-adaptive methods. *Adaptive methods* adjust their operations to image content, and *non-adaptive methods* process the whole image uniformly. Usually, adaptive methods apply a few interpolation methods accordingly to the contents of original images. The adaptive methods obtain better results but are more inefficient than non-adaptive methods [3].

The image interpolation methods can also be classified as grey-level (scene-based and intensity-based) and shape-based [25, 35]. This classification is used to explain the different methods used in this work as it distinguishes between traditional and newer methods.

A.10.1 Grey-level Image Interpolation

Maybe this is an inaccurate use of “grey-level” because these methods can be applied to binary, grey-level, and color images. Consequently, the usual meaning of grey-level is used in all this document except in this section.

Grey-level (image) interpolation methods compute the value of an interpolated pixel by using the values of the corresponding neighbor pixels in the original images [35, 215]. In general, they are fast and easy [215]. However, these methods suppose that a smooth curve can model image data [35], so they make artifacts (see Fig. A.16 [106]) and blurred edges [35, 215]. Some methods of this kind are nearest-neighbor, the simplest; linear, the most used; splines, and polynomial [25, 35].

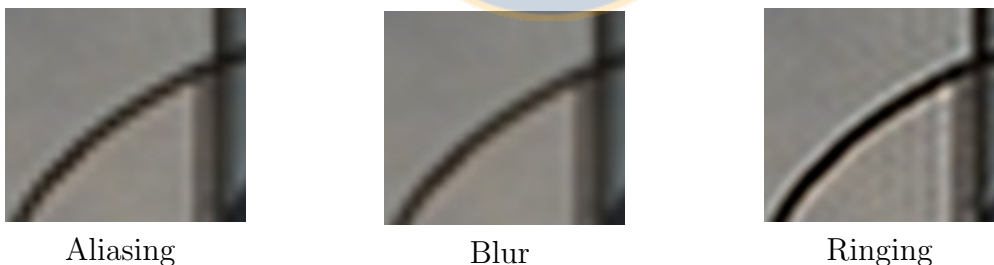


Figure A.16: Typical artifacts of linear interpolation methods

Nearest-neighbor Image Interpolation

In one-image interpolation, the nearest-neighbor (image) interpolation algorithm assigns to each interpolated pixel the value of the nearest pixel in the original image [145]. It produces an interpolated image that looks like as a checkerboard (see

Fig. A.17 [207]).

In two-image interpolation, it is equivalent to copying the nearest original image in the interpolated image.

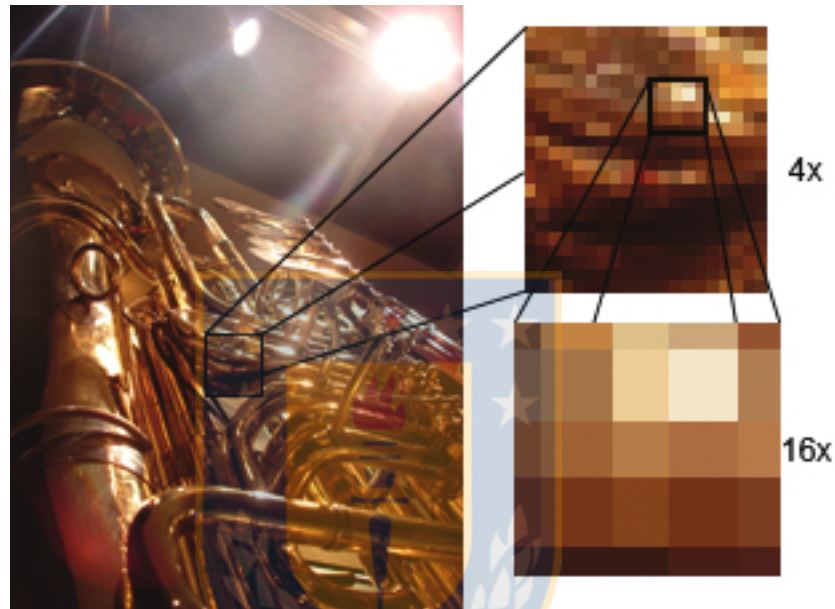


Figure A.17: Photograph section zoomed with nearest-neighbor interpolation method

Linear Image Interpolation

In one-image interpolation, linear interpolation it is known as bilinear interpolation (see Fig. A.18 [207]). Generally, it offers the best trade-off between quality and processing time [226]. In two-image interpolation, this method (also known as cross-dissolve) obtains new images by using weighted combinations of corresponding pixels [196]. It is efficient and simple, but it degrades significantly the interpolated image [71]. In particular, for color images, the interpolated image loses both color contrast and details [71]. For example, the linear interpolation of the images (a) and (b) gives (c) (see Fig. A.19).

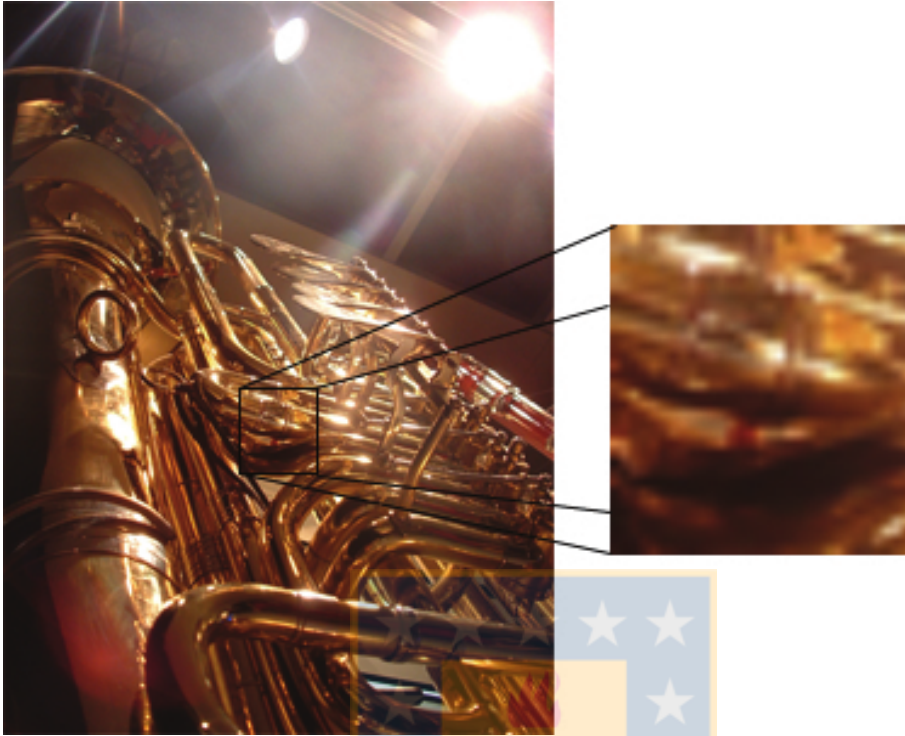


Figure A.18: Photograph section zoomed with bilinear interpolation method

Simultaneous Nearest-neighbor and Linear Image Interpolation

In image morphing, image warping and then color interpolation are performed [224]. There is an approach applied within image morphing that was applied in this work. It has been proposed segmenting the warped images before the interpolation so that no region cause self-occlusion. If a region is visible in both warped images, then these visible regions are interpolated linearly, otherwise the interpolated pixels are given only by the visible region. This approach reduces the problems produced by occlusions but causes artifacts, such as the region boundaries appear as seams in the final image [58].

In this work, objects that self-occlude were frequently segmented as it was suggested above, before applying *deformation*. Also, in the *overlapping* method, it is used something similar but instead of considering entire regions that appear in the original images, part of them are considered. In each part of a region visible in both original images, linear interpolation is applied, otherwise nearest-neighbor interpolation is applied by using the original image in which the part is visible.



(a)



(b)



(c)

Figure A.19: Linear interpolation (c) of the initial (a) and final (b) original images

A.10.2 Shape-based Interpolation

Shape-based interpolation methods interpolate by using shape features [25] instead of grey-level values. In binary images, the former methods are necessary because applying grey-level methods is almost the same as applying the nearest-neighbor method [128].

A.11 Image Quality Assessment

In this work, the interpolated images were compared against their reference images to know which interpolation method is better. This comparison required to choose a method for image quality assessment. Consequently, the latter methods were studied.

Initially, human beings assessed the images (subjective assessment) [174]. Later, images were assessed automatically —by using simple formulae or complex mathematical models of the *human visual system*— getting similar results as humans do (objective assessment) [182, 53].

This work used only objective assessment since the subjective assessment is more burdensome and expensive than the objective one [180]. As different observers could not agree about the quality of an image, multiple human comparisons are necessary to assess each image. In contrast, comparing images objectively only requires a formula implemented in a computer, and an operator that enters the images.

The objective (assessment) algorithms can be categorized by the presence or absence of an original (distortion-free) image: the full-reference algorithms have a reference; the no-reference algorithms have not, and the reduced-reference algorithms have only part of it [217]. The full-reference algorithms are considered the best option when there exists a reference image [31]. As, in this work, there are reference images, the goal is to choose one of the latter algorithms.

In the full-reference class, there are methods such as *mean squared error* (*MSE*) and *peak signal to noise ratio* (*PSNR*) [155]. The MSE is the simplest and most used full-reference method [217], but it is inaccurate to measure perceptual image quality [33]. The PSNR is another metric frequently used, but it has a small correlation with human perception [155]. As these metrics (see Appendix G) only use the image intensities, they are unsuitable to assess color images.

In a recent comparison between eleven full-reference metrics over all publicly available image databases, FSIM_C ranked first according to the rank-order correlations (see Sec. H.8) of Kendall, and Spearman, and the linear correlation of Pearson [233]. For this reason, this metric is used to compare images in this work.

A.11.1 FSIMc

The *human visual system* (HVS) considers salient low-level features, such as edges and zero-crossings to interpret a scene (Marr, 1980; Marr and Hildreth, 1980, and Morrone and Burr, 1998, cited in [234]). The comparison of these features can be used to create accurate indexes for image quality assessment [234].

The *Feature SIMilarity* (FSIM) index allows to compare images (grey-level images directly, and color images by using their luma) by considering their low-level features. As salient features coincide with the points where the Fourier waves had congruent phases, *phase congruency* (PC) is used as a primary feature in FSIM [234].

In addition, the HVS considers contrast information, but PC is contrast invariant. Therefore, FSIM includes the gradient magnitude (GM) calculated with the Scharr operator as a secondary feature [234]⁹.

FSIM first computes similarity measures for both PC and GM [234]. To compute

⁹It does not consider $\frac{1}{2h}$. Probably, it does not impact notoriously because it appears in Eq. A.13 where gradients are divided.

the phase congruency similarity, S_{PC} , it uses

$$S_{PC}(x) = \frac{2 PC_1(x) PC_2(x) + T_1}{PC_1^2(x) + PC_2^2(x) + T_1}, \quad (\text{A.12})$$

where PC_1 and PC_2 are the PC of the images, and T_1 is a positive constant to increase the stability of S_{PC} [234]. To compute the gradient similarity, S_G , it uses

$$S_G(x) = \frac{2 G_1(x) G_2(x) + T_2}{G_1^2(x) + G_2^2(x) + T_2}, \quad (\text{A.13})$$

where G_1 and G_2 are the GM of the images, and T_2 is a positive constant [234].

Then, the similarity, S_L , is [234]

$$S_L(x) = S_{PC}(x) S_G(x). \quad (\text{A.14})$$

Finally, a single score, FSIM, is computed as

$$FSIM = \frac{\sum S_L(x) PC_m(x)}{\sum PC_m(x)}, \quad (\text{A.15})$$

where $PC_m(x) = \max(PC_1(x), PC_2(x))$ [234].

Although FSIM performed well, it did not consider color information.

$FSIM_C$ improves FSIM by considering color information. $FSIM_C$ translates $R'G'B'$ images to YIQ to get their chrominance and luma. On one hand, to compute chrominance similarity, each channel (I, and Q) similarity is computed as

$$S_I(x) = \frac{2 I_1(x) I_2(x) + T_3}{I_1^2(x) + I_2^2(x) + T_3}, \quad (\text{A.16})$$

$$S_Q(x) = \frac{2 Q_1(x) Q_2(x) + T_4}{Q_1^2(x) + Q_2^2(x) + T_4}, \quad (\text{A.17})$$

where I_1 and Q_1 , I_2 and Q_2 are the chromatic channels of the images, and T_3 and T_4 are positive constant [234].

Then, chrominance similarity, S_C , is [234]

$$S_C(x) = S_I(x) S_Q(x). \quad (\text{A.18})$$

On the other hand, $S_L(x)$ and $PC_m(x)$ are computed on luma (Y') [234].

Finally, a single score, $FSIM_C$, is computed as

$$FSIM_C = \frac{\Sigma S_L(x) [S_C(x)]^\lambda PC_m(x)}{\Sigma PC_m(x)}, \quad (\text{A.19})$$

where $\lambda > 0$ is the parameter to adjust the importance of the chromatic components [234].





Mathematical Morphology

Mathematical morphology is a theory to analyze and process digital images. Specifically, mathematical morphology provides tools to analyze the shape and form of objects in images [189]. Although mathematical morphology started using set theory, later it was extended (generalized) to complete lattices [194, 67].

B.1 Mathematical Basis

A *partial order* is a binary relation \leq over a set P which is reflexive, anti-symmetric, and transitive. The set P with the relation \leq is called a *partially ordered set*, also known as *poset* [54].

If a partial order satisfies $\forall x, y \in P, x \leq y$ or $y \leq x$, then \leq is a *total order*. The set P with the relation \leq is called a *totally ordered set* [54].

Given a poset (P, \leq) , and $M, M \subseteq P$, an *infimum*, also known as *the greatest lower bound*, of M is an element $a, a \in P$ such that $a \leq z$ for every $z \in M$ but $x' \leq a$ for each $x' \in L$ with $x' \leq z$ for every $z \in M$ [193].

Given a poset (P, \leq) , and $M, M \subseteq P$, a *supremum*, also known as *the lowest*

upper bound, of M is an element a , $a \in P$ such that $a \geq z$ for every $z \in M$ but $x' \geq a$ for each $x' \in L$ with $x' \geq z$ for every $z \in M$ [193].

A *complete lattice* is a partially ordered set P such that every subset of P has an infimum and a supremum [15].

Multichannel images cannot be processed easily because lattices require an order, but there is no *universal* natural order for spaces with dimensions greater than 1 [140]. This problem limited to color images is treated in Sec. C.2.

A mathematical morphology operator (method) processes an image, called the *active image*, by using another image, called *structuring element*, as a probe or filter. Its result depends on the structuring element used [184].

In this work, the morphological operators are almost always applied to binary images. The only exception is the color median image generation.

Unless something different is clearly stated, this section treats about binary images and binary operations. Next sections explain succinctly grey-level and color operations.

B.2 Structuring Element

A structuring element is a small set used by the morphological operators [189]. Structuring elements have a crucial role in these operators: the different shapes and sizes of the former change the results of the latter [92].

It is necessary to define the origin of each structuring element [189]. Sometimes this pixel (point) is signaled with a circle [92], or a cross on it. Usually, the structuring element center is assumed as the origin.

In a morphological operation, the structuring element is superimposed so that its origin coincides with some point or pixel on the active image [189]. It is possible that the structuring element fits or not (misses) there. This determines the result of the operation. For example, a binary structuring element fits a binary image when there is a “1” on each point of the active image under the structuring element [92].

Some elementary discrete structuring elements for 4- and 8-connected grids are shown in Figs. B.1 and B.2, respectively [195]. They are approximations of a disk [189].

	1	
1	1	1
	1	

Figure B.1: 3x3 cross or diamond

1	1	1
1	1	1
1	1	1

Figure B.2: 3x3 square

Other examples of discrete structuring elements are shown in Figs. B.3 and B.4.

		1		
		1		
1	1	1	1	1
		1		
		1		

Figure B.3: 5 x 5 cross

		1		
	1	1	1	
1	1	1	1	1
	1	1	1	
		1		

Figure B.4: 5 x 5 diamond

If the previous discrete structuring elements are applied to two-dimensional images, they are called flat structuring elements because they have the same number of dimensions as the images under study [189].

The structuring elements with weights associated with its points are called grey-scale, non-flat or volumic structuring elements [189].

B.3 Binary Operators

In mathematical morphology, there are two basic morphological operators — dilation and erosion — that build complex ones [189], for instance, closing and opening [184].

Binary Dilation

In very simple terms, binary dilation (\oplus) of a set A by the structuring element B leaves a set Y whose elements are the elements of X superimposed with B (the center of B is placed on each point of X). Formally, $A \oplus B = \{z | (B^s)_z \cap A \neq \emptyset\}$, where B^s is the reflection, also called symmetric set, of B , $B^s = \{x \in E | -x \in B\}$ [65]. This definition is slightly different from those of Serra [179], and Soille [189]. However, they are equivalent when the structuring elements are symmetric. Another method to compute binary dilation is by using Minkowski addition. Let $A = (a_1, a_2, \dots, a_n)$ and $B = (b_1, b_2, \dots, b_m)$. Then $Y = A + B$, where $+$ represent the Minkowski sum [184]. See some examples below.

In Fig. B.5, there is only one pixel in the image at the left whose dilation creates a set with the same shape as the structuring element in the image at the right. The dilation was made with the 3x3 cross (Fig. B.1).

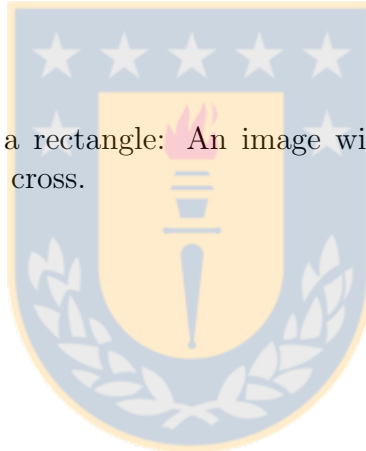


Figure B.5: Dilation of a pixel: An image with a pixel (a) and its dilation (b) with the 3x3 cross.

In Fig. B.6, there is a rectangle in the image at the left whose dilation creates a new set with a different shape in the image at the right. The dilation was made with the 3x3 cross (Fig. B.1).



Figure B.6: Dilation of a rectangle: An image with a small rectangle (a) and its dilation (b) with the 3x3 cross.



Binary Erosion

Binary erosion (\ominus) of a set A by the structuring element B leaves a set Y whose elements are the elements of X in which B can be fit. This is equivalent to $A \ominus B = \{z | (B)_z \subseteq A\}$ [65, 189]. Notice that Gonzalez [65], and Soille [189] agree on the binary erosion in spite of their disagree in the binary dilation. See some examples below.

There are some pixels in the left image in Fig. B.7. Its erosion with the 3x3 cross (Fig. B.1) erases almost all of them (see the image at the right in Fig. B.7). Notice that the element in the right bottom is preserved because the pixels out of the image are assumed as “1” in the erosion.



Figure B.7: Example of erosion: An image with some pixels (a) and its erosion (b) with the 3x3 cross.

In Fig. B.8, there is a rectangle at the left image whose erosion with the 3x3 cross (Fig. B.1) creates a small one (see the image at the right).



Figure B.8: Erosion of a rectangle: An image with a large rectangle (a) and its erosion (b) with the 3x3 cross

B.4 Ultimate Eroded Set

Applying binary erosion to a binary image successively erases all connected components sooner or later. The union of the disappearing connected components is the ultimate eroded set [189]. See some examples in Figs. B.9 and B.10. The 3x3 cross and the 3x3 square are shown in Figs. B.1 and B.2, respectively.

0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 1 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 0 0	0 0 0 1 0 0 1 0 0 0	0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 0	0 0 1 1 1 1 1 1 0 0	0 0 0 1 0 0 1 0 0 0
0 0 1 1 1 1 1 1 0 0	0 0 0 1 0 0 1 0 0 0	0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 1 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0
Original set	Eroded set	Ultimate eroded set

Figure B.9: Computing ultimate eroded set with the 3x3 cross

0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 1 0 0 0	0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 0 0	0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 0	0 0 0 1 1 1 1 0 0 0
0 0 1 1 1 1 1 1 0 0	0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 1 0 0 0	0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0
Original set	Ultimate eroded set

Figure B.10: Computing ultimate eroded set with the 3x3 square

B.5 Hit and Miss Transform

The hit and miss transform allows to detect a certain configuration of pixels in a position of an image. It is erroneously or inappropriately written as hit or miss transform [46, 117, 189]¹.

This transform uses a *composite structuring element* formed by two disjoint structuring elements which have the same origin. One of them has to fit the object under study (C), and the other has to miss it (D). For this reason, hit and miss is a proper name for this transform [189]. It is possible to detect foreground or background pixels depending on who owns the origin. A composite structuring element that detects upper left corners is shown below in Fig. B.11.

¹The hit and miss transform name originates neither from the adjective hit-or-miss (US) nor from the adjective hit-and-miss (UK) [159].

	1	1
	1	

(C)

1	1	1
1		
1		

(D)

Figure B.11: Composite structuring element for detecting upper left corners: (C) For the foreground and (D) for the background.

This transform is applied to images by using the formula

$$A \odot B = (A \ominus C) \cap (A^c \ominus D),$$

where \ominus is the erosion and A^c is the set complement of A [49].

B.6 Thinning

Thinning deletes pixels of an object without holes until minimally connected lines remain halfway its borders, and deletes pixels of an object with holes until minimally connected rings remain halfway between the external border of each hole and the nearest external border belonging to the object or to another hole [157]².

Thinning removes the border pixels that match the configuration given by a composite structuring element. Thinning \otimes is computed by using

$$X \otimes T = X \setminus (X \odot T),$$

where X is a set, T is a composite structuring element, \setminus is set difference, and \odot is the hit and miss transform [179].

Although this definition seems simple, there are many thinning algorithms [108].

²This definition extends that of Pratt to consider adjacent holes.

It is possible to apply thinning successively to a set what is called *iterative thinning*. There are two ways of applying iterative thinning. First, examining the pixels one after another, what is called *sequential thinning*. The result of this thinning depends on the order in which the pixels are processed. Second, examining all the pixels simultaneously, what is called *parallel thinning*. The result of this thinning in an iteration, only depends on the result of the previous iteration [108].

B.7 Shrinking

Shrinking deletes pixels of an object without holes until a pixel remains at or near the geometric center of the object, and deletes pixels of an object with holes until connected rings remain halfway between the external border of each hole and the nearest external border belonging to the object or to another hole³. Particularly, a 3x3 pixel object shrinks to a pixel in its center⁴; and a 2x2 pixel object shrinks —by definition— to a pixel in the lower right corner [157].

According to the definitions of thinning and shrinking, there would be no difference when an object has holes. However, they might obtain different results since they use distinct algorithms. For example, the results of applying thinning and shrinking on the image (a) can be seen in the images (b) and (c) in Fig. B.12.

B.8 Skeletonization

Skeletonization, also known as *Skeletonizing*, reduces an object to a set of lines that are called skeleton. The extension of skeletons from Euclidean sets to discrete sets is complex what leads to skeletons that have different properties [189].

³This definition extends that of Pratt to consider adjacent holes.

⁴Pratt uses *center of mass*, but in a uniform density object, the center of mass coincides with the geometric center or centroid.



(a) A circle with holes



(b) Applying shrinking



(c) Applying thinning

Figure B.12: Comparing shrinking (b) with thinning (c) on the image (a)

Euclidean Skeletons

There are several formal definitions for Euclidean skeletons. These definitions give similar thin lines and preserve the homotopy of the original set. Among them, maximal disks, grass-fire or wavefront propagation, distance function, minimal paths, and openings [189]. The first two definitions are explained here.

A *maximal disk*, also known as *maximum disk*, is one of the disks contained in a shape so that no another disk within the shape contains it [98, 179].

A *skeleton* is the set of centers of the maximal disks (balls) of a set [179]. Formally,

$$S(X) = \bigcup_{\rho > 0} \bigcap_{\mu > 0} [(X \ominus \rho B) \setminus ((X \ominus \rho B) \circ \mu \bar{B})], \quad (\text{B.1})$$

where ρB is an open ball of radius ρ , and \bar{B} is the closure of B (Lantuéjoul, 1977, cited in [179]).

For example, a set of two overlapped disks has a skeleton formed by a line between the centers of both disks (see Fig. B.13).

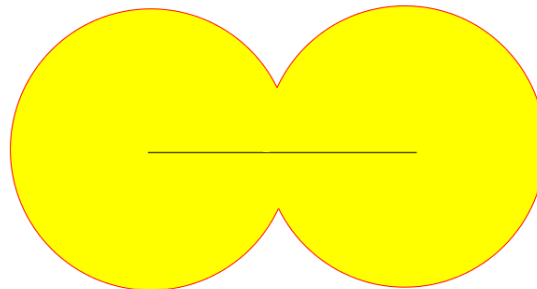


Figure B.13: Overlapped disks and its skeleton

Medial axis transformation reduces an object to a set of lines that are called medial axis.

Medial axis is the set of centers of disks within a set X that overlap the borders of X at various points [179]. In simpler words, the medial axis of an object P is the set of points inside P having the same minimal distance to various borders of P [149]. A method based on wave fronts was proposed to compute it [23]. Suppose a simultaneous excitation on all the borders of an object. This excitation creates waves that spread uniformly in all directions but without flowing through each other. The medial axis is the set of points where various waves meet. Notice that waves that meet outside the object are discarded. In addition, it was suggested visualizing the contours (waves) as the front of a grass fire [23]. In this case, it is supposed that there exist grass on an object. Then, the boundaries of the object are set on fire. The fire spreads inwards from these points. Montanari (1968, cited in [98]) suggests that each fire front propagates with a constant velocity along its normal direction. The medial axis is the set of points where various fire fronts meet [38].

The medial axis has been called [23] or considered [98, 189] a skeleton. Precisely, the medial axis is a subset of the skeleton, but the difference is minimal. However, distinct morphological methods are used to compute them [179]. In this work, these concepts are considered equivalent; they are called skeleton.

Discrete Skeletons

The extension of skeletons to discrete sets is not direct [189, 146] due to the fact that most concepts applied to Euclidean skeletons do not have a discrete equivalent [189]. For example, a line cannot be infinitely thin; a disk cannot be represented precisely, and a centered skeleton is placed approximately (notorious in thin shapes) [189].

Considering the constraints to apply skeletonization to discrete sets, these desirable properties of discrete skeletons have been proposed [146]:

1. Topology preservation: the Betti numbers⁵ of a set and its skeleton are the same.
2. One-pixel-thickness: the skeleton should be as thin as possible⁶.
3. Medial position: the skeleton is midway the borders of the image.
4. Rotation invariance: if an image is rotated, its skeleton is the same but rotated.
5. Noise immunity: the skeleton should not change when there is noise.
6. Reconstructibility: the skeleton should allow to get the original set [189].

Unfortunately, it is impossible to fulfill all of them simultaneously. Therefore, each method complies some of them [146].

There exist many types of skeletonizing algorithms, each of them with its own characteristics [189, 146]. Some of them are maximal disks, homotopic sequential thinnings, order independent homotopic thinning, discrete distance function, skeleton by influence zones (SKIZ), and openings [189]. Some notes about some of these methods are shown below.

Maximal disks is a discretization of Eq. B.1. This is $S(X) = \bigcup_{n=0}^{\infty} S_n$, where $S_n(X) = (X \ominus nB) \setminus ((X \ominus nB) \circ B)$, and B is the elementary ball [19, 49, 222]. A skeleton computed accordingly to this definition might be non-connected. Consequently, this skeleton does not preserve homotopy [211]. In addition, it might have lines wider than one pixel [222].

Homotopic sequential thinning applies sequential thinning with *homotopic structuring elements* (they are structuring elements that do not break the connectivity of a set [19]). This thinning produces a medial axis. These points and their minimal

⁵For two-dimensional images, the Betti numbers are the number of connected components and the number of holes.

⁶To get this, it was suggested that the foreground would be considered 8-connected and the background 4-connected [108].

distance to the borders allow to reconstruct the original set [189]. This thinning conserves topology and to some extent the shape of the object, but it is time-consuming, noise-sensitive, and gives very rough skeletons [146].

The skeleton by influence zones (SKIZ) is a subset of the medial axis [179].

A skeleton is shown in Fig. B.14 (b). Notice that this skeleton seems strange; it was expected a symmetric skeleton since the set is symmetrical.

A skeleton can be seen in Fig. B.14 (c). This is a parallel thinning that produces thin medial curves [73]. Notice that this skeleton is symmetrical.

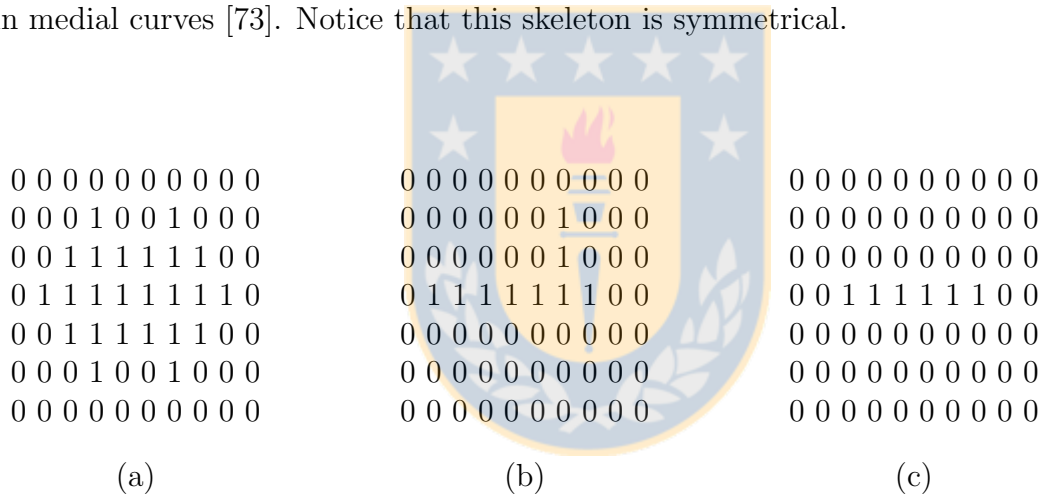


Figure B.14: Different skeletons of a set: The set (a), and the skeletons computed in Matlab with bwmorph operations “skel” (b), and “thin” (c).

B.9 Pruning

Pruning is a process that cuts spurious branches that appear in skeletonizing operations [150]. These branches can be located by finding points with only one neighbor [184]. This can be done by using the hit and miss operator [119].

B.10 MSP

As a skeleton is thin, pruning it until stability finishes with “a single point or ringlike structures” [49]. Particularly, pruning the skeleton of a filled region—a region whose holes have been filled—finishes with a point. This point is called *MSP* [210].

In addition, it has been computed by shrinking [206]. This method is better because the position of this point is at or near the center of mass [157]. Notice that the center of mass might lie outside the object what never happens with the MSPs computed with the previous method.

B.11 Interpolation of Sets

Several morphological interpolation methods processed sets. For example, the *median set* (see Sec. B.11.1), the *sequence of interpolations through median sets* of Beucher (see Sec. B.11.2), the *Interpolation function* of Meyer (see Sec. B.11.3), and the *Interpolations based on Hausdorff distance* of Serra (see Sec. B.11.4).

The methods of Meyer and Beucher cannot interpolate when the sets do not intersect; the methods of Serra have problems to interpolate distant sets⁷. In general, interpolating distant sets is either impossible or almost unrealistic [87].

B.11.1 Median Set

Serra [178] defined the median set M of X and Y , being $X \subseteq Y$, as

$$M = \bigcup_{\lambda \geq 0} \{(X \oplus \lambda B) \cap (Y \ominus \lambda B)\}, \quad (\text{B.2})$$

where B is the elementary structuring element [85]. An example of a step of the application of this formula is shown in Fig. B.15. Here, the intersection of $X \oplus \lambda B$

⁷Meyer published before Beucher and Serra, but they had written separate technical reports about the same theme of their articles in 1994 [178].

and $Y \ominus \lambda B$, that is a part of the median set M , equals $X \oplus \lambda B$.

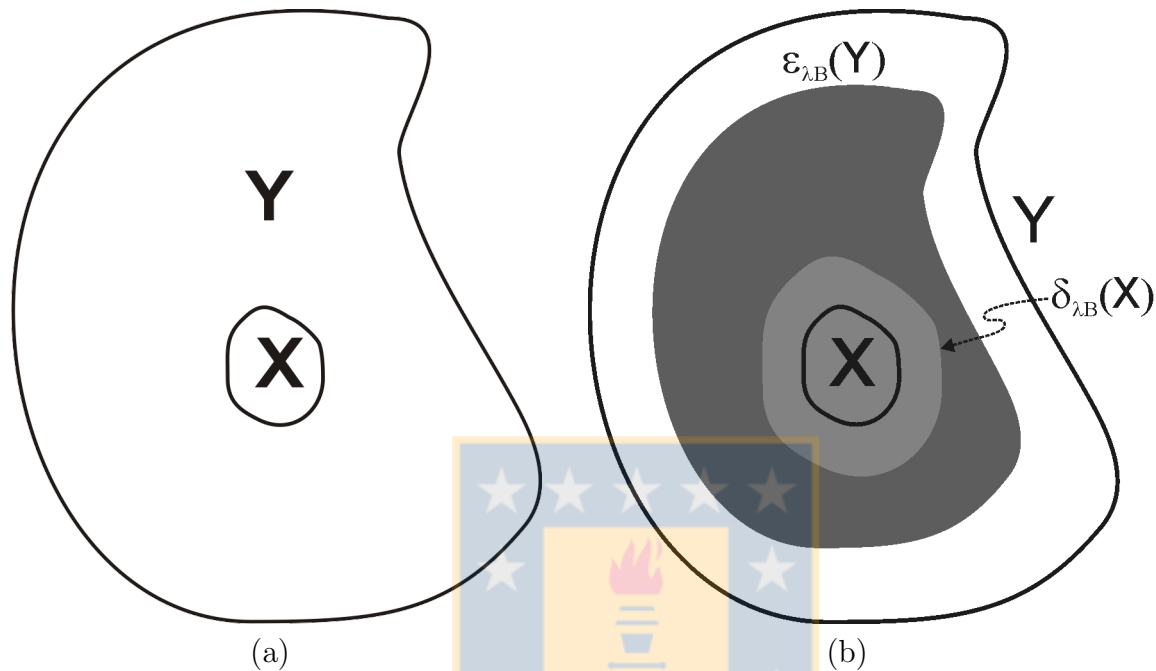


Figure B.15: Computing median set: (a) X and Y sets (taken from [207]). (b) A step of median set computation (modified from [207]).

Observe that the median set of nested sets is calculated by dilating X and eroding Y . While Beucher [20] proposed to compute median sets by using only dilation operations.

The median set is halfway between the two original sets. If these sets are shapes, the median set obtains a shape midway the original shapes. The median set is an interpolation of these sets [178].

Median set can easily be extended to intersecting sets [85]. Suppose two sets P and Q , $P \cap Q \neq \phi$. Let $X = P \cap Q$, and $Y = P \cup Q$. Then it is possible to compute the median set by using Eq. B.2. An example of a median set can be seen in Fig. B.16.

An algorithm to compute median sets is described below [85].

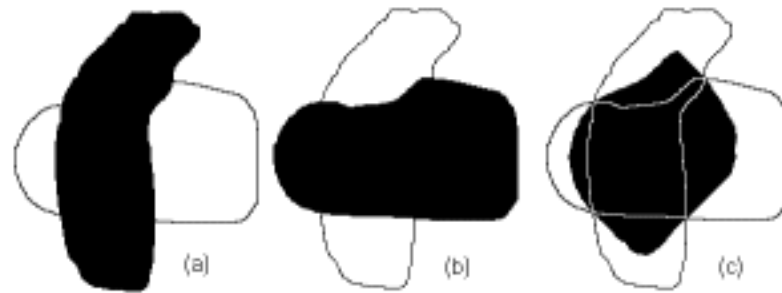


Figure B.16: Median set from two input sets: (a) an input set, (b) another input set and (c) its median set.

Initially, three auxiliary sets are defined

$$Z_0 = M_0 = X \cap Y$$

$$W_0 = X \cup Y$$

Then, new values are computed iteratively until idempotency by using

$$Z_i = Z_{i-1} \oplus B$$

$$W_i = W_{i-1} \ominus B$$

$$M_i = (Z_i \cap W_i) \cup M_{i-1},$$

where B is the elementary structuring element.

Consequently, median set is

$$M(X, Y) = M_\infty = M_i, \quad (\text{B.3})$$

where $i : M_{i+1} = M_i$.

B.11.2 Sequence of Interpolations through Median Sets

Beucher [20] used median sets (see Sec. B.11.1) to interpolate sets. The idea is to create a sequence of deformations from X to Y . Be $K_0 = X$; $K_n = Y$ with n a power of 2, and M the median set between two sets. Then

$$\begin{aligned} K_{n/2} &= M(K_0, K_n); \\ K_{n/4} &= M(K_0, K_{n/2}); \\ K_{3n/4} &= M(K_{n/2}, K_n); \\ K_{n/8} &= M(K_0, K_{n/4}); \\ &\dots \end{aligned}$$

B.11.3 Interpolation Function

Meyer [130] proposes using geodesic distances (see Sec. A.7.1) to build an interpolation function between a set U and a set V containing it. For any point x belonging to V/U passes a shortest path between U and V^C . Be d_1 the geodesic distance between x and V^C . This distance can be computed eroding V in U^C with the geodesic erosion ($\epsilon_U(V) = \epsilon(V) \cup U$). Be d_2 the geodesic distance between x and U . This distance can be computed dilating U in V with the geodesic dilation ($\delta_U(V) = \delta(U) \cap V$). The shortest path has a distance equal to $d_1(x) + d_2(x)$. Hence, it is possible to define an interpolation function as

$$Int_U^V = \begin{cases} 1 & U \\ \frac{d_1}{d_1+d_2} & V/U \\ -\infty & V^C \end{cases}$$

The interpolated sets between U and V are obtained by thresholding between 0 and 1 by using

$$Int_U^V(\alpha) = \left\{ x \mid Int_U^V \geq \alpha \right\}$$

If the value of α is 0, it gives V ; if the value is 1, it gives U .

In the same paper, Meyer extended it to any two intersecting sets X and Y . To do this, the set U is constructed as $X \cap Y$. The set X becomes Y by, at the same

time, shrinking X until it becomes U and expanding U to become Y .

Thus, the interpolated set T is

$$T = \underset{X}{\overset{Y}{\text{Interpolation}}}(\alpha) = \underset{U}{\overset{X}{\text{Int}}}(\alpha) \cup \underset{U}{\overset{Y}{\text{Int}}}(1 - \alpha),$$

where $\alpha \in [0,1]$.

B.11.4 Interpolations based on Hausdorff Distance

Serra [178] used Hausdorff distance (see Sec. A.7.2) to create several types of interpolations, such as the *first Hausdorff geodesic* and the *second Hausdorff geodesic*. The first method is explained below.

First Hausdorff Geodesic Interpolation

The *first Hausdorff geodesic interpolation* considers that a pair X, Y at Hausdorff distance ρ apart, admits this geodesic [178]:

$$\{Z_\alpha = \delta_{\alpha\rho}(X) \cap \delta_{(1-\alpha)\rho}(Y), \alpha \in [0, 1]\}$$

When $\alpha = 0$, this interpolation gets X ; when $\alpha = 1$, this interpolation gets Y ; when α takes an intermediate value and the sets X and Y are disjoint, this interpolation is greater than both X and Y (a swelling).

An example is shown in Fig. B.17. Consider the sets X and Y , and the structuring element shown in (a), (b), and (c), respectively. Multiples dilations —each area enclosed by a red line— of the set X and Y are shown in (d) and (e), respectively. Note that $\rho = 11$. Thus, to compute all the interpolations for this example, $\alpha = n/11$, with $n=0, 1, \dots, 11$. Finally, the interpolation with *alpha* = 4/11 is shown in (f).

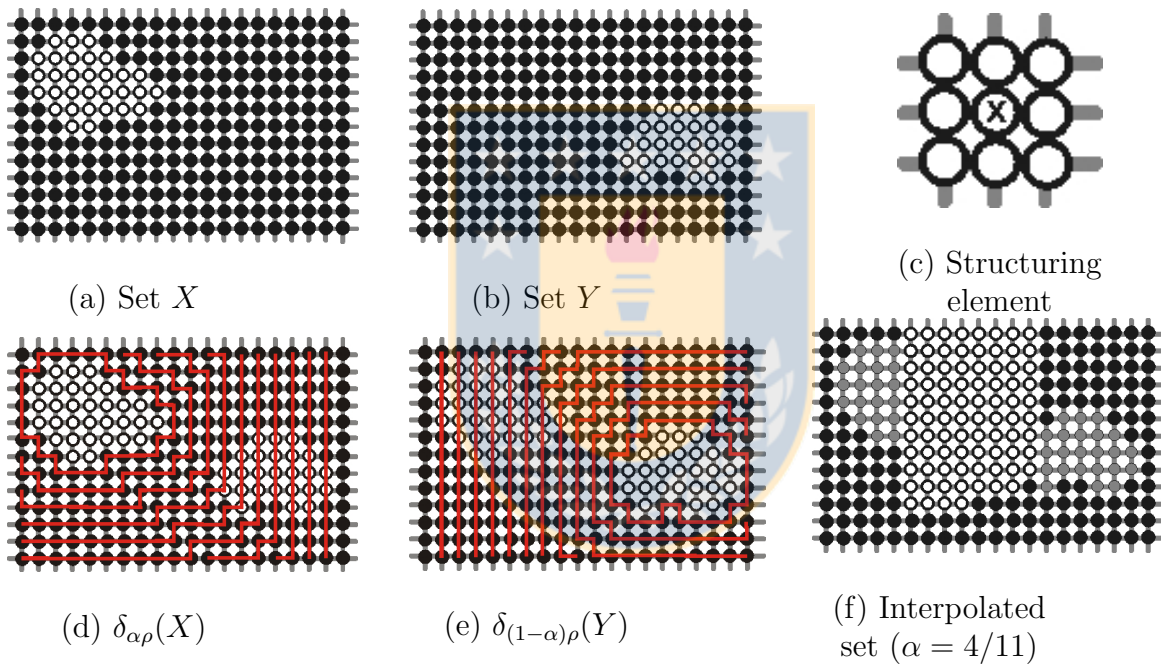


Figure B.17: Example of first Hausdorff geodesic interpolation

Mathematical Morphology for Images

C.1 Grey-level Mathematical Morphology

This section should refer to mathematical morphology applied to grey-level images. However, in this work, grey-level operators were not used. However, the grey-level watershed transform has been extended to color images.

C.1.1 Grey-level Watershed Transform

There are many concepts of physical geography used in mathematical morphology. One of them is watershed. Unfortunately, watershed has two usages in mathematical morphology: as drainage divide [103, 189], and as catchment or drainage basin [65, 157, 179]. To avoid confusions, watershed is synonymous with drainage divide in this document.

A grey-level image can be seen as a topographical map where high values represent high altitudes, and low values represent low altitudes [65]. The watershed transform obtains its drainage divide (watershed) [103].

There are two intuitive approaches to explain the watershed transform: One based

on raining and the other, on flooding. On one hand, if it rains, each drop after contacting the surface slides by the steepest descent to the lowest point in the region (regional minimum). However, in some points, a drop might slide to several regional minima. These points are the watershed [189]. On the other hand, first suppose that holes are pierced in each regional minimum, then the water table level rises, but water can only enter by the holes. A dam is constructed where the water fronts originating from several holes meet. When all the surface is underwater, the resulting set of dams is the watershed [189, 103].

The watershed transform can be implemented with the flooding approach and cannot be implemented with the raining approach [189].

The watershed transform (classical or regular watershed [49]) can be applied to the original image, and to its gradient [103]. It produces over-segmentation on both cases [166, 65, 184]. An example of over-segmentation is shown in Fig. C.1 .

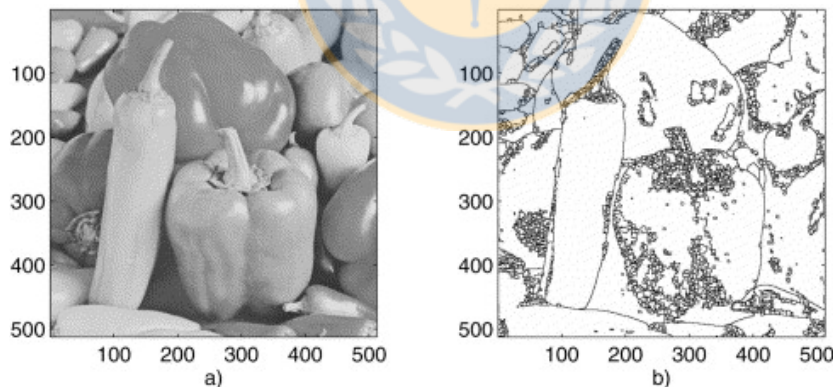


Figure C.1: Watershed of an image: (a) An image and (b) its watershed (Alina N. Moga and M. Gabbouj, taken from [157])

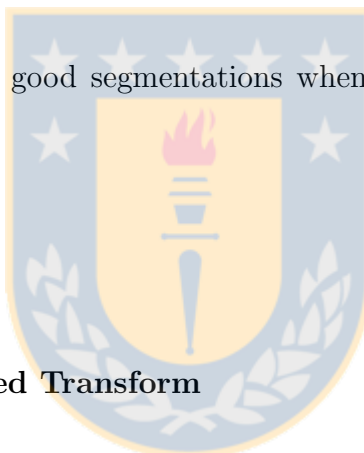
Several approaches have been used to solve this problem, among them: watershed from markers and hierarchical watershed [166]. They are useful for interactive image segmentation, and they have been used conjointly in *SegmentIt* [101].

Watershed from Markers

The flooding analogy helps to explain this approach easily. As each hole creates a region, the idea is to make holes only in the regions of interest (objects, for example). The user puts a marker in each region of interest, and the watershed transform drill holes only in these markers [49, 101, 116].

A marker could be a connected component [65] or a set of disconnected points [116]. Markers could be defined by the user or computed automatically [189]. Many markers might correspond to the same segment [189].

This method obtains good segmentations when it is known where the markers should be placed [49].



Hierarchical Watershed Transform

Hierarchical or multiscale watershed transform creates a hierarchy of water basins. It can be understood as an improvement of the watershed from markers approach. Starting from the case in which each minimal region has a marker, the number of markers is reduced one-by-one until only one marker stays, by taking first the markers belonging to the upstream basins (in Dougherty terms, those with lower dynamics) [49]. A “upstream basin” has its regional minima upper than another basin (downstream basin). The basins constructed with this procedure form a hierarchy of basins: the upstream basin is the child and the downstream basin is the father.

These hierarchies help to solve the over-segmentation problem [171]. To do that, the adequate basins are selected from the hierarchy.

C.2 Color Mathematical Morphology

Mathematical morphology cannot be applied to colors directly. As mathematical morphology requires an order relation, there is a problem because colors do not have a universal natural order. A solution is to use a *sub*-(less than total) *ordering*, that is to say, an incomplete ordering relation. There are four classes of sub-ordering: marginal, reduced, partial, and conditional [16, 10, 231].

In the marginal ordering, each component is ordered independently of the other components. Thus, morphological operators that use this ordering produce *component-wise operators*. For example, the color dilation of

$$f(x, y) = [f_R(x, y), f_G(x, y), f_B(x, y)]$$

by the structuring element

$$h(x, y) = [h_R(x, y), h_G(x, y), h_B(x, y)]^T$$

in RGB color space is defined as

$$(f \oplus_c h)(x, y) = [(f_R \oplus h_R)(x, y), (f_G \oplus h_G)(x, y), (f_B \oplus h_B)(x, y)]^T,$$

where the symbol \oplus_c represents component-wise dilation and \oplus represents grey-scale dilation [40].

In the reduced ordering, each vector maps to a single scalar value; these values order the vectors (Barnett, 1976, cited in [40]). Usually, the function that maps the vectors computes the distance to a reference vector [140].

In the partial ordering¹, pre-orders partition the vectors in groups of equivalence (Titterton, 1978, cited in [9]). In simpler words, vectors are partitioned into smaller groups [59]. Although the groups are ordered between them, there is no order within the groups [154].

¹This is distinct to the partial order relation [140] (see partial order in Sec. B.1).

In the conditional ordering, vectors are ordered by using a hierarchical order of its components [114]. This ordering is usually called the lexicographical order. Different priorities can be defined between the components. If all of them are considered, this is a total order. In addition, the infimum and supremum are members of the initial set of vectors [140]. However, it does not consider the vectorial nature of the color input [59].

Although all these families of orders have been used, only a few orders have emerged as suitable. All of them change a space to a new one in which it is possible to order the vectors [140].

However, the ambiguity of ordering vectors and the subjectivity of colors have prevented the general acceptance of any color morphological operator [10].

C.2.1 Color Watershed

Color segmentation methods are essentially grey-level segmentation methods applied to different color spaces [37]. For example, the watershed transform processes only grey-level images [34, 47]; many techniques have extended it to color images [47]. Obviously, these techniques might produce different watersheds [47].

There is an interactive program that uses color watershed: SegmentIt. This program allows both watershed from markers and hierarchical watershed. Furthermore, it is possible to change from one approach to the other. In the hierarchical watershed, the user chooses which partition should be split to obtain each region of interest. It is also possible to merge regions. This program uses a weighted gradient to segment. A weighted gradient is the ponderation of the gradients defined in each band of the HSB (HSV) color model [101].

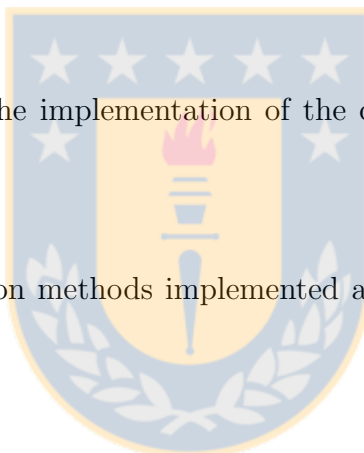




Implementing Existent Interpolation Methods for Color Images

This chapter describes the implementation of the color interpolation methods that were used in this work.

The color interpolation methods implemented are the linear, and the median of Iwanowski and Serra.



D.1 Implementing Linear Interpolation for Color Images

The two-image method explained in Sec. A.10.1 was implemented [196]. The linear interpolation method requires a perceptually uniform color model to get better results. Here, CIELAB was chosen. So, the RGB images are transformed into CIELAB images, then each component is interpolated linearly, and finally, the result is converted into RGB. The implementation in Matlab can be seen in Fig. D.1.

```

function sRGB3 = Linear_interpolation(sRGB1, sRGB2)

C = makecform('srgb2lab');
lab1 = applycform(sRGB1, C);
lab2 = applycform(sRGB2, C);

lab3= lab1 / 2 + lab2 / 2;

C = makecform('lab2srgb');
sRGB3 = applycform(lab3, C);

```

Figure D.1: Linear color interpolation program (written in Matlab)

D.2 Implementing Color Median of Images

This is an adaptation of the *median image generation method for color images* of Iwanowski and Serra explained in Sec. 2.4 [85]. In this adaptation (see Fig. D.2), pixels were compared using lightness.

As in CIELAB, the first component, L^* , is lightness, the RGB images are converted into CIELAB ones within `calcularInfimoLuminosidad`, `calcularSupremoLuminosidad`, `dilatarColorLuminosidad`, and `erosionarColorLuminosidad`.

The infimum is the image that results from a point-to-point operation that takes the pixel with lower lightness from the input images. It is computed in `calcularInfimoLuminosidad` (see Fig. D.3).

The supremum is the image that results from a point-to-point operation that takes the pixel with higher lightness from the input images. It is computed in `calcularSupremoLuminosidad`.

In the dilation, computed in `dilatarColorLuminosidad`, the structuring element is superimposed on each pixel of the input image; the pixel under the structuring element with the highest lightness is put under the center of the structuring element in the output image.

```

function M = computingColorMedianImage (X,Y)

if sum(sum(sum(X & Y))) ~ = 0
    SE = strel('disk',1,8);
    Z = calcularInfimoLuminosidad (X, Y);
    W = calcularSupremoLuminosidad (X, Y);
    Mant = W;
    M = Z;
    while ~isequal(Mant, M)
        Mant = repmat(M,1);
        Z = dilatarColorLuminosidad (Z, SE);
        W = erosionarColorLuminosidad (W, SE);
        M = calcularSupremoLuminosidad (calcularInfimoLuminosidad (Z, W), M);
    end
else
    M = X;
    display('Warning, the set intersection is empty!');
end

```

Figure D.2: Color median image generation program (written in Matlab)

```

function M = calcularInfimoLuminosidad (sRGB1, sRGB2)

C = makecform('srgb2lab ');
lab1 = applycform(sRGB1, C);
lab2 = applycform(sRGB2, C);
[filas, columnas, dummy] = size(sRGB1);

for i = 1:filas
    for j = 1:columnas
        if (lab1(i,j,1) <= lab2(i,j,1))
            M(i,j,1) = sRGB1(i,j,1); M(i,j,2) = sRGB1(i,j,2); M(i,j,3) = sRGB1(i,j,3);
        else
            M(i,j,1) = sRGB2(i,j,1); M(i,j,2) = sRGB2(i,j,2); M(i,j,3) = sRGB2(i,j,3);
        end
    end
end
end

```

Figure D.3: calcularInfimoLuminosidad program (written in Matlab)

In the erosion, computed in `erosionarColorLuminosidad`, the structuring element is superimposed on each pixel of the input image; the pixel under the structuring element with the lowest lightness is put under the center of the structuring element in the output image.



Prewitt Operator

Extending the kernels in Figs. A.10 and A.11 to a 3x3 matrix leads to the Prewitt kernel [160].

Two of the eight Prewitt masks are shown in Eqs. E.1 and E.2 [18]. Notice that G_x goes rightwards, and G_y , upwards.

$$G_x = \frac{1}{6} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad (\text{E.1})$$

$$G_y = \frac{1}{6} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad (\text{E.2})$$

Many authors use approximate Prewitt masks as they are not interested in the exact values of the derivatives. They are interested in the relative differences; for example, to compute the gradient orientation, or to detect a border. Some authors do not consider $\frac{1}{2h}$ in Eq. A.2. Two of the eight approximate Prewitt masks are shown in Eqs. E.3 and E.4 [234]. Notice that G_x goes rightwards, and G_y , upwards. Other authors consider neither $\frac{1}{2h}$ nor $\frac{1}{w+2}$ in Eq. A.2. Thus, two of the eight approximate

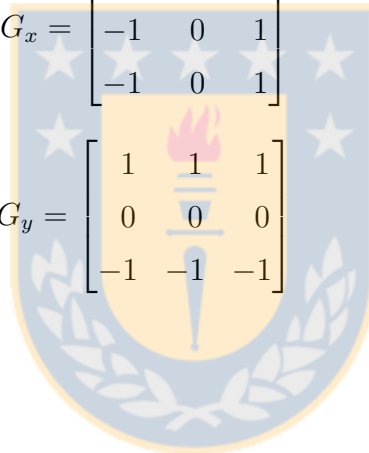
Prewitt masks are shown in Eqs. E.5 and E.6 [2].

$$G_x = \frac{1}{3} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad (\text{E.3})$$

$$G_y = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad (\text{E.4})$$

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad (\text{E.5})$$

$$G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad (\text{E.6})$$



Sobel Operator

The Sobel operator gives more importance to the difference across the central pixel [82].

This operator is often better than the Prewitt operator [26] as it suppresses noise better [50, 92].

Two Sobel kernels [18] are shown below. Notice that G_x goes rightwards, and G_y , upwards.

$$G_x = \frac{1}{8} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad G_y = \frac{1}{8} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

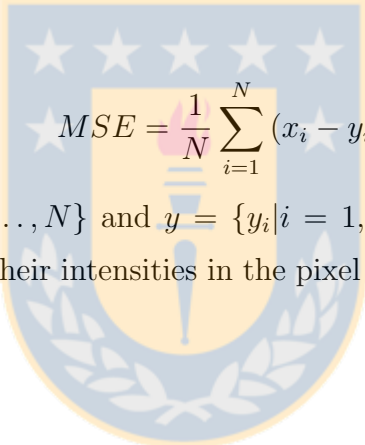
For the same reason given for the Prewitt operator, approximate Sobel masks are frequently used. For example, two commonly used kernels are the matrices shown above but without the fractions [151].



Mean Squared Error and Peak Signal to Noise Ratio

G.1 Mean Squared Error

MSE is computed as


$$MSE = \frac{1}{N} \sum_{i=1}^N (x_i - y_i)^2,$$

where $x = \{x_i | i = 1, 2, \dots, N\}$ and $y = \{y_i | i = 1, 2, \dots, N\}$ are two images to be compared, x_i and y_i are their intensities in the pixel i , and N is the number of pixels of these images [216].

G.2 Peak Signal to Noise Ratio

PSNR is computed as

$$PSNR = 10 \log_{10} \frac{L^2}{MSE},$$

where MSE is the *mean squared error* and L is the dynamic range of intensities (for 8 bits/pixel, $L = 2^8 - 1 = 255$) [216].



This section explain some basic and advanced concepts about statistics.

H.1 p-value

The p – value is the probability that the results were obtained by chance. Formally, the p – value is the probability that the test statistic obtains values as or more extreme than the observed test statistic by assuming that H_0 is true [28].

The smaller p – value, the stronger evidence against the null hypothesis [28].

H.2 One- and Two-tailed Tests

The way of computing the p – value depends on the alternate hypotheses and the type of test [28]. For example,

This is the null hypothesis:

$$H_0: m = 0.$$

These are alternative hypotheses:

$$H_1: m \neq 0.$$

$$H_1: m > 0.$$

$$H_1: m < 0.$$

If the alternative hypotheses is $H_1: m \neq 0$, the p – *value* is the probability of getting a test statistic as high or higher than the observed test statistic or as low or lower than the observed test statistic. This is a *two-tailed test* [28].

If $H_1: m > 0$, the p – *value* is the probability of getting a test statistic as high or higher than the observed test statistic. This is a *right-tailed test* [28].

If $H_1: m < 0$, the p – *value* is the probability of getting a test statistic as low or lower than the observed test statistic. This is a *left-tailed test* [28].

The right- and left-tailed tests are *one-tailed tests* [218].

H.3 Significance Level

The significance level is the maximum probability of arriving to a “wrong” conclusion that someone tolerates (is willing to take). For example, a significance level of 1% means that if you collect 1000 samples and perform a statistical test on each of them, you will make the “wrong” conclusion ten times (1% of 1000 is 10). Here, *wrong* means that the alternative hypotheses is accepted (it is believed to be true) when it is false. This is a *Type I error* [175].

The significance level, α – *level*, usually is set to 0.10, 0.05, and 0.01. They are called moderately significant, significant, and highly significant, respectively [175].

When a statistical test is performed, if the p -value is less than α -level, the result is statistically significant [175]. Consequently, the null hypothesis can be rejected [168]

The significance level must be set before the experiments [221].

H.4 Statistical Power

The statistical power of a test is the probability of rejecting the null hypothesis when that hypothesis is in fact wrong [138].

The error of not rejecting a false null hypothesis is called a *type II error*. The probability of this type of error is *beta*. Therefore, the power is 1 minus beta [14].

H.5 Skewness

Skewness is defined as the opposite of symmetry; when there is skewness, the mean, median and mode values are different [90]. The skew can be negative or positive (see Fig. H.1). It is positive when more than half of the area under the curve is at the right of the mode, and it is negative when more than half of the area below the curve is at the left of the mode [90].

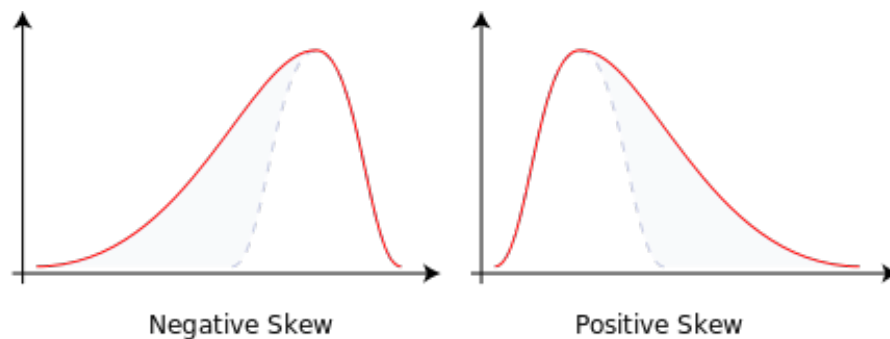


Figure H.1: Negative and positive skew diagrams (Hermans, 2008)

The main methods for measuring skewness are the methods of Karl Pearson, Bowley, and Kelly [90].

There are many methods applicable to small samples [6]. The definition used here is

$$\text{Skewness} = \frac{n^2}{(n-1)(n-2)} \frac{m_3}{s^3},$$

where m_3 , the sample third central moment, is

$$m_3 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^3,$$

and s , the sample standard deviation, is

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2} \quad [24]^1.$$

The variance of skewness is

$$\text{var}(\text{Skewness}) = \frac{6n(n-1)}{(n-2)(n+1)(n+3)} \quad [24, 104].$$

Then, the standard deviation of the skewness is

$$\text{SD}(\text{Skewness}) = \sqrt{\text{var}(\text{Skewness})}.$$

There exist a test statistic for skewness,

$$z_1 = \frac{\text{Skewness}}{\text{SD}(\text{Skewness})} \sim N(0, 1) \quad [24].$$

¹taken from http://www.xycoon.com/skewness_small_sample_test_1.htm

H.6 Normality

A population or sample has *normality* when it follows the normal distribution [107].

There are many methods for appraising normality. They fall into one of three categories [107]:

Descriptive statistics

Descriptive statistics can be used to judge if the data follows the normal distribution [161] as this distribution has some unique characteristics relatives to “centralization” and “dispersion” [69].

Among the descriptive statistics that appraise centralization are the mean, the mode, and the median. In this distribution, all of them are equal [69].

Among the descriptive statistics that evaluate dispersion are the skewness and the kurtosis [124]. In this distribution, skewness is 0 and kurtosis is 3 [170]. Some authors [107] suggests kurtosis must be close to 0, but they refer to *excess kurtosis*.

If descriptive statistics for a sample are close to the expected values for a normal distribution, a normal distribution can be assumed.

Statistical graphics

Graphs are more useful than descriptive statistics to appraise normality. Histograms, box plots, and (normal) probability plots are the most used graphs for this. Others used sometimes are stem-and-leaf diagrams, dot plots, and Q-Q plots [107].

The normal probability plot is a more specialized display to check normality than histograms, box plots, stem-and-leaf diagrams, and dot plots [190]. For this reason, the normal probability plot has been used in this work (this graph is explained below).

Although it is possible to judge normality by using plots, it is subjective and rather difficult to do this, so it is useful to apply a test of normality [60, 170].

Statistical tests

Apparently, this is the best choice to appraise normality.

The most used are the Shapiro-Wilk, Kolmogorov-Smirnov, and Chi-squared tests [107].

It is crucial to be aware of the power of these tests, and their power depends on the sample size [152]. These tests may suffer from low power, i.e. they may fail to reject H_0 when it is false (Wilcox, 2003, cited in [110]). If the sample is small, a test of normality can be useless [76].

The hypotheses for the tests of normality are [170]

H_0 : The sample comes from a normal distribution.

H_1 : The sample comes from a non-normal distribution.

Remember that fail to reject H_0 is not the same as accept H_0 . If the sample is small, low power impedes rejecting H_0 . In this case, it has been advised to look at the statistical graphics as well [110].

H.6.1 Normal Probability Plot

To construct this graph [212, 137], first rearrange the data set in ascending order. Let x_1, x_2, \dots, x_n the reordered data set. The subscript i represents the *rank* of the particular data value. The estimated cumulative probability, P_i , for each value in the reordered data set is

$$P_i = \frac{i - 0.5}{n}. \quad (\text{H.1})$$

The ordered observations x_i are then plotted against P_i on a *normal probability paper* [137]². If this graph shows roughly a straight line, the data follows roughly a

²It is easy to find this kind of paper on the Internet.

normal distribution [190].

It is also possible to use a software package to plot this graphic.

H.6.2 Shapiro-Wilk Test

In the last years, this test has become the preferred one as it compares favorably to a wide range of alternative tests [91]. The Shapiro-Wilk test is the most powerful normality test when it is compared with the Kolmogorov-Smirnov, Lilliefors, and Anderson-Darling tests [163].

This test is suitable when the sample is small (less than 50); otherwise, it is too exigent [164].

This test statistic is denoted by W and ranges from 0 to 1 [152]. The critical values can be searched in a table; for example, in [123].

To compute this test by hand is burdensome [11].

H.7 Paired Difference Tests

If a sample of differences comes from a normal population with standard deviation unknown, a t-test for correlated samples is appropriate [66]. Otherwise, it is compulsory to apply non-parametric tests, such as the sign and the Wilcoxon signed-rank test. These tests are explained below.

H.7.1 t-Test for Correlated Samples

It is also called *t-test for pairs*, *t-test for paired samples*, *t-test for matched samples*, and *t-test for related samples* *t-test for dependent samples*, *repeated-measures t-test*,

and *within-subjects t* [167, 134].

This test has these assumptions [204]:

1. The distribution of the population is normal.
2. The distribution of pairwise differences is normal, and the differences are a random sample.
3. Cases must be independent of each other.

As it was not applied in this work, it is not described.

H.7.2 Wilcoxon Signed-rank Test

The *Wilcoxon signed-rank test* is used to test hypotheses about the median of a population or the difference between paired measurements. The second use is more frequent [93]; this section explains it.

The Wilcoxon signed-rank test is the non-parametric test appropriate to compare two related samples [41]. This test is especially useful when the sample is small ($n < 30$), and the differences are distributed strongly non-normal [187]. It requires numerical data and a population (conformed by the differences in paired values) symmetrically distributed [64, 187, 213]. When the data are non-symmetrical or highly skewed, it can be used the sign test [213].

The power efficiency of this test relative to the t-test for correlated samples is 95.5% when the assumptions of the latter test are met. Thus, the Wilcoxon signed-rank test is an excellent alternative to the t-test for correlated samples [99].

These assumptions must be checked [1]

1. The differences are a random sample from a population of differences with unknown median, M .

2. The differences are calculated on quantitative data having the *interval property*.
3. The population of differences has a continuous distribution and is symmetric.

It is computed following these steps [169, 129]:

1. *This is the null hypothesis:*

$$H_0: m = 0.$$

2. *Choose one of these alternative hypotheses:*

- (a) $H_1: m \neq 0$

- (b) $H_1: m > 0$

- (c) $H_1: m < 0$.

3. *Collect or choose a random sample of paired data.*
4. Compute the difference for each pair of observations. Discard differences equal to zero, and reduce the number of pairs, n , accordingly.
5. Rank the remaining pairs by assigning 1 to the smallest absolute difference, 2 to the next one, \dots , n to the highest one. If there are some ties in the differences, assign the mean of their ranks to each of them.
6. Calculate the rank sum for the negative differences, T^- , and for the positive ones, T^+ . Be T the lower value between T^- and T^+ .
7. *Find the p -value or, alternatively, T_0 .*

To find the p -value,

- (a) To test $H_1: m \neq 0$, a two-tailed test, use T and n to compute the p -value.

- (b) To test $H_1: m > 0$, a one-sided test, use T^+ and n to compute the p -value.

- (c) To test $H_1: m < 0$, use T^- and n to compute the p -value.

Alternatively, use α and n to find the value T_0 in a table of critical values of T .

8. *Conclude.*

If the p -value is less than α (alternatively, if $T_0 < T$ [T^+ or T^-]), reject H_0 and conclude that the median difference is not zero, is greater, or less than zero respectively, otherwise do not reject H_0 .

H.7.3 Sign Test

The *sign test* is used to test hypotheses about the median of a population or the difference between paired measurements [169]. This section explains the latter.

This is a non-parametric test used to compare sample distributions from two populations that are not independent [28]. This test can be used without taking care of the distributions. However, this test is not very powerful [200].

It only requires numerical or ordinal data [200]. These measurements are replaced by plus and minus signs, and then the signs are considered a sample from a binomial population [56].

It is computed following these steps [169]:

1. *This is the null hypothesis:*

$$H_0: m = 0.$$

2. *Choose one of these alternative hypotheses:*

(a) $H_1: m \neq 0$

(b) $H_1: m > 0$

(c) $H_1: m < 0$.

3. *Collect or choose a random sample of paired data.*4. Compute the number of times k that the difference for each pair of observations is greater than zero. Discard differences equal to zero and reduce the number of pairs, n , accordingly.

5. *Find the p-value.*

The p -value, aka P -value, is the probability that the found effect, or a more extreme one, would occur if the null hypothesis were true [191].

The number k under the null hypothesis has a binomial distribution, with $\pi = \frac{1}{2}$ [202] (the proportion of positive signs in the population is denoted π [220]).

(a) For $H_1: m < 0$, it is necessary to compute p -value = $P(x \leq k) = P(x=0) + P(x=1) + \dots + P(x=k)$, where $P(x=i)$ is the probability of $x=i$ in the binomial distribution. As computing this could be cumbersome, it is easier to find $P(x \leq k)$ directly in a cumulative binomial distribution table [129].

(b) For $H_1: m > 0$, it is necessary to compute p -value = $P(x \geq k) = 1 - P(x \leq k)$; find $P(x \leq k)$ as before.

(c) For $H_1: m \neq 0$ it is necessary to compute p -value = $2P(x \leq k)$ [7]; find $P(x \leq k)$ as before.

6. *Conclude.*

If the p -value is less than α , reject H_0 and conclude that the median difference is not zero, is greater, or less than zero respectively, otherwise do not reject H_0 .

H.8 Rank Correlation

A ranking is an ordering of some elements, so an element is higher, at the same level, or lower than another. Rank correlations try to establish a relationship between different rankings.

The rank correlations of Kendall and Spearman assume that the relationship between the two variables is monotonic (i.e. either increasing or decreasing), though not necessarily linearly. The rank correlation coefficients range from -1 to +1, where -1 means perfect negative correlation and +1 means perfect positive correlation [113].

The correlation of Kendall has approximately the same power as the correlation of Spearman [113], but it is more appropriate if there are tied ranks [75]. Consequently, only the former is described.

H.8.1 Correlation Coefficient of Kendall

The *correlation coefficient of Kendall* is a non-parametric test computed on ranks [75]. It is subject to less stringent assumptions than the parametric correlation of Pearson [75]. It has good properties but high computation complexity ($O(n^2)$) [227].

To compute this coefficient [131], for each pair of observations, assign +1 if the pair is ordered (concordant), otherwise assign -1 (discordant). Compute the number C of concordant pairs and the number D of discordant pairs. Compute K as C less D . The coefficient known as *tau* of Kendall is

$$\tau = \frac{2K}{n(n-1)} \quad (\text{H.2})$$

The distribution of *tau* can be found in tables for small samples or approximated through a normal distribution for large samples ($n > 10$) by using these parameters [131]:

$$E[\tau] = 0; \text{Var}[\tau] = \frac{2(2n+5)}{9n(n-1)}. \quad (\text{H.3})$$

Choosing an Interactive Segmentation Method

This chapter explains how to choose an interactive segmentation method. However, it has recommended a method whose program is not available in the Internet.

I.1 Measurement of Interactive Segmentation Methods

Some criteria are needed to appraise interactive segmentation methods. In the literature, three criteria have emerged as a standard for this: accuracy, repeatability, and efficiency (Udupa and Herman, 2000, cited in [147]). These criteria are interdependent [201]; for example, improving accuracy usually implies losing efficiency.

I.1.1 Accuracy

Accuracy is the degree of agreement between a segmented image and its ground truth [201]. It is possible to evaluate the accuracy by using human experts (subjective evaluation), or by using different distance measures between the ground truth and the segmented image (objective evaluation). Accuracy is the most used evaluation criterion [147].

In interactive methods, the user can improve the accuracy to a level that is always

satisfactory. This is not true when some limitations are imposed on him [147]. For example, limiting the time to complete the segmentation [126].

Previous works have considered both boundary and object accuracy. Boundary accuracy shows how well the boundary of the segmented region matches its ground truth. Object accuracy shows how well the whole region matches its ground truth [127]. Large overlapped areas overcome small defects around the boundary when object accuracy is evaluated [126]. In this work, object accuracy is more important than boundary accuracy because we are interested in the entire region accuracy.

It has been proposed [173] appraising object accuracy by using the arithmetic mean of the coefficient of Dice over all the image segments. This is

$$Dice - score = \frac{1}{N} \sum_{i=1}^N Dice(E_i, GT_i),$$

where E_i and GT_i are the i -th of N segment for the machine segmented image and the ground truth, respectively. The Dice coefficient is defined as

$$Dice(E_1, E_2) = \frac{2|E_1 \cap E_2|}{|E_1| + |E_2|},$$

where E_1 and E_2 are two segments, and $| |$ denotes the area of a segment.

Also, the binary *Jaccard index*, J , has been adapted to measure object accuracy. This is

$$J = \frac{|G_O \cap M_O|}{|G_O \cup M_O|},$$

where G_O is the ground-truth object, and M_O is the segmented object [126].

This index outperformed several commonly used measures (precision, recall, and the Rand index) in a comparison between measured and perceived accuracy [126].

Notice that the Dice coefficient can be computed from the Jaccard index by using

this simple formula

$$D = 2J/(1 + J).$$

In addition, the Dice coefficient can be converted into the Jaccard index by using this formula

$$J = D/(2 - D).$$

However, there are no such formulae for the Dice-score. Thus, the Dice-score and the Jaccard index cannot be compared. As the Jaccard index was tested against perceived accuracy, it was chosen.

I.1.2 Repeatability

Repeatability, also known as precision, measures the similarity of different segmentations obtained when a user has always pursued the same goal. In other words, a user, trying to get identical ROIs, segments an image many times. The difference in the results depends on the inputs given to the segmentation tool, such as where the user traces scribbles or makes clicks. This is an *intra-operator repeatability*. It is also possible to measure the *inter-operator repeatability*, but different users might differ in what they consider the ideal segmentation [147]. The latter class of repeatability was used in [126].

I.1.3 Efficiency

In this context, efficiency is to segment an image by using as few resources as possible.

Evaluating the efficiency of interactive segmentation methods is very subjective. However, it is reasonable to say that a method of this kind is efficient when its computational part is fast, autonomous, and predictable, and when the human interactions are few, quick, and simple [147].

Time is an important measure for interactive methods since they should give an

accurate segmentation faster than it would take to make it by hand [126].

I.2 Datasets for Evaluating Interactive Segmentation Methods

This work requires an interactive, multi-label segmentation method. Consequently, it is necessary a multi-label dataset with the regions to segment signaled somehow (see some methods of signaling in Fig. A.15). Unfortunately, only *IcgBench*¹ fulfills these conditions. However, the *Berkeley Segmentation Dataset* [122] (BSDS) has been used anyway.

The publicly available IcgBench dataset could be used to evaluate methods that use scribbles. This dataset gives 262 seed/ground truth pairs on 158 different natural images [173]. This dataset contains 243 images; 85 images do not have a seed/ground truth pair.

BSDS contains segmentations made by humans for images from a wide range of natural scenes. It was proposed that segmentations could be evaluated by comparing them to the segmentations made by multiple humans. As different human segmentations of the same image are very consistent, this comparison is reliable [122]. As different segmentations of the same image made by one human has lower variability than those of distinct humans, the former could not be used as ground truth [197].

BSDS cannot be used to compare interactive segmentation methods *automatically* since it is an unsupervised segmentation benchmark (it does not provide seeds) [173]. However, BSDS has been used to compare these methods by using user experiments [126].

The publicly available *MSRC*² database was constructed to train and test object recognition and segmentation methods. This database contains 591 photographs;

¹www.gpu4vision.org

²<http://research.microsoft.com/en-us/projects/objectclassrecognition/>

they were segmented and classified in 21 classes (for example, building, grass, tree, cow). The ground-truth images also contain pixels labeled as “void”. This class signals pixels that do not belong to any other class, and to allow for a rough and quick, hand-made segmentation [185].

MSRC seems suitable for figure-ground segmentation but not for image segmentation.

The publicly available *GrabCut*³ dataset specifies background, foreground, and mixed area. The publicly available *LHI*⁴ dataset specifies both a set of objects (foreground) and their background.

Consequently, Grabcut and LHI can be used to evaluate only foreground and background segmentations.

I.3 Comparison of Interactive Segmentation Methods

Comparison of accuracy between this class of methods has been made by using the Jaccard index on the BSDS and the Dice-score on the IcgBench dataset. Other indexes and datasets were not considered by the reasons given in the previous sections. For this reason, some promising methods were not considered, such as *Multi-label Interactive CRF* (Conditional Random Fields) and *Multi-label Interactive Higher Order CRF* [141].

Seeded Region Growing (SRG), *Simple Interactive Object Extraction* (SIOX), *Interactive Graph Cuts* (IGC), and *Interactive Segmentation using Binary Partition Trees* (BPT) give a good coverage of the algorithmic approaches for interactive segmentation. They were compared by using the Jaccard index on 100 distinct objects selected from 96 images in the BSDS. In this evaluation, the users segmented each

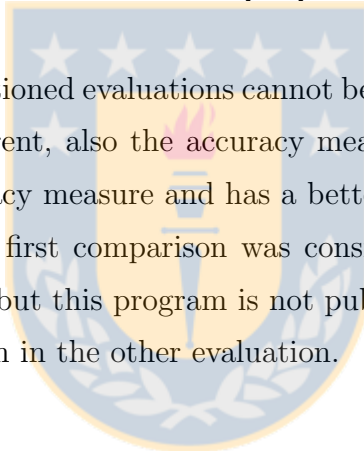
³<http://research.microsoft.com/en-us/um/cambridge/projects/visionimagevideoediting/segmentation/grabcut.htm>

⁴<http://www.imageparsing.com/interactivesegmentation.html>

object by marking the foreground and background areas with the mouse. The time to segment each object was limited to 2 minutes, but the users could finish before. BPT and IGC achieve similar accuracy. Both attained better accuracy than SRG and SIOX [126]. IGC and SIOX are two-label [27, 57] while BPT and SRG are multi-label [172, 4]. This comparison uses a two-label segmentation, so it is quite possible that BPT and SRG cannot segment multiple labels as efficiently as segmenting only two.

By using the Dice-score on the IcgBench dataset were compared *Random Forest*, *Graph cuts with α -expansion*, and *Partial Differential Equations* (PDE). PDE attained better accuracy than the other methods [142].

Unfortunately, the aforementioned evaluations cannot be compared between them. Not only the datasets are different, also the accuracy measures. However, the first comparison uses a better accuracy measure and has a better coverage of algorithmic approaches. Consequently, the first comparison was considered to choose an algorithm. Thus, BPT was chosen, but this program is not publicly available. The same happens with the best algorithm in the other evaluation.



Compression and Expansion between the Borders and the Birthplace

This approach, even with the improvement suggested below, cannot be used always. It should be used only when the *Compression and Expansion between the Borders and the Skeleton* fails. Its justification lies on replacing incorrect nucleus with birthplaces and on replacing swinging line segments with stable ones.

Take a couple of convex regions without holes —a big and a small one. Line segments are computed from each border point of the big region to the birthplace. Imagine that these (line) segments also exist in the slice that contains the small region. Consequently, these segments go through the small region. For example, in Fig. J.4, a point in the border of the big region is a , the birthplace is the point c , the segment is \overline{ac} and the part within the small region is \overline{bc} . In the compression, each segment is compacted in the part of it within the small region. In the expansion, the part of each segment within the small region is extended to the whole segment.

If a region is not convex, some segments might not belong entirely to this region. In this case, it would be better to trace the shortest curve within the region between the birthplace and the border point. It can be obtained by computing the geodesic distance with the birthplace as a seed.

As both the big and small regions could be non-convex, the geodesic distance in the big region can be computed by considering the small region as a seed. Consequently, the path between each point in the external border and the birthplace would be the union of two minimal paths: one from this border point to the small region, and the other from the end of the first path to the birthplace. In the compression, each path would be compacted in the part of it within the small region. In the expansion, the part of each path within the small region would be extended in the whole path.

Nevertheless, as it was not available a routine to compute geodesic distances, they were not used.

As a solution for the (line) segments that went out the processed region, each part of these segments —that passed through the big region— was considered a new segment. For example, in Fig. J.1, a region included the body of a man but excluded his head, neck, arms, and left shoe. This region touched the border in the right foot. The birthplace was defined (the real program would have computed a point) as the intersection of this foot with the border of the image. So, some segments started in the external border of the left leg (rightward in the image); passed through this leg, the background, and the right leg, and finished in its right foot. However, following this solution, there were new segments in the left and right legs. Then, each new segment was processed by following the rules stated in the second paragraph. The interpolation obtained can be seen in Fig. J.2. There, it was possible to see some conspicuous segments pointing to the right foot. This solution is far from perfect: the segments traced following this idea have no theoretical justification.

As birthplaces might take different shapes, the method for computing the compression and the expansion changes. The methods proposed are described below. They were integrated in the same algorithm (see Fig. J.3). The initial steps of this algorithm are:

- (1) The compression and expansion are similar, so it was possible to unify them in

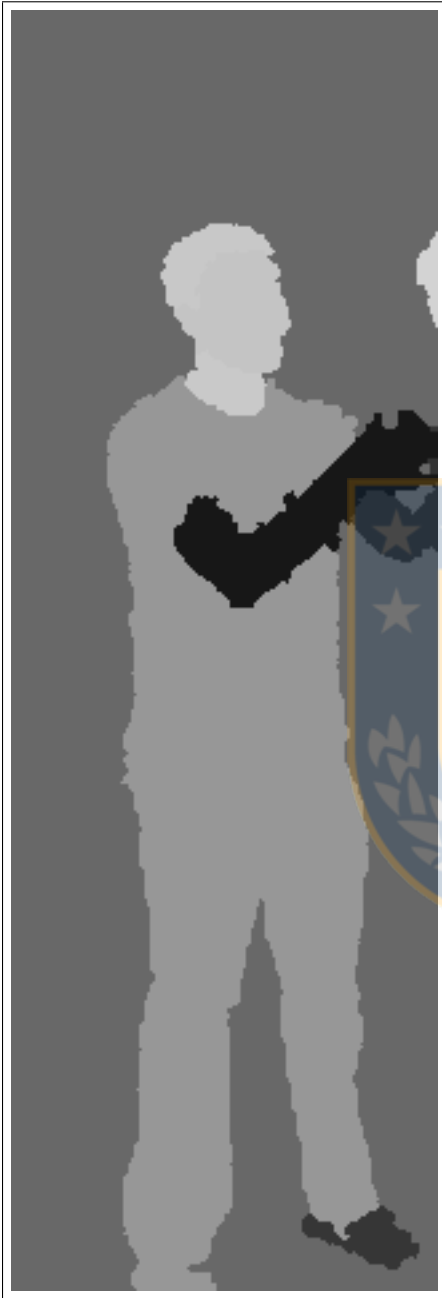


Figure J.1: Detail of a man in the grey-level mosaic obtained segmenting the image basketball12.png



Figure J.2: Detail of a man in the interpolation of basketball12.png and basketball14.png by using birthplaces

the same algorithm. This step followed this goal by filling the small and big regions. Notice that in “expand” the final region was the big one, and in “compress” the final region was the small one.

(2) The birthplace was computed by shrinking the small region.

(3) Depending on the presence of holes in the small region, different steps were taken to perform the compression or expansion. If there was no holes, it went to step 4; otherwise, to step 5.

The step (4) was described in *Compression and Expansion between the Outer Border and a Set of Points* (see Sec. J.1). The step (5) was described in *Compression and Expansion between the Borders and Rings* (see Sec. J.2).

J.1 Compression and Expansion between the Outer Border and a Set of Points

This method can compress and expand paired regions without holes. It should function well when an object without holes moves forward (or backward) or grows, such as a balloon being filled.

It was supposed that the birthplace is a point (MSP) computed with shrinking on the small region. Segments were computed from the border of the big region to the MSP.

For example (see below, Fig. J.4), there was a point (a) in the border of the big region, a point (b) in the border of the small region, and a point (c) that corresponded to the MSP of the small region. In the compression, the pixels on \overline{ac} were squeezed on \overline{bc} ; and in the expansion, the pixels on \overline{bc} were stretched on \overline{ac} .

This method can manage regions with multiple MSPs (disjoint regions). For this,

```

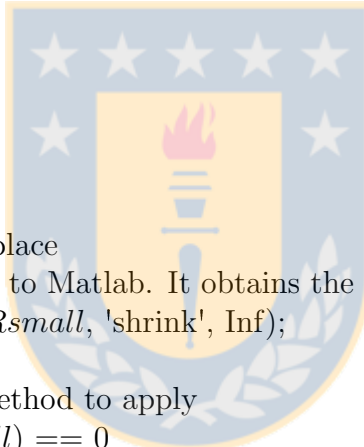
expanding_or_compressing_Image_by_using_birthplace (Rcolorinicial: colorImage;
Rinicial, Rfinal: binaryImage, operacion:string): colorRegion {
Rbig, Rsmall, borde, borde_imagen, centro, agujeros, agujeros_rellenos,
punto_de_fuga: binaryImage
SE = 8; // 8-connected neighborhood
// See the description of bordes in the description of B in
// http://www.mathworks.com/help/images/ref/bwboundaries.html
bordes: array
camino, camino_siguiente: line segment

// (1) These sentences allow that the expansion and the compression work in
// the same algorithm. The next instruction belongs to Matlab.
if strcmp (operacion, "expand") == 1
    Rsmall = Rinicial
    Rbig = Rfinal
else
    Rsmall = Rfinal
    Rbig = Rinicial
end

// (2) Finding the birthplace
// This sentence belongs to Matlab. It obtains the birthplace
birthplace = bwmorph(Rsmall, 'shrink', Inf);

// (3) Deciding which method to apply
If numberofholes (Rsmall) == 0
    // (4) There is no holes, so the birthplace is a set of points.
    // (4.1) This Matlab sentence obtains the border from Rbig.
    borde = bwperim (Rbig,SE);
    // (4.2) This Matlab sentence obtains the points from the birthplace.
    [row_birthplace, col_birthplace] = find(birthplace);
    // (4.3) For each point in the border
    For each pixel p in borde
        // (4.3.1) Finding the segment between p and the (nearest) point in the
        // birthplace
        If nnz(birthplace)==1
            camino = Buscar_camino (row (p), col (p), row_birthplace,
            col_birthplace);
        else
            camino_mas_corto_al_birthplace = buscar_camino_usando_la_forma_
            interna(row (p), col (p), row (p), col (p), SE);

```



```

camino = Buscar_camino (row (p), col (p), camino_mas_corto_al_
    birthplace (length(camino_mas_corto_al_birthplace)-1), camino_mas_
    corto_al_birthplace (length(camino_mas_corto_al_birthplace)));
end
// (4.3.2) Drawing segment between the border of Rbig or Rsmall and
// the birthplace.
Rcolorfinal = ponerPixeles_en_la_forma_small_o_big (camino,
    Rcolorinicial, Rcolorfinal, Rsmall, Rbig, operacion);
end
else
// (5) There are some holes, so the birthplace is a set of rings.
// (5.1) This sentence obtains the external borders from the region and from
// its holes.
bordes = bwboundaries (Rbig, 4, 'holes');
// (5.2) Finding the center of the area enclosed by the ring(s)
birthplace_relleno = imfill (birthplace, 'holes');
centro = bwmorph(birthplace_relleno, 'shrink ', Inf);
[fila_centro, columna_centro] = find (centro);
// (5.3) Extracting the most external border
circunvalacion = bordes {1, 1};
// (5.4) For each pixel p in circunvalacion
// The sentences belonging to this “for” cycle belong to Matlab
for indice_pixel=1:length(circunvalacion)
// (5.4.1) Finding the segment between p and the (nearest) point in a
// center.
If nnz(centro) == 1
// The center has only one pixel
camino = Buscar_camino (circunvalacion (indice_pixel,1),
    circunvalacion (indice_pixel,2), fila_centro(1), columna_centro (1));
else
// Finding the shortest route to a center
camino_mas_corto_al_centro = buscar_camino_usando_la_forma_interna
(circunvalacion (indice_pixel,1), circunvalacion (indice_pixel,2),
    circunvalacion (indice_pixel,1), circunvalacion (indice_pixel,2), SE);
// Finding the segment to the nearest point in the center
camino = Buscar_camino (circunvalacion (indice_pixel,1),
    circunvalacion (indice_pixel,2), camino_mas_corto_al_centro (length
    (camino_mas_corto_al_centro)-1), camino_mas_corto_al_centro (length
    (camino_mas_corto_al_centro)));
end
end

```

```

// (5.4.2) Drawing segments between the border(s) and the filled ring(s).
Rcolorfinal = ponerPixeles.en.la.forma.small.o.big (camino,
Rcolorinicial, Rcolorfinal, Rsmall, Rbig, operacion);
end
// (6) Tracing segments between the ring(s) and the internal borders
// (6.1) Finding vanishing points (called punto_de_fuga).
agujeros_rellenos = birthplace_relleno & imcomplement(Rsmall);
punto_de_fuga = bwmorph(agujeros_rellenos, 'shrink', Inf);
[filas_puntos_de_fuga, columnas_puntos_de_fuga] = find (punto_de_fuga);

// (6.2) Finding perimeter(s) of the internal ring(s). This sentence belong
// to Matlab.
[bordes_birthplace, L, numero_de_objetos, ~] = bwboundaries(birthplace, 8,
'holes');

// (6.3) For each internal ring
for index=numero_de_objetos+1:length(bordes_birthplace)
// (6.3.1) Finding the vanishing point belonging to this ring.
for indice_punto_de_fuga=1:nnz(punto_de_fuga)
if L(filas_puntos_de_fuga (indice_punto_de_fuga), columnas_puntos_de_
fuga (indice_punto_de_fuga)) == index
filas_punto_de_fuga=filas_puntos_de_fuga(indice_punto_de_fuga);
columnas_punto_de_fuga=columnas_puntos_de_fuga(indice_punto_de_fuga);
end
end
// (6.3.2) Extracting the perimeter of this ring.
circunvalacion_anillo=bordes_birthplace{index};
// (6.3.3) For each pixel p in circunvalacion_anillo
// The sentences belonging to this “for” cycle belong to Matlab
for indice_pixel=1:length(circunvalacion_anillo)
// (6.3.3.1) Finding the segment from p to this vanishing point
camino = Buscar_camino (circunvalacion_anillo (indice_pixel,1),
circunvalacion_anillo (indice_pixel,2), filas_punto_de_fuga,
columnas_punto_de_fuga);
// (6.3.3.2) Drawing along camino between p and a border
Rcolorfinal = ponerPixeles.en.la.forma.small.o.big (camino,
Rcolorinicial, Rcolorfinal, Rsmall, Rbig, operacion);
end
end
end
return (Rcolorfinal)
}

```

Figure J.3: expanding_or_compressing_Image_by_using_birthplace

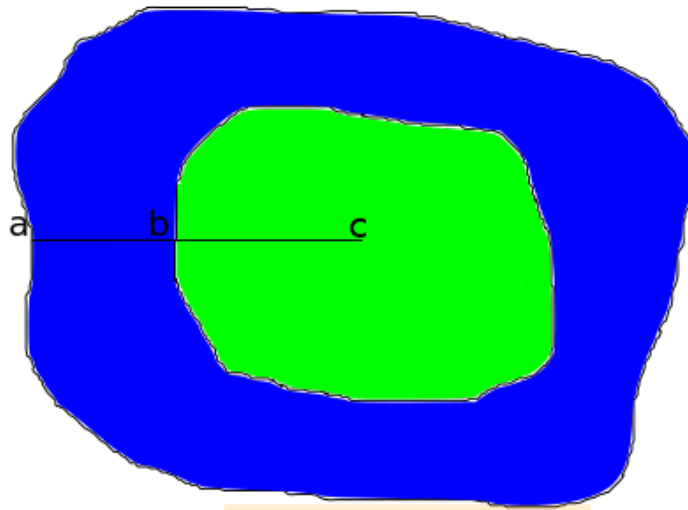


Figure J.4: Segment between the outer border and the MSP

it finds, for each point of the border, the nearest point belonging to the birthplace and computes the segment between these points.

Also, this method can be used to implement the *Compression and Expansion between the Borders and an Artificial Connected Component* (except those corresponding to type 8 regions) by changing the borders and the birthplace. The birthplace should be replaced by an artificial connected component. Replacing the border is a little more complex (see Sec. J.3).

This method was implemented in the step (4) of the *expanding_or_compressing_Image_by_using_birthplace* algorithm (see Fig. J.3). This step includes these sub-steps:

(4.1) Obtaining the border of the big region.

(4.2) Computing the coordinates of each point belonging to the birthplace.

(4.3) For each point p in the border,

(4.3.1) Finding the (shortest) segment between p and the birthplace, and store it in *camino*.

(4.3.2) Three cases were possible considering that these segments began in the borders of *Rbig* and finished in *Rsmall*: 1) *camino* did not reach *Rsmall* before leaving *Rbig*—these pixels were discarded—; 2) *camino* reached *Rsmall* and without leaving it reached the birthplace, and 3) *camino* reached *Rsmall*, and leaving and reentering it, reached the birthplace—if *camino* did not leave *Rbig* before reentering *Rsmall*, the results were erroneous. In the case 2, the pixels along *camino* were copied along the part of it inside the small region in the compression and the pixels along *camino* inside the small region were stretched along *camino* in the expansion. In the case 3, it was applied the same procedure as in the case 2, but it was considered only the part of *camino* between the beginning of it and the first time it leaved the big region.

J.2 Compression and Expansion between the Borders and Rings

This method can compress and expand paired regions with some holes. In this case, each region has an external border and some internal ones.

If a region has a hole, shrinking (see Sec. B.7) creates a ring halfway its hole and its outer border. It was supposed that an object starts growing from its ring. It was very difficult to find examples in which this premise functions: maybe a (ring) doughnut before and after leavening or frying it.

If a region has several holes, shrinking creates a ring for each hole, but some rings may overlap.

According to this strategy, segments were traced 1) between the outer border of a region and the perimeter of the filled rings; and then 2) between the rings and the

inner borders (of this region):

1) It seemed possible to compute segments from each point in the outer border of the big region to the nearest point in the rings. However, this left many black areas when the rings had rough borders. To avoid this problem, segments were computed from the outer border of the big region to the “MSP of the filled rings” (center). The pixels in each of these segments between the outer border of the big region and the rings were squeezed between the outer border of the small region and the rings in the compression; and the pixels between the outer border of the small region and the rings were stretched between the outer border of the big region and the rings in the expansion.

This part of the method was implemented in the step (5) of the *expanding_or_compressing_Image_by_using_birthplace* algorithm (see Fig. J.3). This step includes these sub-steps:

(5.1) The external borders from the big region and from its holes are computed;

(5.2) the regions enclosed by the rings are computed and stored in *birthplace_relleno*. Then, the centers of these regions are computed and stored in *centro*.

(5.3) The external border from the big region is separated and stored in *circunvalacion*.

(5.4) For each point p in *circunvalacion*,

(5.4.1) Finding the segment between p and the (nearest) point in *centro*, and store it in *camino*.

(5.4.2) Three cases are possible considering that these segments begin in the borders of R_{big} and finish in R_{small} : 1) *camino* does not reach R_{small} before leaving

R_{big} —these pixels are discarded—; 2) *camino* reaches R_{small} and without leaving it reaches *birthplace_relleno*, and 3) *camino* reaches R_{small} , and leaving and reentering it, reaches the *birthplace_relleno*—if *camino* does not leave R_{big} before reentering R_{small} , the results are erroneous. In the case 2, only the pixels from *camino* between its beginning and *birthplace_relleno* are considered. In other words, it is considered that *camino* finishes at *birthplace_relleno*. The pixels along *camino* are copied along the part of it inside the small region in the compression and the pixels along *camino* inside the small region are stretched along *camino* in the expansion. In case 3, it is applied the same procedure as in the case 2, but it is considered only the part of *camino* between the beginning of it and the first time it leaves the big region.

2) Segments are computed from the rings to the “MSPs of the holes” (vanishing points). The pixels of these segments between the inner border of the big region and the ring are squeezed between the inner border of the small region and the ring in the compression; and the pixels of these segments between the inner border of the small region and the ring are stretched between the inner border of the big region and the ring in the expansion.

This task is performed by the step (6) of the *expanding_or_compressing_Image_by_using_birthplace* algorithm (see Fig. J.3). This step includes these sub-steps:

(6.1) Computing vanishing points as the centers of the holes, and store them in *punto_de_fuga*.

(6.2) Finding perimeters of the rings. As this is difficult to compute, the perimeters of the holes adjacent to the rings were computed, they are called *internal rings*.

(6.3) For each internal ring,

(6.3.1) Finding the vanishing point associated to this ring.

(6.3.2) Extracting the perimeter of this ring, and store it in *circunvalacion_anillo*.

(6.3.3) For each pixel p in *circunvalacion_anillo*,

(6.3.3.1) Finding the segment between p and the vanishing point, and store it in *camino*.

(6.3.3.2) Drawing along *camino* between p and a border. The border is the internal border of the small region in the compression and the internal border of the big region in the expansion.

For example (see below, Fig. J.5), there is a point (a) in the outer border of the big region, a point (b) in the outer border of the small region, a point (c) in the ring, a point (d) in the inner border of the small region, a point (e) in the inner border of the big region, and a point (f) that corresponds to the center and the vanishing point simultaneously. In the compression, the pixels in \overline{ac} and \overline{ce} should be squeezed in \overline{bc} and \overline{cd} , respectively; and in the expansion, the pixels on \overline{bc} and \overline{cd} should be stretched on \overline{ac} and \overline{ce} , respectively.

Why not compress directly \overline{ae} in \overline{bd} , and expand \overline{bd} in \overline{ae} ? Because it was supposed that the regions start to grow from their ring and because the center and the vanishing point does not necessarily coincide.

When an object and its hole grew too much, sometimes this strategy did not handle them adequately. In addition, a small deficiency exists in the implementation of this algorithm. Sometimes, there were black lines where the birthplace divides various holes because segments were traced from the border of the holes surrounded by the birthplaces instead of the birthplaces themselves. For example, the interpolation of the images in Fig. 4.6 by using the segmentations shown in Fig. J.6 gave Fig. J.7.



Figure J.5: Segment between the outer border and the center

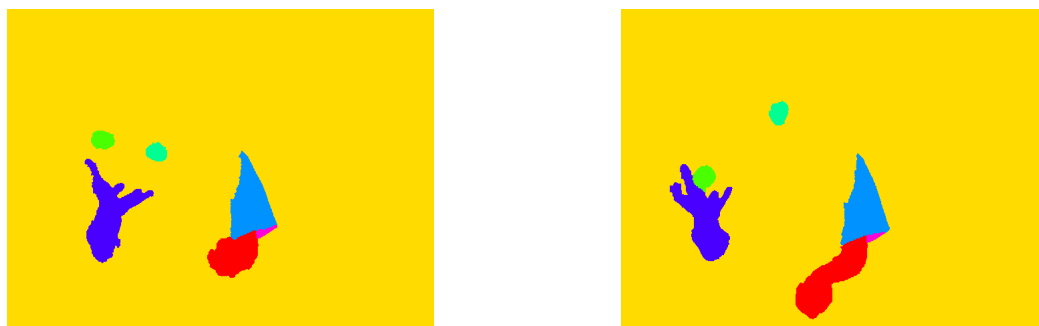


Figure J.6: Matched segmentation of the initial and final images of the sequence called beanbags: Leftward and rightward, respectively



Figure J.7: Interpolated image of the sequence called beanbags by using birthplaces and the linear interpolation

J.3 Compression and Expansion between the Borders and an Artificial Connected Component

Although the two methods explained above can process conjointly any kind of regions, when a *original* region is not paired, lies in the border of the image, and has no holes, the birthplace could not be optimal. So, it was tried to use an artificial connected component instead. It was not considered in the final version of the compression and expansion algorithm as the grey-level interpolation algorithm sometimes gave incorrect grey-level interpolated images in this case.

Bibliography

- [1] M.L. Abell, J.P. Braselton, and J.A. Rafter. *Statistics with Mathematica: Text. Statistics with Mathematica*. Academic Press, 1999.
- [2] T. Acharya and A.K. Ray. *Image Processing: Principles and Applications*. Wiley-Interscience, 2005.
- [3] K. Adamczyk and A. Walczak. Application of 2D Anisotropic Wavelet Edge Extractors for Image Interpolation. *Human-Computer Systems Interaction: Backgrounds and Applications 2*, pages 205–222, 2012.
- [4] R. Adams and L. Bischof. Seeded region growing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(6):641–647, 1994.
- [5] A. Albu, J.M. Schwartz, D. Laurendeau, and C. Moisan. Integrating geometric and biomechanical models of a liver tumour for cryosurgery simulation. *Surgery Simulation and Soft Tissue Modeling*, pages 999–999, 2003.
- [6] C. Alexander. *Market Risk Analysis, Quantitative Methods in Finance*. Wiley Desktop Editions. Wiley, 2008.
- [7] D.R. Anderson, D.J. Sweeney, and T.A. Williams. *Statistics for Business and Economics (Book Only)*. Thomson South-Western, 2008.
- [8] E. Aptoula and S. Lefèvre. A basin morphology approach to colour image segmentation by region merging. In *Proceedings of the 8th Asian conference on Computer vision-Volume Part I*, pages 935–944. Springer-Verlag, 2007.
- [9] E. Aptoula and S. Lefèvre. A comparative study on multivariate mathematical morphology. *Pattern Recognition*, 40(11):2914–2929, 2007.

- [10] Erchan Aptoula and Sébastien Lefèvre. 1 Morphological Texture Description of Grey-Scale and Color Images. In Peter. W. Hawkes, editor, *Advances in Imaging and Electron Physics: Optics of Charged Particle Analyzers*, volume 169. Elsevier Science, 2011.
- [11] J.S. Armstrong. *Principles of Forecasting: A Handbook for Researchers and Practitioners*. International Series in Operations Research & Management Science. Springer, 2001.
- [12] D.G. Bailey. *Design for Embedded Image Processing on FPGAs*. Wiley, 2011.
- [13] Simon Baker, Daniel Scharstein, J.P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.
- [14] P.G. Barash, B.F. Cullen, R.K. Stoelting, M. Cahalan, and C. Stock. *Clinical Anesthesia*. Wolters Kluwer Health, 2012.
- [15] K. Barkaoui, A. Cavalcanti, and A. Cerone. *Theoretical Aspects of Computing - ICTAC 2006: Third International Colloquium, Tunis, Tunisia, November 20-24, 2006 Proceedings*. Lecture Notes in Computer Science / Theoretical Computer Science and General Issues. Springer, 2006.
- [16] V. Barnett. The ordering of multivariate data. *Journal of the Royal Statistical Society. Series A (General)*, 139(3):318–355, 1976.
- [17] R.A. Beasley. Finding the best edge image. In *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition*, pages 504–510, 2011.
- [18] A. Belyaev. On implicit image derivatives and their applications. *Journal of Computational Physics*, 103:16–42, 1992.
- [19] S. Beucher. Digital skeletons in Euclidean and geodesic spaces. *Signal Processing*, 38(1):127–141, 1994.
- [20] Serge Beucher. Sets, partitions and functions interpolations. In *Proceedings of the fourth international symposium on Mathematical morphology and its applications to image and signal processing*, ISMM '98, pages 307–314, Norwell, MA, USA, 1998. Kluwer Academic Publishers.
- [21] J. Beutel, Y.M. Kim, and S.C. Horii. *Handbook of Medical Imaging, Volume 3: Display and Pacs*. Press Monographs. SPIE Optical Engineering Press, 2000.
- [22] S. Bhattacharyya. A Brief Survey of Color Image Preprocessing and Segmentation Techniques. *Journal of Pattern Recognition Research*, 6(1):120–129, 2011.

- [23] H. Blum et al. A transformation for extracting new descriptors of shape. *Models for the perception of speech and visual form*, 19(5):362–380, 1967.
- [24] E. Borghers and P. Wessa. *Statistics - Econometrics - Forecasting*, January 2013.
- [25] A.G. Bors, L. Kechagias, and I. Pitas. Binary morphological shape-based interpolation applied to 3-D tooth reconstruction. *Medical Imaging, IEEE Transactions on*, 21(2):100–108, 2002.
- [26] A.C. Bovik. *Handbook Of Image And Video Processing*. Communications, Networking and Multimedia Series. Elsevier Academic Press, 2005.
- [27] Y.Y. Boykov and M.P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in ND images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112. IEEE, 2001.
- [28] C.H. Brase and C.P. Brase. *Understandable Statistics: Concepts and Methods*. Brooks/Cole, 2011.
- [29] R. Brémond and F. Marqués. Segmentation-based morphological interpolation of partition sequences. *Mathematical Morphology and its Applications to Image and Signal Processing*, pages 369–376, 1996.
- [30] B. Brundage. *Photoshop Elements 10: The Missing Manual*. O'Reilly Media, 2011.
- [31] M. Carnec, P. Le Callet, and D. Barba. Objective quality assessment of color images based on a generic perceptual reduced reference. *Signal Processing: Image Communication*, 23(4):239–256, 2008.
- [32] Y. Chan. *Location Theory and Decision Analysis: Analytics of Spatial Information Technology*. SpringerLink : Bücher. Springer, 2011.
- [33] S.S. Channappayya, The University of Texas at Austin. Electrical, and Computer Engineering. *Image Communication System Design Based on the Structural Similarity Index*. The University of Texas at Austin, 2007.
- [34] J. Chanussot and P. Lambert. Watershed approaches for color image segmentation. In *Proceedings of the IEEE Workshop on Nonlinear Signal and Image Processing*, pages 129–133, 1999.
- [35] Vassilios Chatzis and Ioannis Pitas. Shape-Based Interpolation of Binary 3-D Images using Morphological Skeletonization. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems - Volume 2*, volume 2 of *ICMCS '99*, pages 939–943, Washington, DC, USA, 1999. IEEE Computer Society.

- [36] Fakulta Chemická. HARFA: Screenshots and Tutorials: Overview.
- [37] H.D. Cheng, XH Jiang, Y. Sun, and J. Wang. Color image segmentation: advances and prospects. *Pattern recognition*, 34(12):2259–2281, 2001.
- [38] F. Chin, J. Snoeyink, and C. Wang. Finding the medial axis of a simple polygon in linear time. *Algorithms and Computations*, pages 382–391, 1995.
- [39] S.J. Chiu, X.T. Li, P. Nicholas, C.A. Toth, J.A. Izatt, and S. Farsiu. Automatic segmentation of seven retinal layers in SDOCT images congruent with expert manual segmentation. *Optics express*, 18(18):19413–19428, 2010.
- [40] Mary L. Comer and Edward J. Delp. Morphological operations for color image processing. *J. Electron. Imaging*, 8(3):279–289, July 1999.
- [41] G.W. Corder and D.I. Foreman. *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*. Wiley, 2011.
- [42] D. Cremers, M. Rousson, and R. Deriche. A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape. *International Journal of Computer Vision*, 72(2):195–215, 2007.
- [43] O. Dalmau and M. Rivera. A General Bayesian Markov Random Field Model for Probabilistic Image Segmentation. In *Proceedings of the 13th International Workshop on Combinatorial Image Analysis*, pages 149–161. Springer-Verlag, 2009.
- [44] P. Das, O. Veksler, V. Zavadsky, and Y. Boykov. Semiautomatic segmentation with compact shape prior. *Image and Vision Computing*, 27(1):206–219, 2009.
- [45] V.V. Das and R. Vijaykumar. *Information and Communication Technologies: International Conference, ICT 2010, Kochi, Kerala, India, September 7-9, 2010, Proceedings*. Communications in Computer and Information Science. Springer, 2010.
- [46] E.R. Davies. *Machine Vision: Theory, Algorithms, Practicalities*. Signal Processing and Its Applications. Elsevier Science, 2004.
- [47] Marcos Cordeiro d’Ornellas and R. van Den Boomgaard. A Morphological Multi-Scale Gradient for Color Image Segmentation. In John Goutsias, Luc Vincent, and Dan S. Bloomberg, editors, *Mathematical morphology and its applications to image and signal processing*, volume 18 of *Computational Imaging and Vision*, pages 199–206. Kluwer Academic Publishers, 2002. 10.1007/0-306-47025-X_10.
- [48] E.R. Dougherty. *Electronic Imaging Technology*. Spie Press Series. SPIE Optical Engineering Press, 1999.

- [49] E.R. Dougherty and R.A. Lotufo. *Hands-on Morphological Image Processing*, volume 59. Society of Photo Optical, 2003.
- [50] G. Dougherty. *Digital Image Processing for Medical Applications*. Cambridge University Press, 2009.
- [51] Stephen A. Drury. *Image Interpretation in Geology*. Taylor & Francis, 2004.
- [52] E. DuBois. *The Structure and Properties of Color Spaces and the Representation of Color Images*. Synthesis Lectures on Image, Video, and Multimedia Processing. Morgan & Claypool Publishers, 2010.
- [53] D. Edwards and The University of Utah. *Practical Sampling for Ray-based Rendering*. The University of Utah, 2008.
- [54] J.C. Ferrando and V.G. Gregori. *Matemática discreta 2a Ed.* Reverté, 1995.
- [55] J.D. Foley. *Computer Graphics: Principles and Practice*. Systems Programming Series. Addison-Wesley, 1996.
- [56] J.E. Freud and C.F.J. Williams. *Dictionary/Outline of Basic Statistics*. Dover books on advanced mathematics. Dover, 1966.
- [57] G. Friedland, K. Jantz, and R. Rojas. SIOX: Simple interactive object extraction in still images. In *Multimedia, Seventh IEEE International Symposium on*, pages 253–260. IEEE, December 2005.
- [58] T. Fu and H. Foroosh. Expression morphing from distant viewpoints. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 5, pages 3519–3522. IEEE, 2004.
- [59] B. Furht. *Encyclopedia of Multimedia*. Springer Reference. Springer, 2008.
- [60] W.P. Gardiner. *Statistical Analysis Methods for Chemists: A Software Based Approach*. Royal Society of Chemistry, 1997.
- [61] L. Garrido. *Hierarchical region based processing of images and video sequences: application to filtering, segmentation and information retrieval*. PhD thesis, Department of Signal theory and Communications - Universitat Politècnica de Catalunya, April 2002.
- [62] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361, Providence, USA, June 2012. IEEE.
- [63] Theo Gevers and Arnold W.M. Smeulders. Color based object recognition. *Pattern recognition*, 32(3):453–464, 1999.

- [64] T.A. Goh, R. Williamson, and G. Buqué. *Advancing Maths for AQA 2nd Edition*. Heinemann Educational Books, 2004.
- [65] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall Press, ISBN 0-201-18075-8, 2002.
- [66] J. Gosling. *Quicksmart Introductory Statistics*. Quicksmart University Guides Series. Pascal Press, 1995.
- [67] John Goutsias and Henk J.A.M. Heijmans. Fundamenta morphologicae mathematicae. *Fundamenta Informaticae*, 41(1):1–31, 2000.
- [68] John Goutsias, Luc Vincent, and Dan S Bloomberg. *Mathematical morphology and its applications to image and signal processing*, volume 18. Springer, 2000.
- [69] D. Green, M. Ervine, and S. White. *Fundamentals of Perioperative Management*. Greenwich Medical Media, 2002.
- [70] G.J. Grevera and J.K. Udupa. An objective comparison of 3-D image interpolation methods. *Medical Imaging, IEEE Transactions on*, 17(4):642–652, 1998.
- [71] M. Grundland, R. Vohra, G.P. Williams, and N.A. Dodgson. Cross dissolve without cross fade: Preserving contrast, color and salience in image compositing. In *Computer graphics forum*, volume 25, pages 577–586. Wiley Online Library, 2006.
- [72] S. Guha. *Computer Graphics Through OpenGL: From Theory to Experiments*. Chapman & Hall/CRC computer graphics, geometric modeling, and animation series. CRC Press, 2010.
- [73] Z. Guo and R.W. Hall. Parallel thinning with two-subiteration algorithms. *Communications of the ACM*, 32(3):359–373, 1989.
- [74] A. Hanbury and J. Serra. Colour image analysis in 3d-polar coordinates. *Pattern Recognition*, pages 124–131, 2003.
- [75] Donncha Hanna and Martin Dempster. *Psychology Statistics For Dummies*. Wiley, 2012.
- [76] A. Hart. *Making Sense of Statistics in Healthcare*. Radcliffe Series. Radcliffe Medical Press, 2001.
- [77] Frank Heckel, Olaf Konrad, Horst Karl Hahn, and Heinz-Otto Peitgen. Interactive 3D medical image segmentation with energy-minimizing implicit functions. *Computers & Graphics*, 35(2):275–287, 2011.
- [78] G. Helander, T.K. Landauer, and P.V. Prabhu. *Handbook of Human-Computer Interaction*. Elsevier Science, 1997.

- [79] Y.S. Ho and H.J. Kim. *Advances in Multimedia Information Processing - PCM 2005: 6th Pacific Rim Conference on Multimedia, Jeju Island, Korea, November 11-13, 2005, Proceedings*. Lecture Notes in Computer Science. Springer, 2005.
- [80] A. Hornberg. *Handbook of Machine Vision*. John Wiley & Sons, 2007.
- [81] M.C. Hung and Y.H. Wu. Mapping and visualizing the Great Salt Lake landscape dynamics using multi-temporal satellite images, 1972–1996. *International Journal of Remote Sensing*, 26(9):1815–1834, 2005.
- [82] K.A. Hunt. *The Art of Image Processing With Java*. A K Peters, 2010.
- [83] Recommendation BT.601-7 ITU-R. Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios. Technical report, International Telecommunication Union, March 2011.
- [84] Recommendation BT.709-5 ITU-R. Parameter values for the HDTV standards for production and international programme exchange. Technical report, International Telecommunication Union, April 2002.
- [85] M. Iwanowski and J. Serra. Morphological interpolation and color images. In *Image Analysis and Processing, 1999. Proceedings. International Conference on*, pages 50–55. IEEE, 1999.
- [86] Marcin Iwanowski. Generalized Morphological Mosaic Interpolation and Its Application to Computer-Aided Animations. In Wladyslaw Skarbek, editor, *Computer Analysis of Images and Patterns*, volume 2124 of *Lecture Notes in Computer Science*, pages 493–501. Springer Berlin / Heidelberg, 2001. 10.1007/3-540-44692-3_60.
- [87] Marcin Iwanowski and Jean Serra. The Morphological-Affine Object Deformation. In John Goutsias, Luc Vincent, and Dan S. Bloomberg, editors, *Mathematical Morphology and its Applications to Image and Signal Processing*, volume 18 of *Computational Imaging and Vision*, pages 81–90. Kluwer Academic Publishers, 2002. 10.1007/0-306-47025-X_10.
- [88] K. Jack. *Video Demystified: A Handbook for the Digital Engineer*. Demystifying technology series. Elsevier Science, 2007.
- [89] B. Jähne. *Digital Image Processing*. Springer, 2005.
- [90] T.R. Jain and A.S. Sandhu. *Quantitative Methods*. Vk Publications, 2009.
- [91] M.E. James and University of Phoenix. *An Empirical Investigation Into the Extent of Quality Management Practices in the Jamaican Manufacturing Industry*. University of Phoenix, 2009.

- [92] S. Jayaraman, S. Esakkirajan, and T. Veerakumar. *Digital Image Processing*. Tata Mc Graw Hill Education Private, 2009.
- [93] Jessica M. Utts and Robert F. Heckard. *Statistical Ideas And Methods*. Brooks/Cole, 2005.
- [94] C.V. Jones. *Visualization and Optimization*. Operations Research/Computer Science Interfaces Series. Springer, 1996.
- [95] P. Kakar and N. Sudha. Exposing Postprocessed Copy–Paste Forgeries Through Transform-Invariant Features. *Information Forensics and Security, IEEE Transactions on*, 7(3):1018–1028, 2012.
- [96] C. Kamath. *Scientific Data Mining: A Practical Perspective*. Society for Industrial and Applied Mathematics, 2009.
- [97] Arie Kaufman and Klaus Mueller. Overview of volume rendering. *The visualization handbook*, pages 127–174, 2005.
- [98] R. Kimmel, D. Shaked, N. Kiryati, and A.M. Bruckstein. Skeletonization via distance maps and level sets. *Computer Vision and Image Understanding*, 62(3):382–391, 1995.
- [99] R.E. Kirk. *Statistics: An Introduction*. Thomson/Wadsworth, 2007.
- [100] C.M. Kitchen Ramirez. Nonparametric versus parametric tests of location in biomedical research. *American journal of ophthalmology*, 147(4):571, 2009.
- [101] B. Klava and N.S.T. Hirata. Interactive image segmentation with integrated use of the markers and the hierarchical watershed approaches. In *International Conference on Computer Vision Theory and Applications-VISSAPP*, pages 186–193, 2009.
- [102] A. Kopra. *Writing Mental Ray Shaders: A Perceptual Introduction*. Mental ray handbooks. Springer, 2008.
- [103] A. Koschan and M.A. Abidi. *Digital Color Image Processing*. Wiley-Interscience, 2008.
- [104] N.T. Kottegoda and R. Rosso. *Applied Statistics for Civil and Environmental Engineers*. Wiley, 2009.
- [105] D.J. Kroon. Numerical optimization of kernel based image derivatives. 2009.
- [106] A.S. Krylov, A.V. Nasonov, and A.A. Chernomorets. Combined linear resampling method with ringing control. In *Proceedings of GraphiCon*, volume 2009, pages 163–165, 2009.

- [107] C. Kufs. *Stats with Cats: The Domesticated Guide to Statistics, Models, Graphs, and Other Breeds of Data Analysis*. Wheatmark, 2011.
- [108] L. Lam, S.W. Lee, and C.Y. Suen. Thinning methodologies-a comprehensive survey. *IEEE Transactions on pattern analysis and machine intelligence*, 14(9):869–885, 1992.
- [109] D. Larlus, J. Verbeek, and F. Jurie. Category level object segmentation by combining bag-of-words models with dirichlet processes and random fields. *International journal of computer vision*, 88(2):238–253, 2010.
- [110] J. Larson-Hall. *A Guide to Doing Statistics in Second Language Research Using SPSS*. Second Language Acquisition Research. Routledge, 2010.
- [111] Tong-Yee Lee and Wen-Hsiu Wang. Morphology-Based Three-Dimensional Interpolation. *IEEE Transactions on Medical Imaging*, 19(7):711–721, July 2000.
- [112] T.M. Lehmann, C. Gonner, and K. Spitzer. Survey: Interpolation methods in medical image processing. *Medical Imaging, IEEE Transactions on*, 18(11):1049–1075, 1999.
- [113] W.C. Leung. *Statistics and Evidence Based Medicine for Exams*. Petroc Press, 2001.
- [114] O. Lezoray, C. Meurie, and A. Elmoataz. A graph approach to color mathematical morphology. In *Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology, 2005.*, pages 856–861. IEEE, 2005.
- [115] R.S. Llopis. *Problemas Resueltos de Teoría de Sistemas*. Treballs d’Informàtica i Tecnologia. Universitat Jaume I, 2002.
- [116] R. Lotufo and W. Silva. Minimal set of markers for the watershed transform. In *Proceedings of ISMM*, volume 2002, pages 359–368, 2002.
- [117] Robert M. Loughheed, David L. McCubbrey, and Ramesh Jain. *Applying Iconic Processing in Machine Vision*. Springer Series in Perception Engineering. Springer, 1988.
- [118] Jitendra Malik, Serge Belongie, Thomas Leung, and Jianbo Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 43(1):7–27, 2001.
- [119] S. Marchand-Maillet and Y.M. Sharaiha. *Binary Digital Image Processing: A Discrete Approach*. Electronics & Electrical. Elsevier Science, 1999.

- [120] J.S. Marques, N.P. de la Blanca, and P. Pina. *Pattern Recognition and Image Analysis: Second Iberian Conference, IbPRIA 2005, Estoril, Portugal, June 7-9, 2005, Proceeding*. Lecture Notes in Computer Science. Springer, 2005.
- [121] O. Marques. *Practical Image and Video Processing Using MATLAB*. Wiley, 2011.
- [122] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.
- [123] R.L. Mason, R.F. Gunst, and J.L. Hess. *Statistical Design and Analysis of Experiments: With Applications to Engineering and Science*. Wiley Series in Probability and Statistics - Applied Probability and Statistics Section Series. Wiley, 2003.
- [124] R. Matignon. *Neural Network Modeling Using Sas Enterprise Miner*. AuthorHouse, 2005.
- [125] J.J. McConnell. *Computer Graphics: Theory Into Practice*. Jones and Bartlett Publishers, 2006.
- [126] K. McGuinness and N.E. O'Connor. A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition*, 43(2):434–444, 2010.
- [127] K. McGuinness and N.E. O'Connor. Toward automated evaluation of interactive segmentation. *Computer Vision and Image Understanding*, 115(6):868–884, 2011.
- [128] E. Meijering. A chronology of interpolation: From ancient astronomy to modern signal and image processing. *Proceedings of the IEEE*, 90(3):319–342, 2002.
- [129] William Mendenhall, Robert J. Beaver, and Barbara M. Beaver. *Introduction to Probability & Statistics*. Available 2010 Titles Enhanced Web Assign Series. Brooks/Cole, 2009.
- [130] Fernand Meyer. A morphological interpolation method for mosaic images. In Petros A. Maragos, Ronald W. Schafer, and Muhammad Akmal Butt, editors, *Mathematical morphology and its applications to image and signal processing*, volume 5 of *Computational Imaging and Vision*, pages 337–344. Kluwer Academic Publishers, 1996.
- [131] P. Meylan, A.C. Favre, and A. Musy. *Predictive Hydrology: A Frequency Analysis Approach*. Taylor & Francis Group, 2012.

- [132] B. Migeon, R. Charreyron, P. Deforge, and P. Marche. Improvement of morphology-based interpolation. In *Engineering in Medicine and Biology Society, 1998. Proceedings of the 20th Annual International Conference of the IEEE*, volume 2, pages 585–587. IEEE, 1998.
- [133] A. Mishra. Active Segmentation: A New Approach. In C.W. Tyler, editor, *Computer Vision: From Surfaces to 3D Objects*, pages 25–49. CRC Press, 2011.
- [134] M.L. Mitchell and J.M. Jolley. *Research Design Explained*. Wadsworth/Cengage Learning, 2012.
- [135] M.K. Monaco. *Color Space Analysis for Iris Recognition*. ProQuest, 2007.
- [136] S. Montabone. *Beginning Digital Image Processing: Using Free Tools for Photographers*. Apresspod Series. Apress, 2010.
- [137] D.C. Montgomery and G.C. Runger. *Applied Statistics and Probability for Engineers*. John Wiley & Sons, 2010.
- [138] K. Murphy, B. Myors, and A. Wolach. *Statistical Power Analysis: A Simple and General Model for Traditional and Modern Hypothesis Tests, Third Edition*. Taylor & Francis, 2011.
- [139] S. Nagabhushana. *Computer Vision And Image Processing*. New Age International, 2006.
- [140] L. Najman and H. Talbot. *Mathematical Morphology*. Wiley, 2013.
- [141] T.V. Nguyen, N. Pham, T. Tran, and B. Le. Higher Order Conditional Random Field for Multi-Label Interactive Image Segmentation. In *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2012 IEEE RIVF International Conference on*, pages 1–4. IEEE, 2012.
- [142] C. Nieuwenhuis, E. Töppe, and D. Cremers. Space-varying color distributions for interactive multiregion segmentation: Discrete versus continuous approaches. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 177–190. Springer, 2011.
- [143] A.M. Noll. *Principles of Modern Communications Technology*. Artech House Telecommunications Library. Artech House, 2001.
- [144] A. Obuchowicz, M. Hrebień, T. Nieczkowski, and A. Marciniak. Computational intelligence techniques in image segmentation for cytopathology. *Computational intelligence in biomedicine and bioinformatics*, pages 169–199, 2008.
- [145] L. O’Gorman, M.J. Sammon, and M. Seul. *Practical Algorithms for Image Analysis with CD-ROM*. Cambridge University Press, 2008.

- [146] J. Ohser and K. Schladitz. *3D Images of Materials Structures: Processing and Analysis*. Wiley, 2009.
- [147] S.D. Olabarriaga and Arnold W.M. Smeulders. Interaction in the segmentation of medical images: A survey. *Medical image analysis*, 5(2):127–142, 2001.
- [148] American Congress on Surveying, Mapping, A.C.S.M. Meeting, American Society for Photogrammetry, and Remote Sensing. *Technical Papers, ... ACSM-ASPRS Annual Convention*. Technical papers. American Congress on Surveying and Mapping, 1987.
- [149] J. O’Rourke. *Computational Geometry in C*. Cambridge University Press, 1998.
- [150] K. Palágyi, J. Tschirren, and M. Sonka. Quantitative analysis of intrathoracic airway trees: methods and validation. In *Information Processing in Medical Imaging*, pages 222–233. Springer, 2003.
- [151] S. Patnaik and Y.M. Yang. *Soft Computing Techniques in Vision Science*, volume 395. Springer, 2012.
- [152] C.Y.J. Peng. *Data Analysis Using SAS*. SAGE Publications, 2008.
- [153] Martino Pesaresi and Jon Atli Benediktsson. Image segmentation based on the derivative of the morphological profile. In J. Goutsias, L. Vincent, and D.S. Bloomberg, editors, *Mathematical morphology and its applications to image and signal processing*, volume 18 of *Computational Imaging and Vision*, pages 179–188. Kluwer Academic Publishers, 2002. 10.1007/0-306-47025-X.10.
- [154] K.N. Plataniotis and A.N. Venetsanopoulos. *Color Image Processing and Applications*. Digital Signal Processing. Springer, 2000.
- [155] N. Ponomarenko, V. Lukin, A. Zelensky, K. Egiazarian, M. Carli, and F. Battisti. TID2008-a database for evaluation of full-reference visual quality assessment metrics. *Advances of Modern Radioelectronics*, 10(10):30–45, 2009.
- [156] C.A. Poynton. *Digital Video and HDTV: Algorithms and Interfaces*. Morgan Kaufmann, 2003.
- [157] W.K. Pratt. *Digital Image Processing: PIKS Scientific Inside*. Wiley-Interscience publication. John Wiley & Sons, 2007.
- [158] B. Preim and D. Bartz. *Visualization in Medicine: Theory, Algorithms, and Applications*. The Morgan Kaufmann Series in Computer Graphics. Elsevier Science, 2007.
- [159] Cambridge University Press. *Diccionario Bilingue Cambridge Spanish-English Paperback Compact edition*. Cambridge University Press, 2008.

- [160] S. Qureshi. *Embedded Image Processing on the TMS320C6000TM DSP: Examples in Code Composer StudioTM and MATLAB*. Springer, 2005.
- [161] K.M. Ramachandran and C.P. Tsokos. *Mathematical Statistics with Applications*. Elsevier Science, 2009.
- [162] T.A. Ramstad, S.O. Aase, and J.H. Husøy. *Subband Compression of Images: Principles and Examples*. Advances in image communication. Elsevier, 1995.
- [163] Nornadiah Mohd Razali and Yap Bee Wah. Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. *Journal of Statistical Modeling and Analytics Vol, 2(1):21–33*, 2011.
- [164] A. Rial Boubeta and J. Varela Mallou. *Estadística práctica para la investigación en ciencias de la salud*. Catálogo General. Netbiblo, 2008.
- [165] A. Rodney. *Color Management for Photographers: Hands on Techniques for Photoshop Users*. Taylor & Francis, 2012.
- [166] Jos B.T.M. Roerdink and Arnold Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta Informaticae*, 41(1-2):187–228, 2000.
- [167] Rosenthal, James A. *Statistics and Data Interpretation for Social Work*. Springer Series. Springer Publishing Company, 2011.
- [168] A. Rubin. *Statistics for Evidence-based Practice and Evaluation*. Research, Statistics, and Program Evaluation Series. BROOKS COLE Publishing Company, 2012.
- [169] D. Rumsey-Johnson. *Intermediate Statistics For Dummies*. For Dummies. John Wiley & Sons, 2007.
- [170] D. Ruppert. *Statistics and Finance: An Introduction*. Springer Texts in Statistics. Springer, 2004.
- [171] Tomoya Sakai and Atsushi Imiya. Validation of Watershed Regions by Scale-Space Statistics. In Xue-Cheng Tai, Knut Morken, Marius Lysaker, and Knut-Andreas Lie, editors, *Scale Space and Variational Methods in Computer Vision: Second International Conference, SSVN 2009, June 1-5, 2009*, volume 5567. Springer-Verlag New York Inc, 2009.
- [172] P. Salembier and L. Garrido. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *Image Processing, IEEE Transactions on*, 9(4):561–576, 2000.

- [173] Jakob Santner, Thomas Pock, and Horst Bischof. Interactive multi-label segmentation. In *Proceedings 10th Asian Conference on Computer Vision (ACCV), Queenstown, New Zealand*, pages 397–410. Springer, November 2010.
- [174] Masaharu Sato, Dorin Gutu, and Yuukou Horita. A new image quality assessment model based on the MPEG-7 descriptor. In *Advances in Multimedia Information Processing-PCM 2010*, pages 159–170. Springer, 2010.
- [175] S.D. Schlotzhauer. *Elementary Statistics Using JMP*. SAS Publishing, 2007.
- [176] R.A. Schowengerdt. *Remote Sensing: Models and Methods for Image Processing*. Elsevier Science, 2006.
- [177] H.G. Senel. Image Gradient Estimation with Wide Support Kernels. In *Information and Automation for Sustainability, 2008. ICIAFS 2008. 4th International Conference on*, pages 132–137. IEEE, 2008.
- [178] Jean Serra. Hausdorff distances and interpolations. In *Proceedings of the fourth international symposium on Mathematical morphology and its applications to image and signal processing, ISMM '98*, pages 107–114, Norwell, MA, USA, 1998. Kluwer Academic Publishers.
- [179] J.P. Serra. *Image Analysis and Mathematical Morphology*. Academic Press (London and New York), 1982.
- [180] K. Seshadrinathan, T.N. Pappas, R.J. Safranek, Junqing Chen, Zhou Wang, Hamid R. Sheikh, and Alan C. Bovik. Image quality assessment. In Al Bovik, editor, *The essential guide to image processing*, chapter 21, pages 553–595. Elsevier, 2009.
- [181] G. Sharma. *Digital Color Imaging Handbook*. CRC Press, Boca Raton, FL, 2003.
- [182] H.R. Sheikh, M.F. Sabir, and A.C. Bovik. A statistical evaluation of recent full reference image quality assessment algorithms. *Image Processing, IEEE Transactions on*, 15(11):3440–3451, 2006.
- [183] R. Shekhar, V. Walimbe, and W. Plishker. Medical Image Processing. *Handbook of Signal Processing Systems*, pages 213–242, 2010.
- [184] F.Y. Shih. *Image Processing and Mathematical Morphology: Fundamentals and Applications*. CRC Press, 2009.
- [185] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. *Computer Vision-ECCV 2006*, pages 1–15, 2006.

- [186] Hedvig Sidenbladh. *Probabilistic tracking and reconstruction of 3d human motion in monocular video sequences*. PhD thesis, Dept. of Numerical Analysis and Computer Science, 2001.
- [187] J. Sim and C.C. Wright. *Research in Health Care: Designs and Methods*. Nelson Thornes, 2000.
- [188] A.K. Sinop and L. Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.
- [189] P. Soille. *Morphological Image Analysis: Principles and Applications*. 2003. Springer-Verlag, Berlin, 2003.
- [190] M. Sternstein. *Barron's AP Statistics, 5th Ed.* Barron's AP Statistics. Barron's Educational Series, 2010.
- [191] A. Stewart. *Basic Statistics and Epidemiology: A Practical Guide*. Radcliffe Pub., 2010.
- [192] Gilbert Strang. *Calculus*, volume 1. Wellesley-Cambridge Press, 1991.
- [193] M. Studeny. *On Probabilistic Conditional Independence Structures*. Information Science and Statistics. Springer-Verlag London Limited, 2005.
- [194] Peter Sussner and Estevão Laureano Esmi. Constructive Morphological Neural Networks: Some Theoretical Aspects and Experimental Results in Classification. In Leonardo Franco, David A Elizondo, and José M Jerez, editors, *Constructive Neural Networks*, volume 258 of *Studies in Computational Intelligence*, pages 123–144. Springer Berlin Heidelberg.
- [195] M. Swiercz and Marcin Iwanowski. Image features based on morphological class distribution functions and its application to binary pattern recognition. *Przegląd Elektrotechniczny*, 88(2), February 2012.
- [196] Takuma Terada, Takayuki Fukui, Takanori Igarashi, Keisuke Nakao, Akio Kashimoto, and Yen-Wei Chen. Automatic Facial Image Manipulation System and Facial Texture Analysis. In *Natural Computation, 2009. ICNC'09. Fifth International Conference on*, volume 6, pages 8–12. IEEE, 2009.
- [197] K. Tingelhoff, K.W.G. Eichhorn, I. Wagner, M.E. Kunkel, A.I. Moral, M.E. Rilk, F.M. Wahl, and F. Bootz. Analysis of manual segmentation in paranasal CT images. *European Archives of Oto-Rhino-Laryngology*, 265(9):1061–1070, 2008.

- [198] J. Toriwaki and H. Yoshida. *Fundamentals of Three-dimensional Digital Image Processing*. Springer, 2009.
- [199] F. Torres, R. Marfil, and A. Bandera. 3D Image Segmentation Using the Bounded Irregular Pyramid. In *Computer Analysis of Images and Patterns*, pages 979–986. Springer, 2009.
- [200] J. Townend. *Practical Statistics for Environmental and Biological Scientists*. Wiley, 2012.
- [201] Jayaram K. Udupa, Vicki R. LaBlanc, Hilary Schmidt, Celina Imielinska, Punam K. Saha, George J. Grevera, Ying Zhuge, L.M. Currie, Pat Molholt, and Yinpeng Jin. A methodology for evaluating image segmentation algorithms. In M. Sonka, The Society of Photo-Optical Instrumentation Engineers, J.M. Fitzpatrick, and A.A.P. Medicine, editors, *Medical Imaging: Image processing*, volume 4684 of *Proceedings of SPIE, the International Society for Optical Engineering*, pages 266–277. International Society for Optics and Photonics, SPIE, 2002.
- [202] G. van Belle, L.D. Fisher, P.J. Heagerty, and T. Lumley. *Biostatistics: A Methodology For the Health Sciences*. Wiley Series in Probability and Statistics. Wiley, 2004.
- [203] Jaume Vergés Llahí. *Color Constancy and Image Segmentation Techniques for Applications to Mobile Robotics*. PhD thesis, Departament d’Enginyeria de Sistemes, Automàtica i Informàtica Industrial, 2005.
- [204] J.P. Verma. *Data Analysis in Management with SPSS Software*. Springer, 2013.
- [205] V. Vezhnevets, V. Sazonov, and A. Andreeva. A survey on pixel-based skin color detection techniques. In *Proc. Graphicon*, volume 3, pages 85–92. Moscow, Russia, 2003.
- [206] Javier Vidal. Procesar.m algorithm. June 2006.
- [207] Javier Vidal. *Interpolación de formas en imágenes usando morfología matemática*. PhD thesis, Facultad de Informática (Universidad Politécnica de Madrid), September 2008.
- [208] Javier Vidal, José Crespo, and Víctor Maojo. Inclusion Relationships and Homotopy Issues in Shape Interpolation for Binary Images. In Eric Andres, Guillaume Damiand, and Pascal Lienhardt, editors, *Discrete Geometry for Computer Imagery*, volume 3429 of *Lecture Notes in Computer Science*, pages 206–215. Springer Berlin / Heidelberg, 2005. 10.1007/978-3-540-31965-8_20.

- [209] Javier Vidal, José Crespo, and Víctor Maojo. Recursive Interpolation Technique for Binary Images Based on Morphological Median Sets. In Christian Ronse, Laurent Najman, and Etienne Decencière, editors, *Mathematical Morphology: 40 Years On*, volume 30 of *Computational Imaging and Vision*, pages 53–62. Springer Netherlands, 2005. 10.1007/1-4020-3443-1_6.
- [210] Javier Vidal, José Crespo, and Víctor Maojo. A region-based interpolation method for mosaic images. In Gerald Jean Francis Banon, Junior Barrera, Ulisses de Mendonça Braga-Neto, and Nina Sumiko Tomita Hirata, editors, *Proceedings of the eight International Symposium on Mathematical Morphology*, volume 1, pages 201–212, São José dos Campos, October 10–13, 2007 2007. Universidade de São Paulo (USP), Instituto Nacional de Pesquisas Espaciais (INPE).
- [211] L. Vincent. Efficient computation of various types of skeletons. *Medical Imaging V: Image Processing. Volume SPIE-1445*, pages 297–311, 1991.
- [212] G.G. Vining and S.M. Kowalski. *Statistical Methods for Engineers*. Cengage Learning, 2010.
- [213] Glenn A. Walker and Jack Shostak. *Common Statistical Methods for Clinical Research with SAS Examples, : Third Edition*. SAS Institute Inc., 2010.
- [214] H. Wang and P. Yushkevich. Guiding Automatic Segmentation with Multiple Manual Segmentations. *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2012*, pages 429–436, 2012.
- [215] Wei-Jing Wang and Jia-Xin Chen. An Improved Interpolation of Medical Images. *Journal of Henan University of Science & Technology (Natural Science)*, 3:008, 2006.
- [216] Zhou Wang and Alan Conrad Bovik. *Modern Image Quality Assessment*. Synthesis lectures on image, video, and multimedia processing. Morgan & Claypool Publishers, 2006.
- [217] Zhou Wang, Alan Conrad Bovik, Hamid R. Sheikh, and E.P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, 2004.
- [218] T.J. Watsham and K. Parramore. *Quantitative Methods for Finance*. International Thomson Business Press, 1997.
- [219] D. Weiler, V. Willert, J. Eggert, and E. Korner. A probabilistic method for motion pattern segmentation. In *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*, pages 1645–1650. IEEE, 2007.

- [220] S.L. Weinberg and S.K. Abramowitz. *Data Analysis for the Behavioral Sciences Using Spss*. University Press, 2002.
- [221] D. Weisburd and C. Britt. *Statistics in Criminal Justice*. Springer, 2007.
- [222] P.F. Whelan and D. Molloy. *Machine Vision Algorithms in Java: Techniques and Implementation*. Springer, 2001.
- [223] H.F. Wilkinson and F. Schut. *Digital Image Analysis of Microbes: Imaging, Morphometry, Fluorometry and Motility Techniques and Applications*. Modern Microbiological Methods. Wiley, 1998.
- [224] G. Wolberg. Recent advances in image morphing. In *Computer Graphics International, 1996. Proceedings*, pages 64–71. IEEE, 1996.
- [225] Q. Wu, F. Merchant, and K. Castleman. *Microscope Image Processing*. Elsevier Science, 2010.
- [226] Q. Wu, F. Merchant, and K.R. Castleman. *Microscope Image Processing*. Academic Press. Elsevier/Academic Press, 2008.
- [227] H. Xiong and W.B. Lee. *Knowledge Science, Engineering and Management: 5th International Conference, KSEM 2011, Irvine, CA, USA, December 12-14, 2011. Proceedings*. Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence. Springer, 2012.
- [228] A. Yadav and P. Yadav. *Digital Image Processing*. Laxmi Publications Pvt Limited, 2009.
- [229] C.C. Yang and S.H. Kwok. Efficient gamut clipping for color image processing using LHS and YIQ. *Optical Engineering*, 42(3):701–711, 2003.
- [230] W. Yang, J. Cai, J. Zheng, and J. Luo. User-friendly interactive image segmentation through unified combinatorial user inputs. *Image Processing, IEEE Transactions on*, 19(9):2470–2479, 2010.
- [231] Eugen Zaharescu. Analysis of Morphology-like Operators Used in Color Image Contrast Enhancement. In *Proc. of International Conference on Theory and Applications of Mathematics and Informatics-ICTAMI*, 2004.
- [232] L. Zhang and Q. Ji. A Bayesian network model for automatic and interactive image segmentation. *Image Processing, IEEE Transactions on*, 20(9):2582–2593, 2011.
- [233] Lin Zhang, Lei Zhang, Xuanqin Mou, and David Zhang. A comprehensive evaluation of full reference image quality assessment algorithms. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*. IEEE, 2012.

- [234] Lin Zhang, Lei Zhang, Xuanquin Mou, and David Zhang. FSIM: A feature similarity index for image quality assessment. *Image Processing, IEEE Transactions on*, (8):2378–2386, August 2011.
- [235] W. Zhang, P. Yan, and X. Li. Estimating patient-specific shape prior for medical image segmentation. In *Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on*, pages 1451–1454. IEEE, 2011.
- [236] Y.J. Zhang. An overview of image and video segmentation in the last 40 years. *Advances in Image and Video Segmentation*, pages 1–15, 2006.
- [237] C. Zhou and C. Liu. Weakly Supervised Semi-automatic Semantic Segmentation of Natural Scene Images. *Journal of Computational Information Systems*, 8(18):7757–7763, 2012.
- [238] R. Zhu and Y. Ma. *Information Engineering and Applications: International Conference on Information Engineering and Applications (IEA 2011)*. Lecture Notes in Electrical Engineering. Springer, 2011.

