Handbook of Computer Vision Algorithms in Image Algebra

Gerhard X. Ritter
Joseph N. Wilson

Center for Computer Vision and Visualization
University of Florida

Preface

The aim of this book is to acquaint engineers, scientists, and students with the basic concepts of image algebra and its use in the concise representation of computer vision algorithms. In order to achieve this goal we provide a brief survey of commonly used computer vision algorithms that we believe represents a core of knowledge that all computer vision practitioners should have. This survey is not meant to be an encyclopedic summary of computer vision techniques as it is impossible to do justice to the scope and depth of the rapidly expanding field of computer vision.

The arrangement of the book is such that it can serve as a reference for computer vision algorithm developers in general as well as for algorithm developers using the image algebra C++ object library, `iac++`.[1] The techniques and algorithms presented in a given chapter follow a progression of increasing abstractness. Each technique is introduced by way of a brief discussion of its purpose and methodology. Since the intent of this text is to train the practitioner in formulating his algorithms and ideas in the succinct mathematical language provided by image algebra, an effort has been made to provide the precise mathematical formulation of each methodology. Thus, we suspect that practicing engineers and scientists will find this presentation somewhat more practical and perhaps a bit less esoteric than those found in research publications or various textbooks paraphrasing these publications.

Chapter 1 provides a short introduction to field of image algebra. Chapters 2–11 are devoted to particular techniques commonly used in computer vision algorithm development, ranging from early processing techniques to such higher level topics as image descriptors and artificial neural networks. Although the chapters on techniques are most naturally studied in succession, they are not tightly interdependent and can be studied according to the reader's particular interest. In the Appendix we present `iac++` computer programs of some of the techniques surveyed in this book. These programs reflect the image algebra pseudocode presented in the chapters and serve as examples of how image algebra pseudocode can be converted into efficient computer programs.

[1] The `iac++` library supports the use of image algebra in the C++ programming language and is available for anonymous ftp from `ftp://ftp.cis.ufl.edu/pub/src/ia/`.

Acknowledgments

We wish to take this opportunity to express our thanks to our current and former students who have, in various ways, assisted in the preparation of this text. In particular, we wish to extend our appreciation to Dr. Paul Gader, Dr. Jennifer Davidson, Dr. Hongchi Shi, Ms. Brigitte Pracht, Mr. Mark Schmalz, Mr. Venugopal Subramaniam, Mr. Mike Rowlee, Dr. Dong Li, Dr. Huixia Zhu, Ms. Chuanxue Wang, Mr. Jaime Zapata, and Mr. Liang-Ming Chen. We are most deeply indebted to Dr. David Patching who assisted in the preparation of the text and contributed to the material by developing examples that enhanced the algorithmic exposition. Special thanks are due to Mr. Ralph Jackson, who skillfully implemented many of the algorithms herein, and to Mr. Robert Forsman, the primary implementor of the `iac++` library.

We also want to express our gratitude to the Air Force Wright Laboratory for their encouragement and continuous support of image algebra research and development. This book would not have been written without the vision and support provided by numerous scientists of the Wright Laboratory at Eglin Air Force Base in Florida. These supporters include Dr. Lawrence Ankeney who started it all, Dr. Sam Lambert who championed the image algebra project since its inception, Mr. Neil Urquhart our first program manager, Ms. Karen Norris, and most especially Dr. Patrick Coffield who persuaded us to turn a technical report on computer vision algorithms in image algebra into this book.

Last but not least we would like to thank Dr. Robert Lyjack of ERIM for his friendship and enthusiastic support during the formative stages of image algebra.

Notation

The tables presented here provide a brief explantation of the notation used throughout this document. The reader is referred to Ritter [1] for a comprehensive treatise covering the mathematics of image algebra.

Logic

Symbol	Explanation
$p \Rightarrow q$	"p implies q." If p is true, then q is true.
$p \Leftrightarrow q$	"p if and only if q," which means that p and q are logically equivalent.
iff	"if and only if"
$\neg$	"not"
$\exists$	"there exists"
$\nexists$	"there does not exist"
$\forall$	"for each"
s.t.	"such that"

Sets Theoretic Notation and Operations

Symbol	Explanation
X, Y, Z	Uppercase characters represent arbitrary sets.
x, y, z	Lowercase characters represent elements of an arbitrary set.
$\mathbf{X}, \mathbf{Y}, \mathbf{Z}$	Bold, uppercase characters are used to represent point sets.
$\mathbf{x}, \mathbf{y}, \mathbf{z}$	Bold, lowercase characters are used to represent points, i.e., elements of point sets.
$\mathbb{N}$	The set $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.
$\mathbb{Z}, \mathbb{Z}^+, \mathbb{Z}^-$	The set of integers, positive integers, and negative integers.
$\mathbb{Z}_n$	The set $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$.
$\mathbb{Z}_n^+$	The set $\mathbb{Z}_n^+ = \{1, 2, \ldots, n\}$.
$\mathbb{Z}_{\pm n}$	The set $\mathbb{Z}_{\pm n} = \{-n+1, \ldots, -1, 0, 1, \ldots, n-1\}$.
$\mathbb{R}, \mathbb{R}^+, \mathbb{R}^-, \mathbb{R}^{\geq 0}$	The set of real numbers, positive real numbers, negative real numbers, and positive real numbers including 0.

Symbol	Explanation
$\mathbb{C}$	The set of complex numbers.
$\mathbb{F}$	An arbitrary set of values.
$\mathbb{F}_\infty$	The set $\mathbb{F}$ unioned with $\{\infty\}$.
$\mathbb{F}_{-\infty}$	The set $\mathbb{F}$ unioned with $\{\infty\}$.
$\mathbb{F}_{\pm\infty}$	The set $\mathbb{F}$ unioned with $\{-\infty, \infty\}$.
$\varnothing$	The *empty set* (the set that has no elements).
2^X	The *power set* of X (the set of all subsets of X).
$\in$	"is an element of"
$\notin$	"is not an element of"
$\subset$	"is a subset of"
$X \bigcup Y$	*Union* $X \cup Y = \{z : z \in X \text{ or } z \in Y\}$
$\bigcup\limits_{\lambda\in\Lambda} X_\lambda$	Let $\{X_\lambda\}_{\lambda\in\Lambda}$ be a family of sets indexed by an indexing set Λ. $\bigcup\limits_{\lambda\in\Lambda} X_\lambda = \{x : x \in X_\lambda \text{ for at least one } \lambda \in \Lambda\}$
$\bigcup\limits_{i=1}^{n} X_i$	$\bigcup\limits_{i=1}^{n} X_i = X_1 \cup X_2 \cup \ldots \cup X_n$
$\bigcup\limits_{i=1}^{\infty} X_i$	$\bigcup\limits_{i=1}^{\infty} X_i = \{x : x \in X_i \text{ for some } i \in \mathbb{Z}_\infty^+\}$
$X \bigcap Y$	*Intersection* $X \cap Y = \{z : z \in X \text{ and } z \in Y\}$
$\bigcap\limits_{\lambda\in\Lambda} X_\lambda$	Let $\{X_\lambda\}_{\lambda\in\Lambda}$ be a family of sets indexed by an indexing set Λ. $\bigcap\limits_{\lambda\in\Lambda} X_\lambda = \{x : x \in X_\lambda \text{ for all } \lambda \in \Lambda\}$
$\bigcap\limits_{i=1}^{n} X_i$	$\bigcap\limits_{i=1}^{n} X_i = X_1 \cap X_2 \cap \ldots \cap X_n$
$\bigcap\limits_{i=1}^{\infty} X_i$	$\bigcap\limits_{i=1}^{\infty} X_i = \{x : x \in X_i \text{ for all } i \in \mathbb{Z}_\infty^+\}$
$X \times Y$	*Cartesian product* $X \times Y = \{(x, y) : x \in X, y \in Y\}$
$\prod\limits_{i=1}^{n} X_i$	$\prod\limits_{i=1}^{n} X_i = \{(x_1, x_2, \ldots, x_n) : x_i \in X_i\}$
$\prod\limits_{i=1}^{\infty} X_i$	$\prod\limits_{i=1}^{\infty} X_i = \{(x_1, x_2, x_3, \ldots) : x_i \in X_i\}$
$\mathbb{F}^n$	The Cartesian product of n copies of $\mathbb{F}$, i.e., $\mathbb{F}^n = \prod\limits_{i=1}^{n} \mathbb{F}$.

Symbol	Explanation
$X \setminus Y$	*Set difference* Let X and Y be subsets of some universal set U, $X \setminus Y = \{x \in X : x \notin Y\}$.
X'	*Complement* $X' = U \setminus X$, where U is the universal set that contains X.
$card(\mathbf{X})$	The *cardinality* of the set $\mathbf{X}$.
$choice(\mathbf{X})$	A function that randomly selects an element from the set $\mathbf{X}$.

Point and Point Set Operations

Symbol	Explanation
$\mathbf{x}+\mathbf{y}$	If $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$, then $\mathbf{x}+\mathbf{y} = (x_1+y_1, \ldots, x_n+y_n)$
$\mathbf{x}-\mathbf{y}$	If $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$, then $\mathbf{x}-\mathbf{y} = (x_1-y_1, \ldots, x_n-y_n)$
$\mathbf{x}\cdot\mathbf{y}$	If $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$, then $\mathbf{x}\cdot\mathbf{y} = (x_1y_1, \ldots, x_ny_n)$
$\mathbf{x}/\mathbf{y}$	If $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$, then $\mathbf{x}/\mathbf{y} = (x_1/y_1, \ldots, x_n/y_n)$
$\mathbf{x}\vee\mathbf{y}$	If $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$, then $\mathbf{x}\vee\mathbf{y} = (x_1\vee y_1, \ldots, x_n\vee y_n)$
$\mathbf{x}\wedge\mathbf{y}$	If $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$, then $\mathbf{x}\wedge\mathbf{y} = (x_1\wedge y_1, \ldots, x_n\wedge y_n)$
$\mathbf{x}\gamma\mathbf{y}$	In general, if $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$, and $\gamma : \mathbb{R}\times\mathbb{R} \to \mathbb{R}\mathbf{x}\gamma\mathbf{y} = (x_1\gamma y_1, \ldots, x_n\gamma y_n)$
$k\gamma\mathbf{x}$	If $k \in \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^n$ and $\gamma : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$, then $k\gamma\mathbf{x} = (k\gamma x_1, \ldots, k\gamma x_n)$
$\mathbf{x}\bullet\mathbf{y}$	If $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$, then $\mathbf{x}\bullet\mathbf{y} = x_1y_1 + x_2y_2 + \cdots + x_ny_n$
$\mathbf{x}\times\mathbf{y}$	If $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$, then $\mathbf{x}\times\mathbf{y} = (x_2y_3 - x_3y_2, x_3y_1 - x_1y_3, x_1y_2 - x_2y_1)$
$\mathbf{x}\hat{\mathbf{y}}$	If $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$then $\mathbf{x}\hat{\mathbf{y}} = (x_1, \ldots, x_n, y_1, \ldots, y_m)$
$-\mathbf{x}$	If $\mathbf{x} \in \mathbb{R}^n$, then $-\mathbf{x} = (-x_1, \ldots, -x_n)$
$\lceil\mathbf{x}\rceil$	If $\mathbf{x} \in \mathbb{R}^n$, then If $\mathbf{x} \in \mathbb{R}^n$, then $\lceil\mathbf{x}\rceil = (\lceil x_1\rceil, \ldots, \lceil x_n\rceil)$
$\lfloor\mathbf{x}\rfloor$	If $\mathbf{x} \in \mathbb{R}^n$, then $\lfloor\mathbf{x}\rfloor = (\lfloor x_1\rfloor, \ldots, \lfloor x_n\rfloor)$
$[\mathbf{x}]$	If $\mathbf{x} \in \mathbb{R}^n$, then $[\mathbf{x}] = ([x_1], \ldots, [x_n])$
$p_i(\mathbf{x})$	If $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, then $p_i(\mathbf{x}) = x_i$
$\Sigma\mathbf{x}$	If $\mathbf{x} \in \mathbb{R}^n$, then $\Sigma\mathbf{x} = x_1 + x_2 + \cdots + x_n$

Symbol	Explanation
$\Pi\mathbf{x}$	If $\mathbf{x} \in \mathbb{R}^n$, then $\Pi\mathbf{x} = x_1 x_2 \cdots x_n$
$\vee\mathbf{x}$	If $\mathbf{x} \in \mathbb{R}^n$, then $\vee\mathbf{x} = x_1 \vee x_2 \vee \cdots \vee x_n$
$\wedge\mathbf{x}$	If $\mathbf{x} \in \mathbb{R}^n$, then $\wedge\mathbf{x} = x_1 \wedge x_2 \wedge \cdots \wedge x_n$
$\|\mathbf{x}\|_2$	If $\mathbf{x} \in \mathbb{R}^n$, then $\|\mathbf{x}\|_2 = \sqrt{x_1^2 + \cdots + x_n^2}$
$\|\mathbf{x}\|_1$	If $\mathbf{x} \in \mathbb{R}^n$, then $\|\mathbf{x}\|_1 = \|x_1\| + \|x_2\| + \cdots + \|x_n\|$
$\|\mathbf{x}\|_\infty$	If $\mathbf{x} \in \mathbb{R}^n$, then $\|\mathbf{x}\|_\infty = \|x_1\| \vee \|x_2\| \vee \cdots \vee \|x_n\|$
$dim(\mathbf{x})$	If $\mathbf{x} \in \mathbb{R}^n$, then $dim(\mathbf{x}) = n$
$\mathbf{X}+\mathbf{Y}$	If $\mathbf{X}, \mathbf{Y} \subseteq \mathbb{R}^n$, then $\mathbf{X}+\mathbf{Y} = \{\mathbf{x}+\mathbf{y} : \mathbf{x} \in \mathbf{X} \textit{ and } \mathbf{y} \in \mathbf{Y}\}$
$\mathbf{X}-\mathbf{Y}$	If $\mathbf{X}, \mathbf{Y} \subseteq \mathbb{R}^n$, then $\mathbf{X}-\mathbf{Y} = \{\mathbf{x}-\mathbf{y} : \mathbf{x} \in \mathbf{X} \textit{ and } \mathbf{y} \in \mathbf{Y}\}$
$\mathbf{X}+\mathbf{p}$	If $\mathbf{X} \subseteq \mathbb{R}^n$, then $\mathbf{X}+\mathbf{p} = \{\mathbf{x}+\mathbf{p} : \mathbf{x} \in \mathbf{X}\}$
$\mathbf{X}-\mathbf{p}$	If $\mathbf{X} \subseteq \mathbb{R}^n$, then $\mathbf{X}-\mathbf{p} = \{\mathbf{x}-\mathbf{p} : \mathbf{x} \in \mathbf{X}\}$
$\mathbf{X}\cup\mathbf{Y}$	If $\mathbf{X}, \mathbf{Y} \subseteq \mathbb{R}^n$, then $\mathbf{X}\cup\mathbf{Y} = \{\mathbf{z} : \mathbf{z} \in \mathbf{X} \textit{ or } \mathbf{z} \in \mathbf{Y}\}$
$\mathbf{X}\backslash\mathbf{Y}$	If $\mathbf{X}, \mathbf{Y} \subseteq \mathbb{R}^n$, then $\mathbf{X}\backslash\mathbf{Y} = \{\mathbf{z} : \mathbf{z} \in \mathbf{X} \textit{ and } \mathbf{z} \notin \mathbf{Y}\}$
$\mathbf{X}\triangle\mathbf{Y}$	If $\mathbf{X}, \mathbf{Y} \subseteq \mathbb{R}^n$, then $\mathbf{X}\triangle\mathbf{Y} = \{\mathbf{z} : \mathbf{z} \in \mathbf{X}\cup\mathbf{Y} \textit{ and } \mathbf{z} \notin \mathbf{X}\cap\mathbf{Y}\}$
$\mathbf{X}\times\mathbf{Y}$	If $\mathbf{X}, \mathbf{Y} \subseteq \mathbb{R}^n$, then $\mathbf{X}\times\mathbf{Y} = \{(\mathbf{x},\mathbf{y}) : \mathbf{x} \in \mathbf{X} \textit{ and } \mathbf{y} \in \mathbf{Y}\}$
$-\mathbf{X}$	If $\mathbf{X} \subseteq \mathbb{R}^n$, then $-\mathbf{X} = \{-\mathbf{x} : \mathbf{x} \in \mathbf{X}\}$
$\tilde{\mathbf{X}}$	If $\mathbf{X} \subseteq \mathbb{R}^n$, then $\tilde{\mathbf{X}} = \{\mathbf{z} : \mathbf{z} \in \mathbb{R}^{\mathbf{n}} \textit{ and } \mathbf{z} \notin \mathbf{X}\}$
$sup(\mathbf{X})$	If $\mathbf{X} \subseteq \mathbb{R}^n$, then $sup(\mathbf{X}) =$ the supremum of $\mathbf{X}$. If $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, then $sup(\mathbf{X}) = \mathbf{x}_1 \vee \mathbf{x}_2 \vee \ldots \vee \mathbf{x}_n$
$\bigvee\mathbf{X}$	For a point set $\mathbf{X}$ with total order $\prec$, $\mathbf{x}_0 = \bigvee\mathbf{X} \Leftrightarrow \mathbf{x} \prec \mathbf{x}_0, \forall\mathbf{x} \in \mathbf{X}\setminus\{\mathbf{x}_0\}$
$inf(\mathbf{X})$	If $\mathbf{X} \subseteq \mathbb{R}^n$, then $inf(\mathbf{X}) =$ the infimum of $\mathbf{X}$. If $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, , then $sup(\mathbf{X}) = \mathbf{x}_1 \wedge \mathbf{x}_2 \wedge \ldots \wedge \mathbf{x}_n$
$\bigwedge\mathbf{X}$	For a point set $\mathbf{X}$ with total order $\prec$, $\mathbf{x}_0 = \bigwedge\mathbf{X} \Leftrightarrow \mathbf{x_0} \prec x, \forall\mathbf{x} \in \mathbf{X}\setminus\{\mathbf{x}_0\}$
$choice(\mathbf{X})$	If $\mathbf{X} \subseteq \mathbb{R}^n$, then $choice(\mathbf{X}) \in \mathbf{X}$ $(randomly\ chosen\ element)$
$card(\mathbf{X})$	If $\mathbf{X} \subseteq \mathbb{R}^n$, then $card(\mathbf{X}) = the\ cardinality\ of\ \mathbf{X}$

Morphology

In following table $\mathbf{A}, \mathbf{B}, \mathbf{D}$, and $\mathbf{E}$ denote subsets of $\mathbb{R}^n$.

Symbol	Explanation
$\mathbf{A}^*$	The reflection of $\mathbf{A}$ across the origin $\mathbf{0} = (0, 0, \ldots, 0) \in \mathbb{R}^n$.
$\mathbf{A}'$	The complement of $\mathbf{A}$; i.e., $\mathbf{A}' = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \notin \mathbf{A}\}$.
$\mathbf{A_b}$	$\mathbf{A_b} = \{\mathbf{a} + \mathbf{b} : \mathbf{a} \in \mathbf{A}\}$
$\mathbf{A} \times \mathbf{B}$	*Minkowski addition* is defined as $\mathbf{A} \times \mathbf{B} = \{\mathbf{a} + \mathbf{b} : \mathbf{a} \in \mathbf{A}, \mathbf{b} \in \mathbf{B}\}$. (Section 7.2)
$\mathbf{A}/\mathbf{B}$	*Minkowski subtraction* is defined as $\mathbf{A}/\mathbf{B} = (\mathbf{A}' \times \mathbf{B}^*)'$. (Section 7.2)
$\mathbf{A} \circ \mathbf{B}$	The *opening of* $\mathbf{A}$ *by* $\mathbf{B}$ is denoted $\mathbf{A} \circ \mathbf{B}$ and is defined by $\mathbf{A} \circ \mathbf{B} = (\mathbf{A}/\mathbf{B}) \times \mathbf{B}$. (Section 7.3)
$\mathbf{A} \bullet \mathbf{B}$	The *closing of* $\mathbf{A}$ *by* $\mathbf{B}$ is denoted $\mathbf{A} \bullet \mathbf{B}$ and is defined by $\mathbf{A} \bullet \mathbf{B} = (\mathbf{A} \times \mathbf{B})/\mathbf{B}$. (Section 7.3)
$\mathbf{A} \circledast \mathbf{C}$	Let $\mathbf{C} = (\mathbf{D}, \mathbf{E})$ be an ordered pair of structuring elements. The *hit-and-miss transform* of the set $\mathbf{A}$ is given by $\mathbf{A} \circledast \mathbf{C} = \{\mathbf{p} : \mathbf{D_p} \subset \mathbf{A} \text{ and } \mathbf{E_p} \subset \mathbf{A}'\}$. (Section 7.5)

Functions and Scalar Operations

Symbol	Explanation
$f : X \to Y$	f is a *function* from X into Y.
$domain(f)$	The *domain* of the function $f : X \to Y$ is the set X.
$range(f)$	The *range* of the function $f : X \to Y$ is the set $\{f(x) : x \in X\}$.
f^{-1}	The *inverse* of the function f.
Y^X	The set of all functions from X into Y, i.e., if $f \in \mathbf{Y}^{\mathbf{X}}$, then $f : \mathbf{X} \to \mathbf{Y}$.
$f\|_A$	Given a function $f : X \to Y$ and a subset $A \subset X$, the *restriction of* f *to* A, $f\|_A : A \to Y$, is defined by $f\|_A(a) = f(a)$ for $a \in A$.
$f\|^g$	Given $f : A \to Y$ and $g : B \to Y$, the *extension of* f *to* g is defined by $f\|^g(x) = \begin{cases} f(x) & \text{if } x \in A \\ g(x) & \text{if } x \in B \setminus A. \end{cases}$

Symbol	Explanation
$g \circ f$	Given two functions $f : X \to Y$ and $g : Y \to Z$, the *composition* $g \circ f : X \to Z$ is defined by $(g \circ f)(x) = g(f(x))$, for every $x \in X$.
$f + g$	Let f and g be real or complex-valued functions, then $(f + g)(x) = f(x) + g(x)$.
$f \cdot g$	Let f and g be real or complex-valued functions, then $(f \cdot g)(x) = f(x) \cdot g(x)$.
$\mathrm{k} \cdot f$	Let f be a real or complex-valued function, and k be a real or complex number, then $f \in \mathbb{F}^{\mathbf{X}}$, $(\mathrm{k} \cdot f)(x) = \mathrm{k} \cdot (f(x))$.
$\lvert f \rvert$	$\lvert f \rvert(x) = \lvert f(x) \rvert$, where f is a real (or complex)-valued function, and $\lvert f(x) \rvert$ denotes the absolute value (or magnitude) of $f(x)$.
1_X	The *identity function* $1_X : X \to X$ is given by $1_X(x) = x$.
$p_j : \prod_{i=1}^{n} X_i \to X_j$	The *projection function* p_j *onto the* j*th coordinate* is defined by $p_j(x_1, \ldots, x_j, \ldots, x_n) = x_j$.
$card(X)$	The *cardinality* of the set X.
$choice(X)$	A function which randomly selects an element from the set X.
$x \vee y$	For $x, y \in \mathbb{R}$, $x \vee y$ is the maximum of x and y.
$x \wedge y$	For $x, y \in \mathbb{R}$, $x \wedge y$ is the minimun of x and y.
$\lceil x \rceil$	For $x \in \mathbb{R}$ the *ceiling function* $\lceil x \rceil$ returns the smallest integer that is greater than or equal to x.
$\lfloor x \rfloor$	For $x \in \mathbb{R}$ the *floor function* $\lfloor x \rfloor$ returns the largest integer that is less than or equal to x.
$[x]$	For $x \in \mathbb{R}$ the *round function* returns the nearest integer to x. If there are two such integers it yields the integer with greater magnitude.
$x \ mod \ y$	For $x, y \in \mathbb{N}$, $x \ mod \ y = r$ if there exists $k, r \in \mathbb{N}$ with $r < y$ such that $x = yk + r$.
$\chi_S(x)$	The *characteristic function* χ_S is defined by $\chi_S(x) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{otherwise.} \end{cases}$

Images and Image Operations

Symbol	Explanation
$\mathbf{a}, \mathbf{b}, \mathbf{c}$	Bold, lowercase characters are used to represent *images*. Image variables will usually be chosen from the beginning of the alphabet.
$\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$	The image $\mathbf{a}$ is an $\mathbb{F}$-*valued image* on $\mathbf{X}$. The set $\mathbb{F}$ is called the *value set* of $\mathbf{a}$ and $\mathbf{X}$ the *spatial domain* of $\mathbf{a}$.
$\mathbf{1} \in \mathbb{F}^{\mathbf{X}}$	Let $\mathbb{F}$ be a set with unit 1. Then $\mathbf{1}$ denotes an image, all of whose pixel values are 1.
$\mathbf{0} \in \mathbb{F}^{\mathbf{X}}$	Let $\mathbb{F}$ be a set with zero 0. Then $\mathbf{0}$ denotes an image, all of whose pixel values are 0.
$\mathbf{a}\vert_{\mathbf{Z}}$	The *domain restriction* of $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$ to a subset $\mathbf{Z}$ of $\mathbf{X}$ is defined by $\mathbf{a}\vert_{\mathbf{Z}} = \mathbf{a} \cap (\mathbf{Z} \times \mathbb{F})$.
$\mathbf{a}\vert\vert_S$	The *range restriction* of $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$ to the subset $S \subset \mathbb{F}$ is defined by $\mathbf{a}\vert\vert_S = \mathbf{a} \cap (\mathbf{X} \times S)$. The double-bar notation is used to focus attention on the fact that the restriction is applied to the second coordinate of $\mathbf{a} \subset \mathbf{X} \times \mathbb{F}$.
$\mathbf{a}\vert_{(\mathbf{Z},S)}$	If $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$, $\mathbf{Z} \subset \mathbf{X}$, and $S \subset \mathbb{F}$, then the *restriction of* $\mathbf{a}$ *to* $\mathbf{Z}$ *and* S is defined as $\mathbf{a}\vert_{(\mathbf{Z},S)} = \mathbf{a} \cap (\mathbf{Z} \times S)$.
$\mathbf{a}\vert^{\mathbf{b}}$	Let $\mathbf{X}$ and $\mathbf{Y}$ be subsets of the same topological space. The *extension of* $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$ to $\mathbf{b} \in \mathbb{F}^{\mathbf{Y}}$ is defined by $\mathbf{a}\vert^{\mathbf{b}}(\mathbf{x}) = \begin{cases} \mathbf{a}(\mathbf{x}) & \text{if } x \in \mathbf{X} \\ \mathbf{b}(\mathbf{y}) & \text{if } x \in \mathbf{Y} \setminus \mathbf{X}. \end{cases}$
$(\mathbf{a}\vert\mathbf{b}),\ (\mathbf{a}_1\vert\mathbf{a}_2\vert\cdots\vert\mathbf{a}_n)$	Row concatenation of images $\mathbf{a}$ and $\mathbf{b}$, respectively the row concatenation of images $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$.
$\begin{pmatrix} \mathbf{a} \\ - \\ \mathbf{b} \end{pmatrix}$	Column concatenation of images $\mathbf{a}$ and $\mathbf{b}$.
$f(\mathbf{a})$	If $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$ and $f : \mathbb{F} \to Y$, then the image $f(\mathbf{a}) \in Y^{\mathbf{X}}$ is given by $f \circ \mathbf{a}$, i.e., $f(\mathbf{a}) = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = f(\mathbf{a}(\mathbf{x})), \mathbf{x} \in \mathbf{X}\}$.
$\mathbf{a} \circ f$	If $f : \mathbf{Y} \to \mathbf{X}$ and $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$, the induced image $\mathbf{a} \circ f \in \mathbb{F}^{\mathbf{Y}}$ is defined by $\mathbf{a} \circ f = \{(\mathbf{y}, \mathbf{a}(f(\mathbf{y}))) : \mathbf{y} \in \mathbf{Y}\}$.
$\mathbf{a}\,\gamma\,\mathbf{b}$	If γ is a binary operation on $\mathbb{F}$, then an induced operation on $\mathbb{F}^{\mathbf{X}}$ can be defined. Let $\mathbf{a}, \mathbf{b} \in \mathbb{F}^{X}$; the induced operation is given by $\mathbf{a}\,\gamma\,\mathbf{b} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(x) = \mathbf{a}(\mathbf{x})\,\gamma\,\mathbf{b}(\mathbf{x}), \mathbf{x} \in \mathbf{X}\}$.

Symbol	Explanation
$k \,\gamma\, \mathbf{a}$	Let $k \in \mathbb{F}$, $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$, and γ be a binary operation on $\mathbb{F}$. An induced scalar operation on images is defined by $k \,\gamma\, \mathbf{a} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = k \,\gamma\, \mathbf{a}(\mathbf{x}), \mathbf{x} \in \mathbf{X}\}$.
$\mathbf{a}^{\mathbf{b}}$	Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^{\mathbf{X}}$; $\mathbf{a}^{\mathbf{b}} = \left\{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x})^{\mathbf{b}(\mathbf{x})}, \mathbf{x} \in \mathbf{X}\right\}$.
$log_{\mathbf{b}}\mathbf{a}$	Let $\mathbf{a}, \mathbf{b} \in (\mathbb{R}^{+})^{\mathbf{X}}$ $log_{\mathbf{b}}\mathbf{a} = \left\{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = log_{\mathbf{b}(\mathbf{x})}\mathbf{a}(x), \mathbf{x} \in \mathbf{X}\right\}$.
$\mathbf{a}^{*}$	Pointwise complex conjugate of image $\mathbf{a}$, $\mathbf{a}^{*}(\mathbf{x}) = (\mathbf{a}(\mathbf{x}))^{*}$.
$\Gamma\mathbf{a}$	$\Gamma\mathbf{a}$ denotes reduction by a generic reduce operation $\Gamma : \mathbb{F}^{\mathbf{X}} \to \mathbb{F}$.

The following four items are specific examples of the global reduce operation. Each assumes $\mathbf{a} \in \mathbb{R}^{\mathbf{X}}$ and $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$.

$\sum \mathbf{a}$	$\sum \mathbf{a} = \mathbf{a}(\mathbf{x}_1) + \mathbf{a}(\mathbf{x}_2) + \cdots + \mathbf{a}(\mathbf{x}_n)$
$\prod \mathbf{a}$	$\prod \mathbf{a} = \mathbf{a}(\mathbf{x}_1) \cdot \mathbf{a}(\mathbf{x}_2) \cdot \cdots \cdot \mathbf{a}(\mathbf{x}_n)$
$\bigvee \mathbf{a}$	$\bigvee \mathbf{a} = \mathbf{a}(\mathbf{x}_1) \vee \mathbf{a}(\mathbf{x}_2) \vee \cdots \vee \mathbf{a}(\mathbf{x}_n)$
$\bigwedge \mathbf{a}$	$\bigwedge \mathbf{a} = \mathbf{a}(\mathbf{x}_1) \wedge \mathbf{a}(\mathbf{x}_2) \wedge \cdots \wedge \mathbf{a}(\mathbf{x}_n)$

$\mathbf{a} \bullet \mathbf{b}$	Dot product, $\mathbf{a} \bullet \mathbf{b} = \Sigma(\mathbf{a} \cdot \mathbf{b}) = \sum_{\mathbf{x} \in \mathbf{X}} (\mathbf{a}(\mathbf{x}) \cdot \mathbf{b}(\mathbf{x}))$.
$\tilde{\mathbf{a}}$	Complementation of a set-valued image $\mathbf{a}$.
$\mathbf{a}^{c}$	Complementation of a Boolean image $\mathbf{a}$.
$\mathbf{a}'$	Transpose of image $\mathbf{a}$.

Templates and Template Operations

Symbol	Explanation
$\mathbf{s}, \mathbf{t}, \mathbf{u}$	Bold, lowercase characters are used to represent *templates*. Usually characters from the middle of the alphabet are used as template variables.
$\mathbf{t} \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$	A template is an image whose pixel values are images. In particular, an $\mathbb{F}$*-valued template from* $\mathbf{Y}$ *to* $\mathbf{X}$ is a function $\mathbf{t} : \mathbf{Y} \to \mathbb{F}^{\mathbf{X}}$. Thus, $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$ and $\mathbf{t}$ is an $\mathbb{F}^{\mathbf{X}}$-valued image on $\mathbf{Y}$.
$\mathbf{t}_{\mathbf{y}}$	Let $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$. For each $\mathbf{y} \in \mathbf{Y}$, $\mathbf{t}_{\mathbf{y}} = \mathbf{t}(\mathbf{y})$. The image $\mathbf{t}_{\mathbf{y}} \in \mathbb{F}^{\mathbf{X}}$ is given by $\mathbf{t}_{\mathbf{y}} = \{(\mathbf{x}, \mathbf{t}_{\mathbf{y}}(\mathbf{x})) : \mathbf{x} \in \mathbf{X}\}$.
$S(\mathbf{t}_{\mathbf{y}})$	If $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}$ and $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$, then the *support* of $\mathbf{t}$ is denoted by $S(\mathbf{t}_{\mathbf{y}})$ and is defined by $S(\mathbf{t}_{\mathbf{y}}) = \{\mathbf{x} \in \mathbf{X} : \mathbf{t}_{\mathbf{y}}(\mathbf{x}) \neq 0\}$.
$S_{\infty}(\mathbf{t}_{\mathbf{y}})$	If $\mathbf{t} \in \left(\mathbb{R}_{\infty}^{\mathbf{X}}\right)^{\mathbf{Y}}$, then $S_{\infty}(\mathbf{t}_{\mathbf{y}}) = \{\mathbf{x} \in \mathbf{X} : \mathbf{t}_{\mathbf{y}}(\mathbf{x}) \neq \infty\}$.
$S_{-\infty}(\mathbf{t}_{\mathbf{y}})$	If $\mathbf{t} \in \left(\mathbb{R}_{-\infty}^{\mathbf{X}}\right)^{\mathbf{Y}}$, then $S_{-\infty}(\mathbf{t}_{\mathbf{y}}) = \{\mathbf{x} \in \mathbf{X} : \mathbf{t}_{\mathbf{y}}(\mathbf{x}) \neq -\infty\}$.
$S_{\pm\infty}(\mathbf{t}_{\mathbf{y}})$	If $\mathbf{t} \in \left(\mathbb{R}_{\pm\infty}^{\mathbf{X}}\right)^{\mathbf{Y}}$, then $S_{\pm\infty}(\mathbf{t}_{\mathbf{y}}) = \{\mathbf{x} \in \mathbf{X} : \mathbf{t}_{\mathbf{y}}(\mathbf{x}) \neq \pm\infty\}$.
$\mathbf{t}(p)$	A parameterized $\mathbb{F}$*-valued template from* $\mathbf{Y}$ *to* $\mathbf{X}$ *with parameters in* P is a function of the form $\mathbf{t} : P \to \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$.
$\mathbf{t}'$	Let $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$. The *transpose* $\mathbf{t}' \in \left(\mathbb{F}^{\mathbf{Y}}\right)^{\mathbf{X}}$ is defined as $\mathbf{t}'_{\mathbf{x}}(\mathbf{y}) = \mathbf{t}_{\mathbf{y}}(\mathbf{x})$.

Image-Template Operations

In the table below, $\mathbf{X}$ is a finite subset of $\mathbb{R}^n$.

Symbol	Explanation
$\mathbf{a} \circledgamma \mathbf{t}$	Let $(\mathbb{F}, \gamma, \bigcirc)$ be a semiring and $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$, $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$, then the *generic right product of* $\mathbf{a}$ *with* $\mathbf{t}$ is defined as $\mathbf{a} \circledgamma \mathbf{t} = \left\{(\mathbf{y}, \mathbf{b}(\mathbf{y})) : \mathbf{y} \in \mathbf{Y},\ \mathbf{b}(\mathbf{y}) = \underset{x \in \mathbf{X}}{\Gamma} \mathbf{a}(\mathbf{x}) \bigcirc \mathbf{t}_{\mathbf{y}}(\mathbf{x})\right\}$.
$\mathbf{t} \circledgamma \mathbf{a}$	With the conditions above, except that now $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{Y}}\right)^{\mathbf{X}}$, the *generic left product of* $\mathbf{a}$ *with* $\mathbf{t}$ is defined as $\mathbf{t} \circledgamma \mathbf{a} = \left\{(\mathbf{y}, \mathbf{b}(\mathbf{y})) : \mathbf{y} \in \mathbf{Y}, \mathbf{b}(\mathbf{y}) = \underset{x \in \mathbf{X}}{\Gamma} \mathbf{a}(\mathbf{x}) \bigcirc \mathbf{t}_{\mathbf{x}}(\mathbf{y})\right\}$.

Symbol	Explanation
$\mathbf{a} \oplus \mathbf{t}$	Let $\mathbf{Y} \subset \mathbb{R}^m$, $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$, and $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$, where $\mathbb{F} \in \{\mathbb{C}, \mathbb{R}\}$. The *right linear product (or convolution)* is defined as $\mathbf{a} \oplus \mathbf{t} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) : \mathbf{y} \in \mathbf{Y}, \mathbf{b}(\mathbf{y}) = \sum_{x \in \mathbf{X} \cap \mathbf{S}(\mathbf{t}_{\mathbf{y}})} \mathbf{a}(\mathbf{x}) \cdot \mathbf{t}_{\mathbf{y}}(\mathbf{x}) \right\}.$
$\mathbf{t} \oplus \mathbf{a}$	With the conditions above, except that $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{Y}}\right)^{\mathbf{X}}$, the *left linear product (or convolution)* is defined as $\mathbf{t} \oplus \mathbf{a} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) : \mathbf{y} \in \mathbf{Y}, \mathbf{b}(\mathbf{y}) = \sum_{x \in \mathbf{X} \cap \mathbf{S}\left(\mathbf{t}'_{\mathbf{y}}\right)} \mathbf{a}(\mathbf{x}) \cdot \mathbf{t}_{\mathbf{x}}(\mathbf{y}) \right\}.$
$\mathbf{a} \boxed{\vee} \mathbf{t}$	For $\mathbf{a} \in \mathbb{R}^{\mathbf{X}}_{\pm\infty}$ and $\mathbf{t} \in \left(\mathbb{R}^{\mathbf{X}}_{\pm\infty}\right)^{\mathbf{Y}}$, the *right additive maximum* is defined by $\mathbf{a} \boxed{\vee} \mathbf{t} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) : \mathbf{y} \in \mathbf{Y}, \mathbf{b}(\mathbf{y}) = \bigvee_{x \in \mathbf{X} \cap \mathbf{S}_{-\infty}(\mathbf{t}_{\mathbf{y}})} \mathbf{a}(\mathbf{x}) + \mathbf{t}_{\mathbf{y}}(\mathbf{x}) \right\}.$
$\mathbf{t} \boxed{\vee} \mathbf{a}$	For $\mathbf{a} \in \mathbb{R}^{\mathbf{X}}_{\pm\infty}$ and $\mathbf{t} \in \left(\mathbb{R}^{\mathbf{Y}}_{\pm\infty}\right)^{\mathbf{X}}$, the *left additive maximum* is defined by $\mathbf{t} \boxed{\vee} \mathbf{a} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) : \mathbf{y} \in \mathbf{Y}, \mathbf{b}(\mathbf{y}) = \bigvee_{x \in \mathbf{X} \cap \mathbf{S}_{-\infty}\left(\mathbf{t}'_{\mathbf{y}}\right)} \mathbf{a}(\mathbf{x}) + \mathbf{t}_{\mathbf{x}}(\mathbf{y}) \right\}.$
$\mathbf{a} \boxed{\wedge} \mathbf{t}$	For $\mathbf{a} \in \mathbb{R}^{\mathbf{X}}_{\pm\infty}$ and $\mathbf{t} \in \left(\mathbb{R}^{\mathbf{X}}_{\pm\infty}\right)^{\mathbf{Y}}$, the *right additive minimum* is defined by $\mathbf{a} \boxed{\wedge} \mathbf{t} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) : \mathbf{y} \in \mathbf{Y},\ \mathbf{b}(\mathbf{y}) = \bigwedge_{x \in \mathbf{X} \cap S_{\infty}(\mathbf{t}_y)} \mathbf{a}(\mathbf{x}) +' \mathbf{t}_{\mathbf{y}}(\mathbf{x}) \right\}.$
$\mathbf{t} \boxed{\wedge} \mathbf{a}$	For $\mathbf{a} \in \mathbb{R}^{\mathbf{X}}_{\pm\infty}$ and $\mathbf{t} \in \left(\mathbb{R}^{\mathbf{Y}}_{\pm\infty}\right)^{\mathbf{X}}$, the *left additive minimum* is defined by $\mathbf{t} \boxed{\wedge} \mathbf{a} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) : \mathbf{y} \in \mathbf{Y}, \mathbf{b}(\mathbf{y}) = \bigwedge_{x \in \mathbf{X} \cap \mathbf{S}_{\infty}\left(\mathbf{t}'_{\mathbf{y}}\right)} \mathbf{a}(\mathbf{x}) +' \mathbf{t}_{\mathbf{x}}(\mathbf{y}) \right\}.$
$\mathbf{a} \circledvee \mathbf{t}$	For $\mathbf{a} \in \left(\mathbb{R}^{\geq 0}_{\infty}\right)^{\mathbf{X}}$ and $\mathbf{t} \in \left(\left(\mathbb{R}^{\geq 0}_{\infty}\right)^{\mathbf{X}}\right)^{\mathbf{Y}}$, the *right multiplicative maximum* is defined by $\mathbf{a} \circledvee \mathbf{t} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) : \mathbf{y} \in \mathbf{Y}, \mathbf{b}(\mathbf{y}) = \bigvee_{x \in \mathbf{X} \cap \mathbf{S}(\mathbf{t}_{\mathbf{y}})} \mathbf{a}(\mathbf{x}) \times \mathbf{t}_{\mathbf{y}}(\mathbf{x}) \right\}.$
$\mathbf{t} \circledvee \mathbf{a}$	For $\mathbf{a} \in \left(\mathbb{R}^{\geq 0}_{\infty}\right)^{\mathbf{X}}$ and $\mathbf{t} \in \left(\left(\mathbb{R}^{\geq 0}_{\infty}\right)^{\mathbf{Y}}\right)^{\mathbf{X}}$, the *left multiplicative maximum* is defined by $\mathbf{t} \circledvee \mathbf{a} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) : \mathbf{y} \in \mathbf{Y}, \mathbf{b}(\mathbf{y}) = \bigvee_{x \in \mathbf{X} \cap \mathbf{S}\left(\mathbf{t}'_{\mathbf{y}}\right)} \mathbf{a}(\mathbf{x}) \times \mathbf{t}_{\mathbf{x}}(\mathbf{y}) \right\}.$

Symbol	Explanation
$\mathbf{a} \textcircled{\wedge} \mathbf{t}$	For $\mathbf{a} \in \left(\mathbb{R}_{\infty}^{\geq 0}\right)^{\mathbf{X}}$ and $\mathbf{t} \in \left(\left(\mathbb{R}_{\infty}^{\geq 0}\right)^{\mathbf{X}}\right)^{\mathbf{Y}}$, the *right multiplicative minimum* is defined by $\mathbf{a} \textcircled{\wedge} \mathbf{t} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) : \mathbf{y} \in \mathbf{Y}, \mathbf{b}(\mathbf{y}) = \bigwedge_{x \in \mathbf{X} \cap \mathbf{S}_{\infty}(\mathbf{t_y})} \mathbf{a}(\mathbf{x}) \times' \mathbf{t_y}(\mathbf{x}) \right\}$.
$\mathbf{t} \textcircled{\wedge} \mathbf{a}$	For $\mathbf{a} \in \left(\mathbb{R}_{\infty}^{\geq 0}\right)^{\mathbf{X}}$ and $\mathbf{t} \in \left(\left(\mathbb{R}_{\infty}^{\geq 0}\right)^{\mathbf{Y}}\right)^{\mathbf{X}}$, the *left multiplicative minimum* is defined by $\mathbf{t} \textcircled{\wedge} \mathbf{a} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) : \mathbf{y} \in \mathbf{Y}, \mathbf{b}(\mathbf{y}) = \bigwedge_{x \in \mathbf{X} \cap \mathbf{S}_{\infty}(\mathbf{t}'_{\mathbf{y}})} \mathbf{a}(\mathbf{x}) \times' \mathbf{t_x}(\mathbf{y}) \right\}$.

Neighborhoods and Neighborhood Operations

Symbol	Explanation
M, N	Italic uppercase characters are used to denote *neighborhoods*.
$N \in \left(2^{\mathbf{X}}\right)^{\mathbf{Y}}$	A neighborhood is an image whose pixel values are sets of points. In particular, a *neighborhood from* $\mathbf{Y}$ *to* $\mathbf{X}$ is a function $N : \mathbf{Y} \rightarrow 2^{\mathbf{X}}$.
$N(p)$	A parameterized neighborhood *from* $\mathbf{Y}$ *to* $\mathbf{X}$ *with parameters in* P is a function of the form $N : P \rightarrow \left(2^{\mathbf{X}}\right)^{\mathbf{Y}}$.
N'	Let $N \in \left(2^{\mathbf{X}}\right)^{\mathbf{Y}}$, the *transpose* $N' \in \left(2^{\mathbf{Y}}\right)^{\mathbf{X}}$ is defined as $N'(\mathbf{x}) = \{\mathbf{y} \in \mathbf{Y} : \mathbf{x} \in N(\mathbf{y})\}$, that is, $\mathbf{x} \in N(\mathbf{y})$ iff $\mathbf{y} \in \mathrm{N}'(\mathbf{x})$.
$N_1 \oplus N_2$	The *dilation* of N_1 by N_2 is defined by $N(\mathbf{y}) = \bigcup_{\mathbf{p} \in N_2(\mathbf{y})} (N_1(\mathbf{y}) + (\mathbf{p} - \mathbf{y}))$.

Image-Neighborhood Operations

In the table below, $\mathbf{X}$ is a finite subset of $\mathbb{R}^n$.

Symbol	Explanation
$\mathbf{a} \textcircled{\Gamma} N$	Given $\mathbf{a} \in \mathbb{F}^{\mathbf{Y}}$ and $N \in \left(2^{\mathbf{X}}\right)^{\mathbf{Y}}$, and reduce operation $\Gamma : \mathbb{F}^{\mathbf{X}} \rightarrow \mathbb{F}$, the *generic right reduction of* $\mathbf{a}$ *with* N is defined as $(\mathbf{a} \textcircled{\Gamma} N)(\mathbf{x}) = \Gamma \mathbf{a}\|_{N(\mathbf{x})}$.
$N \textcircled{\Gamma} \mathbf{a}$	With the conditions above, except that now $N \in \left(2^{\mathbf{X}}\right)^{\mathbf{Y}}$, the *generic left reduction of* $\mathbf{a}$ *with* $\mathbf{t}$ is defined as $(N \textcircled{\Gamma} \mathbf{a})(\mathbf{x}) = (\mathbf{a} \textcircled{\Gamma} N')(\mathbf{x})$.

Symbol	Explanation
$\mathbf{a} \circledA N$	Given $\mathbf{a} \in \mathbb{R}^{\mathbf{Y}}$, and the *image average* function $a : \mathbb{R}^{\mathbf{Y}} \to \mathbb{R}$, yielding the average of its image argument, $(\mathbf{a} \circledA N)(\mathbf{x}) = a\big(\mathbf{a}\vert_{N(\mathbf{x})}\big)$.
$\mathbf{a} \circledm N$	Given $\mathbf{a} \in \mathbb{R}^{\mathbf{Y}}$, and the *image median* function $m : \mathbb{R}^{\mathbf{Y}} \to \mathbb{R}$, yielding the average of its image argument, $(\mathbf{a} \circledm N)(\mathbf{x}) = m\big(\mathbf{a}\vert_{N(\mathbf{x})}\big)$.

Matrix and Vector Operations

In the table below, A and B represent matrices.

Symbol	Explanation
A^*	The *conjugate* of matrix A.
A'	The *transpose* of matrix A.
$A \times B$, AB	The matrix *product* of matrices A and B.
$A \otimes B$	The *tensor product* of matrices A and B.
$A \bigoplus_p B$	The *p-product* of matrices A and B.
$A \bigoplus_p' B$	The dual *p-product* of matrices A and B, defined by $A \bigoplus_p' B = \big(B' \bigoplus_p A'\big)'$.

References

[1] G. Ritter, "Image algebra with applications." Unpublished manuscript, available via anonymous ftp from `ftp://ftp.cis.ufl.edu/pub/src/ia/documents`, 1994.

To our brothers,
Friedrich Karl and
Scott Winfield

Contents

CHAPTER 1
IMAGE ALGEBRA

1.1. Introduction

Since the field of image algebra is a recent development it will be instructive to provide some background information. In the broad sense, image algebra is a mathematical theory concerned with the transformation and analysis of images. Although much emphasis is focused on the analysis and transformation of digital images, the main goal is the establishment of a comprehensive and unifying theory of image transformations, image analysis, and image understanding in the discrete as well as the continuous domain [1].

The idea of establishing a unifying theory for the various concepts and operations encountered in image and signal processing is not new. Over thirty years ago, Unger proposed that many algorithms for image processing and image analysis could be implemented in parallel using *cellular array* computers [2]. These cellular array computers were inspired by the work of von Neumann in the 1950s [3, 4]. Realization of von Neumann's cellular array machines was made possible with the advent of VLSI technology. NASA's massively parallel processor or MPP and the CLIP series of computers developed by Duff and his colleagues represent the classic embodiment of von Neumann's original automaton [5, 6, 7, 8, 9]. A more general class of cellular array computers are pyramids and Thinking Machines Corporation's Connection Machines [10, 11, 12]. In an abstract sense, the various versions of Connection Machines are universal cellular automatons with an additional mechanism added for non-local communication.

Many operations performed by these cellular array machines can be expressed in terms of simple elementary operations. These elementary operations create a mathematical basis for the theoretical formalism capable of expressing a large number of algorithms for image processing and analysis. In fact, a common thread among designers of parallel image processing architectures is the belief that large classes of image transformations can be described by a small set of standard rules that induce these architectures. This belief led to the creation of mathematical formalisms that were used to aid in the design of special-purpose parallel architectures. Matheron and Serra's Texture Analyzer [13] ERIM's (Environmental Research Institute of Michigan) Cytocomputer [14, 15, 16], and Martin Marietta's GAPP [17, 18, 19] are examples of this approach.

The formalism associated with these cellular architectures is that of pixel neighborhood arithmetic and mathematical morphology. Mathematical morphology is the part of image processing concerned with image filtering and analysis by structuring elements. It grew out of the early work of Minkowski and Hadwiger [20, 21, 22], and entered the modern era through the work of Matheron and Serra of the Ecole des Mines in Fontainebleau, France [23, 24, 25, 26]. Matheron and Serra not only formulated the modern concepts of morphological image transformations, but also designed and built the Texture Analyzer System. Since those early days, morphological operations have been applied from low-level, to intermediate, to high-level vision problems. Among some recent research papers on morphological image processing are Crimmins and Brown [27], Haralick et al. [28, 29], Maragos and Schafer [30, 31, 32], Davidson [33, 34], Dougherty [35], Goutsias [36, 37], and Koskinen and Astola [38].

Serra and Sternberg were the first to unify morphological concepts and methods into a coherent algebraic theory specifically designed for image processing and image analysis. Sternberg was also the first to use the term “image algebra” [39, 40]. In the mid 1980s, Maragos introduced a new theory unifying a large class of linear and nonlinear systems under the theory of mathematical morphology [41]. More recently, Davidson completed the mathematical foundation of mathematical morphology by formulating its embedding into the lattice algebra known as *Mini-Max algebra* [42, 43]. However, despite these profound accomplishments, morphological methods have some well-known limitations. For example, such fairly common image processing techniques as feature extraction based on convolution, Fourier-like transformations, chain coding, histogram equalization transforms, image rotation, and image registration and rectification are — with the exception of a few simple cases — either extremely difficult or impossible to express in terms of morphological operations. The failure of a morphologically based image algebra to express a fairly straightforward U.S. government-furnished FLIR (forward-looking infrared) algorithm was demonstrated by Miller of Perkin-Elmer [44].

The failure of an image algebra based solely on morphological operations to provide a universal image processing algebra is due to its set-theoretic formulation, which rests on the Minkowski addition and subtraction of sets [22]. These operations ignore the linear domain, transformations between different domains (spaces of different sizes and dimensionality), and transformations between different value sets (algebraic structures), e.g., sets consisting of real, complex, or vector valued numbers. The image algebra discussed in this text includes these concepts and extends the morphological operations [1].

The development of image algebra grew out of a need, by the U.S. Air Force Systems Command, for a common image-processing language. Defense contractors do not use a standardized, mathematically rigorous and efficient structure that is specifically designed for image manipulation. Documentation by contractors of algorithms for image processing and rationale underlying algorithm design is often accomplished via word description or analogies that are extremely cumbersome and often ambiguous. The result of these *ad hoc* approaches has been a proliferation of nonstandard notation and increased research and development cost. In response to this chaotic situation, the Air Force Armament Laboratory (AFATL — now known as Wright Laboratory MNGA) of the Air Force Systems Command, in conjunction with the Defense Advanced Research Project Agency (DARPA — now known as the Advanced Research Project Agency or ARPA), supported the early development of image algebra with the intent that the fully developed structure would subsequently form the basis of a common image-processing language. The goal of AFATL was the development of a complete, unified algebraic structure that provides a common mathematical environment for image-processing algorithm development, optimization, comparison, coding, and performance evaluation. The development of this structure proved highly successful, capable of fulfilling the tasks set forth by the government, and is now commonly known as image algebra.

Because of the goals set by the government, the theory of image algebra provides for a language which, if properly implemented as a standard image processing environment, can greatly reduce research and development costs. Since the foundation of this language is purely mathematical and independent of any future computer architecture or language, the longevity of an image algebra standard is assured. Furthermore, savings due to commonality of language and increased productivity could dwarf any reasonable initial investment for adapting image algebra as a standard environment for image processing.

Although commonality of language and cost savings are two major reasons for considering image algebra as a standard language for image processing, there exists a multitude of other reasons for desiring the broad acceptance of image algebra as a component of all image processing development systems. Premier among these is the predictable influence of an image algebra standard on future image processing technology.

In this, it can be compared to the influence on scientific reasoning and the advancement of science due to the replacement of the myriad of different number systems (e.g., Roman, Syrian, Hebrew, Egyptian, Chinese, etc.) by the now common Indo-Arabic notation. Additional benefits provided by the use of image algebra are

- The elemental image algebra operations are small in number, translucent, simple, and provide a method of transforming images that is easily learned and used;

- Image algebra operations and operands provide the capability of expressing all image-to-image transformations;

- Theorems governing image algebra make computer programs based on image algebra notation amenable to both machine dependent and machine independent optimization techniques;

- The algebraic notation provides a deeper understanding of image manipulation operations due to conciseness and brevity of code and is capable of suggesting new techniques;

- The notational adaptability to programming languages allows the substitution of extremely short and concise image algebra expressions for equivalent blocks of code, and therefore increases programmer productivity;

- Image algebra provides a rich mathematical structure that can be exploited to relate image processing problems to other mathematical areas;

- Without image algebra, a programmer will never benefit from the bridge that exists between an image algebra programming language and the multitude of mathematical structures, theorems, and identities that are related to image algebra;

- There is no competing notation that adequately provides all these benefits.

The role of image algebra in computer vision and image processing tasks and theory should not be confused with the government's Ada programming language effort. The goal of the development of the Ada programming language was to provide a single high-order language in which to implement embedded systems. The special architectures being developed nowadays for image processing applications are not often capable of directly executing Ada language programs, often due to support of parallel processing models not accommodated by Ada's tasking mechanism. Hence, most applications designed for such processors are still written in special assembly or microcode languages. Image algebra, on the other hand, provides a level of specification, directly derived from the underlying mathematics on which image processing is based and that is compatible with both sequential and parallel architectures.

Enthusiasm for image algebra must be tempered by the knowledge that image algebra, like any other field of mathematics, will never be a finished product but remain a continuously evolving mathematical theory concerned with the unification of image processing and computer vision tasks. Much of the mathematics associated with image algebra and its implication to computer vision remains largely unchartered territory which awaits discovery. For example, very little work has been done in relating image algebra to computer vision techniques which employ tools from such diverse areas as knowledge representation, graph theory, and surface representation.

Several image algebra programming languages have been developed. These include image algebra Fortran (IAF) [45], an image algebra Ada (IAA) translator [46], image algebra Connection Machine *Lisp [47, 48], an image algebra language (IAL) implementation on transputers [49, 50], and an image algebra C++ class library (`iac++`) [51, 52]. Unfortunately, there is often a tendency among engineers to confuse or equate these languages with image algebra. An image algebra programming language is *not* image algebra, which is a mathematical theory. An image algebra-based programming language typically implements a particular subalgebra of the full image algebra. In addition, simplistic implementations can result in poor computational performance. Restrictions and limitations in implementation are usually due to a combination of factors, the most pertinent being development costs and hardware and software environment constraints. They are not limitations of image algebra, and they should not be confused with the capability of image algebra as a mathematical tool for image manipulation.

Image algebra is a *heterogeneous* or *many-valued algebra* in the sense of Birkhoff and Lipson [53, 1], with multiple sets of operands and operators. Manipulation of images for purposes of image enhancement, analysis, and understanding involves operations not only on images, but also on different types of values and quantities associated with these images. Thus, the basic operands of image algebra are images and the values and quantities associated with these images. Roughly speaking, an image consists of two things, a collection of *points* and a set of *values* associated with these points. Images are therefore endowed with two types of information, namely the spatial relationship of the points, and also some type of numeric or other descriptive information associated with these points. Consequently, the field of image algebra bridges two broad mathematical areas, the theory of point sets and the algebra of value sets, and investigates their interrelationship. In the sections that follow we discuss point and value sets as well as images, templates, and neighborhoods that characterize some of their interrelationships.

1.2. Point Sets

A *point set* is simply a topological space. Thus, a point set consists of two things, a collection of objects called *points* and a topology which provides for such notions as *nearness* of two points, the *connectivity* of a subset of the point set, the *neighborhood* of a point, *boundary points*, and *curves* and *arcs*. Point sets will be denoted by capital bold letters from the end of the alphabet, i.e., $\mathbf{W}$, $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$.

Points (elements of point sets) will be denoted by lower case bold letters from the end of the alphabet, namely $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbf{X}$. Note also that if $\mathbf{x} \in \mathbb{R}^n$, then $\mathbf{x}$ is of form $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where for each $i = 1, 2, \ldots, n$, x_i denotes a real number called the *ith coordinate* of $\mathbf{x}$.

The most common point sets occurring in image processing are discrete subsets of *n*–dimensional Euclidean space $\mathbb{R}^n$ with $n = 1, 2$, or 3 together with the discrete topology. However, other topologies such as the *von Neumann topology* and the *product topology* are also commonly used topologies in computer vision [1].

There is no restriction on the *shape* of the discrete subsets of $\mathbb{R}^n$ used in applications of image algebra to solve vision problems. Point sets can assume arbitrary shapes. In particular, shapes can be rectangular, circular, or snake-like. Some of the more pertinent point sets are the set of integer points $\mathbb{Z}$ (here we view $\mathbb{Z} \subset \mathbb{R}^1$), the *n*–dimensional lattice $\mathbb{Z}^n \subset \mathbb{R}^n$ (i.e., $\mathbb{Z}^n = \mathbb{Z} \times \mathbb{Z} \times \cdots \times \mathbb{Z} = \{\mathbf{x} \in \mathbb{R}^\mathbf{n} : \mathbf{x} = (x_1, \ldots, x_n),\ x_i \in \mathbb{Z}\ for\ i = 1, \ldots, n\}$) with $n = 2$ or $n = 3$, and rectangular subsets of $\mathbb{Z}^2$. Two of the most often encountered rectangular point sets are of form

$$\mathbf{X} = \mathbb{Z}_m \times \mathbb{Z}_n = \left\{(x_1, x_2) \in \mathbb{Z}^2 \,:\, 0 \leq x_1 \leq m-1,\; 0 \leq x_2 \leq n-1\right\},$$

or

$$\mathbf{X} = \mathbb{Z}_m^+ \times \mathbb{Z}_n^+ = \left\{(x_1, x_2) \in \mathbb{Z}^2 \,:\, 1 \leq x_1 \leq m,\; 1 \leq x_2 \leq n\right\}.$$

We follow standard practice and represent these rectangular point sets by listing the points in matrix form. Figure 1.2.1 provides a graphical representation of the point set $\mathbf{X} = \mathbb{Z}_m^+ \times \mathbb{Z}_n^+$.

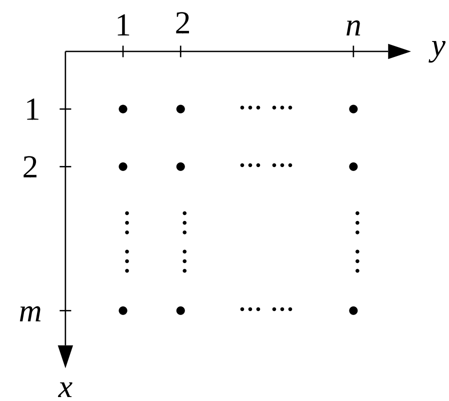

Figure 1.2.1. The rectangular point set $\mathbf{X} = \mathbb{Z}_m \times \mathbb{Z}_n$

Point Operations

As mentioned, some of the more pertinent point sets are discrete subsets of the vector space $\mathbb{R}^n$. These point sets inherit the usual elementary vector space operations. Thus, for example, if $\mathbf{X} \subset \mathbb{Z}^n$ (or $\mathbf{X} \subset \mathbb{R}^n$) and $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathbf{y} = (y_1, \ldots, y_n) \in \mathbf{X}$, then the sum of the points $\mathbf{x}$ and $\mathbf{y}$ is defined as

$$\mathbf{x} + \mathbf{y} = (x_1 + y_1, \ldots, x_n + y_n),$$

while the multiplication and addition of a scalar $k \in \mathbb{Z}$ (or $k \in \mathbb{R}$) and a point $\mathbf{x}$ is given by

$$k \cdot \mathbf{x} = (k \cdot x_1, \ldots, k \cdot x_n)$$

and

$$k + \mathbf{x} = (k + x_1, \ldots, k + x_n),$$

respectively. Point subtraction is also defined in the usual way.

In addition to these standard vector space operations, image algebra also incorporates three basic types of point multiplication. These are the *Hadamard product*, the *cross product* (or *vector product*) for points in $\mathbb{Z}^3$ (or $\mathbb{R}^3$), and the *dot product* which are defined by

$$\mathbf{x} \cdot \mathbf{y} = (x_1 \cdot y_1, \ldots, x_n \cdot y_n),$$
$$\mathbf{x} \times \mathbf{y} = (x_2 \cdot y_3 - x_3 \cdot y_2,\, x_3 \cdot y_1 - x_1 \cdot y_3,\, x_1 \cdot y_2 - x_2 \cdot y_1),$$

and

$$\mathbf{x} \bullet \mathbf{y} = x_1 \cdot y_1 + x_2 \cdot y_2 + \cdots + x_n \cdot y_n,$$

respectively.

Note that the sum of two points, the Hadamard product, and the cross product are binary operations that take as input two points and produce another point. Therefore these operations can be viewed as mappings $\mathbf{X} \times \mathbf{X} \to \mathbf{X}$ whenever $\mathbf{X}$ is closed under these operations. In contrast, the binary operation of dot product is a scalar and not another vector. This provides an example of a mapping $\mathbf{X} \times \mathbf{X} \to \mathbb{F}$, where $\mathbb{F}$ denotes the appropriate field of scalars. Another such mapping, associated with metric spaces, is the distance function $\mathbf{X} \times \mathbf{X} \to \mathbb{R}$ which assigns to each pair of points $\mathbf{x}$ and $\mathbf{y}$ the *distance* from $\mathbf{x}$ to $\mathbf{y}$. The most common distance functions occurring in image processing are the *Euclidean* distance, the *city block* or *diamond* distance, and the *chessboard* distance which are defined by

$$d(\mathbf{x}, \mathbf{y}) = \left[\sum_{k=1}^{n} (x_k - y_k)^2\right]^{\frac{1}{2}},$$

$$\rho(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{n} |x_k - y_k| ,$$

and

$$\delta(\mathbf{x}, \mathbf{y}) = max\{|x_k - y_k| : 1 \le k \le n\},$$

respectively.

Distances can be conveniently computed in terms of the *norm* of a point. The three norms of interest here are derived from the standard L^p norms

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}.$$

The L^∞ norm is given by

$$\|\mathbf{x}\|_\infty = \bigvee_{i=1}^{n} |x_i|,$$

where $\bigvee_{i=1}^{n} |x_i| = max\{|x_1|, \ldots, |x_n|\}$. Specifically, the *Euclidean norm* is given by $\|\mathbf{x}\|_2 = \sqrt{x_1^2 + \cdots + x_n^2}$. Thus, $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2$. Similarly, the city block distance can be computed using the formulation $\rho(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_1$ and the chessboard distance by using $\delta(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_\infty$

Note that the *p*-norm of a point $\mathbf{x}$ is a *unary* operation, namely a function $\| \ \|_p : \mathbf{X} \to \mathbb{R}$. Another assemblage of functions $\mathbf{X} \to \mathbb{R}$ which play a major role in various applications are the projection functions. Given $\mathbf{X} \subset \mathbb{R}^n$, then the *i*th *projection* on $\mathbf{X}$, where $i \in \{1, \ldots, n\}$, is denoted by p_i and defined by $p_i(\mathbf{x}) = x_i$, where x_i denotes the *i*th coordinate of $\mathbf{x}$.

Characteristic functions and neighborhood functions are two of the most frequently occurring unary operations in image processing. In order to define these operations, we need to recall the notion of a *power set* of a set. The power set of a set *S* is defined as the set of all subsets of *S* and is denoted by 2^S. Thus, if $\mathbf{Z}$ is a point set, then $2^{\mathbf{Z}} = \{\mathbf{X} : \mathbf{X} \subset \mathbf{Z}\}$.

Given $\mathbf{X} \in 2^{\mathbf{Z}}$ (i.e., $\mathbf{X} \subset \mathbf{Z}$), then the *characteristic function* associated with $\mathbf{X}$ is the function

$$\chi_{\mathbf{X}} : \mathbf{Z} \to \{0, 1\}$$

defined by

$$\chi_{\mathbf{X}}(\mathbf{z}) = \begin{cases} 1 & if\ \mathbf{z} \in \mathbf{X} \\ 0 & if\ \mathbf{z} \notin \mathbf{X}\,. \end{cases}$$

For a pair of point sets **X** and **Z**, a *neighborhood system for* **X** *in* **Z**, or equivalently, a *neighborhood function from* $\mathbf{X}$ *to* $\mathbf{Z}$, is a function

$$N : \mathbf{X} \to 2^{\mathbf{Z}}\,.$$

It follows that for each point $\mathbf{x} \in \mathbf{X}$, $N(\mathbf{x}) \subset \mathbf{Z}$. The set $N(\mathbf{x})$ is called a *neighborhood for* **x**.

There are two neighborhood functions on subsets of $\mathbb{Z}^2$ which are of particular importance in image processing. These are the *von Neumann* neighborhood and the *Moore* neighborhood. The von Neumann neighborhood $N : \mathbf{X} \to 2^{\mathbb{Z}^2}$ is defined by

$$N(\mathbf{x}) = \{\mathbf{y} : \mathbf{y} = (x_1 \pm j,\, x_2)\ or\ \mathbf{y} = (x_1,\, x_2 \pm k),\ j,\, k \in \{0,\, 1\}\},$$

where $\mathbf{x} = (x_1,\, x_2) \in \mathbf{X} \subset \mathbb{Z}^2$, while the Moore neighborhood $M : \mathbf{X} \to 2^{\mathbb{Z}^2}$ is defined by

$$M(\mathbf{x}) = \{\mathbf{y} : \mathbf{y} = (x_1 \pm j,\, x_2 \pm k),\ j,\, k \in \{0,\, 1\}\}.$$

Figure 1.2.2 provides a pictorial representation of these two neighborhood functions; the hashed center area represents the point **x** and the adjacent cells represent the adjacent points. The von Neumann and Moore neighborhoods are also called the *four neighborhood* and *eight neighborhood*, respectively. They are *local neighborhoods* since they only include the directly adjacent points of a given point.

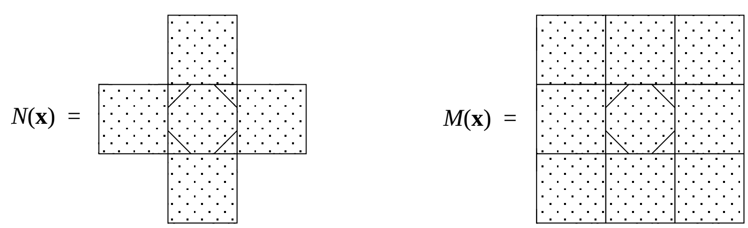

Figure 1.2.2. The von Neumann neighborhood $N(\mathbf{x})$ and the Moore neighborhood $M(\mathbf{x})$ of a point **x**.

There are many other point operations that are useful in expressing computer vision algorithms in succinct algebraic form. For instance, in certain interpolation schemes it becomes necessary to switch from points with real-valued coordinates (floating point coordinates) to corresponding integer-valued coordinate points. One such method uses the induced *floor* operation $\lfloor\ \rfloor : \mathbb{R}^n \to \mathbb{Z}^n$ defined by $\lfloor \mathbf{x} \rfloor = (\lfloor x_1 \rfloor,\, \lfloor x_2 \rfloor,\, \ldots,\, \lfloor x_n \rfloor)$, where $\mathbf{x} = (x_1,\, x_2,\, \ldots,\, x_n) \in \mathbb{R}^n$ and $\lfloor x_i \rfloor \in \mathbb{Z}$ denotes the largest integer *less than or equal to* x_i (i.e., $\lfloor x_i \rfloor \le x_i$ and if $k \in \mathbb{Z}$ with $k \le x_i$, then $k \le \lfloor x_i \rfloor$).

Summary of Point Operations

We summarize some of the more pertinent point operations. Some image algebra implementations such as iac++ provide many additional point operations [54].

Binary operations. Let $\mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{y} = (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n$, and $\mathbf{z} = (z_1, z_2, \ldots, z_m) \in \mathbb{R}^m$.

addition	$\mathbf{x} + \mathbf{y} = (x_1 + y_1, \ldots, x_n + y_n)$
subtraction	$\mathbf{x} - \mathbf{y} = (x_1 - y_1, \ldots, x_n - y_n)$
multiplication	$\mathbf{x} \cdot \mathbf{y} = (x_1 y_1, \ldots, x_n y_n)$
division	$\mathbf{x}/\mathbf{y} = (x_1/y_1, \ldots, x_n/y_n)$
supremum	$sup(\mathbf{x}, \mathbf{y}) = (x_1 \vee y_1, \ldots, x_n \vee y_n)$
infimum	$inf(\mathbf{x}, \mathbf{y}) = (x_1 \wedge y_1, \ldots, x_n \wedge y_n)$
dot product	$\mathbf{x} \bullet \mathbf{y} = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$
cross product $(n = 3)$	$\mathbf{x} \times \mathbf{y} = (x_2 y_3 - x_3 y_2, x_3 y_1 - x_1 y_3, x_1 y_2 - x_2 y_1)$
concatenation	$\mathbf{x}\hat{}\mathbf{z} = (x_1, \ldots, x_n, z_1, \ldots, z_m)$
scalar operations	$k \gamma \mathbf{x} = (k \gamma x_1, \ldots, k \gamma x_n)$, where $\gamma \in \{+, -, *, \vee, \wedge\}$

Unary operations. In the following let $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$.

negation	$-\mathbf{x} = (-x_1, \ldots, -x_n)$
ceiling	$\lceil \mathbf{x} \rceil = (\lceil x_1 \rceil, \ldots, \lceil x_n \rceil)$
floor	$\lfloor \mathbf{x} \rfloor = (\lfloor x_1 \rfloor, \ldots, \lfloor x_n \rfloor)$
rounding	$[\mathbf{x}] = ([x_1], \ldots, [x_n])$
projection	$p_i(\mathbf{x}) = x_i$
sum	$\Sigma \mathbf{x} = x_1 + x_2 + \cdots + x_n$
product	$\Pi \mathbf{x} = x_1 x_2 \cdots x_n$
maximum	$\vee \mathbf{x} = x_1 \vee x_2 \vee \cdots \vee x_n$
minimum	$\wedge \mathbf{x} = x_1 \wedge x_2 \wedge \cdots \wedge x_n$
Euclidean norm	$\lVert \mathbf{x} \rVert_2 = \sqrt{x_1^2 + \cdots + x_n^2}$
L^1 ***norm***	$\lVert \mathbf{x} \rVert_1 = \lvert x_1 \rvert + \lvert x_2 \rvert + \cdots + \lvert x_n \rvert$
L^∞ ***norm***	$\lVert \mathbf{x} \rVert_\infty = \lvert x_1 \rvert \vee \lvert x_2 \rvert \vee \cdots \vee \lvert x_n \rvert$
dimension	$dim(\mathbf{x}) = n$
neighborhood	$N(\mathbf{x}) \subset \mathbb{R}^n$
characteristic function	$\chi_{\mathbf{X}}(\mathbf{z}) = \begin{cases} 1 & if\ \mathbf{z} \in \mathbf{X} \\ 0 & if\ \mathbf{z} \notin \mathbf{X} \end{cases}$

It is important to note that several of the above unary operations are special instances of spatial transformations $\mathbf{X} \rightarrow \mathbf{Y}$. Spatial transforms play a vital role in many image processing and computer vision tasks.

In the above summary we only considered points with real- or integer-valued coordinates. Points of other spaces have their own induced operations. For example, typical operations on points of $\mathbf{X} = (\mathbb{Z}_2)^n$ (i.e., Boolean-valued points) are the usual logical operations of AND, OR, XOR, and complementation.

Point Set Operations

Point arithmetic leads in a natural way to the notion of set arithmetic. Given a vector space $\mathbf{Z}$, then for $\mathbf{X}, \mathbf{Y} \in 2^{\mathbf{Z}}$ (i.e., $\mathbf{X}, \mathbf{Y} \subset \mathbf{Z}$) and an arbitrary point $\mathbf{p} \in \mathbf{Z}$ we define the following arithmetic operations:

addition	$\mathbf{X} + \mathbf{Y} = \{\mathbf{x} + \mathbf{y} : \mathbf{x} \in \mathbf{X}\ and\ \mathbf{y} \in \mathbf{Y}\}$
subtraction	$\mathbf{X} - \mathbf{Y} = \{\mathbf{x} - \mathbf{y} : \mathbf{x} \in \mathbf{X}\ and\ \mathbf{y} \in \mathbf{Y}\}$
point addition	$\mathbf{X} + \mathbf{p} = \{\mathbf{x} + \mathbf{p} : \mathbf{x} \in \mathbf{X}\}$
point subtraction	$\mathbf{X} - \mathbf{p} = \{\mathbf{x} - \mathbf{p} : \mathbf{x} \in \mathbf{X}\}$

Another set of operations on $2^{\mathbf{Z}}$ are the usual set operations of *union*, *intersection*, *set difference* (or *relative complement*), *symmetric difference*, and *Cartesian product* as defined below.

union	$\mathbf{X} \cup \mathbf{Y} = \{\mathbf{z} : \mathbf{z} \in \mathbf{X} \; or \; \mathbf{z} \in \mathbf{Y}\}$
intersection	$\mathbf{X} \cap \mathbf{Y} = \{\mathbf{z} : \mathbf{z} \in \mathbf{X} \; and \; \mathbf{z} \in \mathbf{Y}\}$
set difference	$\mathbf{X} \backslash \mathbf{Y} = \{\mathbf{z} : \mathbf{z} \in \mathbf{X} \; and \; \mathbf{z} \notin \mathbf{Y}\}$
symmetric difference	$\mathbf{X} \triangle \mathbf{Y} = \{\mathbf{z} : \mathbf{z} \in \mathbf{X} \cup \mathbf{Y} \; and \; \mathbf{z} \notin \mathbf{X} \cap \mathbf{Y}\}$
Cartesian product	$\mathbf{X} \times \mathbf{Y} = \{(\mathbf{x}, \mathbf{y}) : \mathbf{x} \in \mathbf{X} \; and \; \mathbf{y} \in \mathbf{Y}\}$

Note that with the exception of the Cartesian product, the set obtained for each of the above operations is again an element of $2^{\mathbf{Z}}$.

Another common set theoretic operation is *set complementation.* For $\mathbf{X} \in 2^{\mathbf{Z}}$, the *complement of* $\mathbf{X}$ is denoted by $\tilde{\mathbf{X}}$, and defined as $\tilde{\mathbf{X}} = \{\mathbf{z} : \mathbf{z} \in \mathbf{Z} \; and \; \mathbf{z} \notin \mathbf{X}\}$. In contrast to the binary set operations defined above, set complementation is a unary operation. However, complementation can be computed in terms of the binary operation of set difference by observing that $\tilde{\mathbf{X}} = \mathbf{Z} \backslash \mathbf{X}$.

In addition to complementation there are various other common unary operations which play a major role in algorithm development using image algebra. Among these is the *cardinality* of a set which, when applied to a finite point set, yields the number of elements in the set, and the *choice* function which, when applied to a set, selects a randomly chosen point from the set. The cardinality of a set **X** will be denoted by *card*(**X**). Note that

$$card : 2^{\mathbf{Z}} \to \mathbb{N} \;\; \left(\text{for all finite elements of } 2^{\mathbf{Z}}\right),$$

while

$$choice : 2^{\mathbf{Z}} \to \mathbf{Z}\,.$$

That is, $card(\mathbf{X}) \in \mathbb{N}$ and $choice(\mathbf{X}) = \mathbf{x}$, where **x** is some randomly chosen element of **X**.

As was the case for operations on points, algebraic operations on point sets are too numerous to discuss at length in a short treatise as this. Therefore, we again only summarize some of the more frequently occurring unary operations.

Summary of Unary Point Set Operations

In the following $\mathbf{X} \subset \mathbb{R}^n$.

negation	$-\mathbf{X} = \{-\mathbf{x} : \mathbf{x} \in \mathbf{X}\}$
complementation	$\tilde{\mathbf{X}} = \{\mathbf{z} : \mathbf{z} \in \mathbf{Z} \; and \; \mathbf{z} \notin \mathbf{X}\}$
supremum	$sup(\mathbf{X}) \;\; (for\ finite\ point\ set\ \mathbf{X})$
infimum	$inf(\mathbf{X}) \;\; (for\ finite\ point\ set\ \mathbf{X})$
choice function	$choice(\mathbf{X}) \in \mathbf{X} \;\; (randomly\ chosen\ element)$
cardinality	$card(\mathbf{X}) = the\ cardinality\ of\ \mathbf{X}$

The interpretation of $sup(\mathbf{X})$ is as follows. Suppose $\mathbf{X}$ is finite, say $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$. Then $sup(\mathbf{X}) = sup(\ldots sup(sup(sup(\mathbf{x}_1, \mathbf{x}_2), \mathbf{x}_3), \mathbf{x}_4), \ldots, \mathbf{x}_n)$, where $sup(\mathbf{x}_i, \mathbf{x}_j)$ denotes the binary operation of the supremum of two points defined earlier. Equivalently, if $\mathbf{x}_i = (x_i, y_i)$ for $i = 1, \ldots, k$, then $sup(\mathbf{X}) = (x_1 \vee x_2 \vee \cdots \vee x_k,\, y_1 \vee y_2 \vee \cdots \vee y_k)$. More generally, $sup(\mathbf{X})$ is defined to be the *least upper bound* of **X** (if it exists). The infimum of **X** is interpreted in a similar fashion.

If **X** is finite and has a total order, then we also define the *maximum* and *minimum* of **X**, denoted by $\bigvee \mathbf{X}$ and $\bigwedge \mathbf{X}$, respectively, as follows. Suppose $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$ and $\mathbf{x}_1 \prec \mathbf{x}_2 \prec \cdots \prec \mathbf{x}_k$, where the symbol $\prec$ denotes the particular total order on **X**.

Then $\bigvee \mathbf{X} = \mathbf{x}_k$ and $\bigwedge \mathbf{X} = \mathbf{x}_1$. The most commonly used order for a subset $\mathbf{X}$ of $\mathbb{Z}^2$ is the row scanning order. Note also that in contrast to the supremum or infimum, the maximum and minimum of a (finite totally ordered) set is always a member of the set.

1.3. Value Sets

A *heterogeneous algebra* is a collection of nonempty sets of possibly different types of elements together with a set of finitary operations which provide the rules of combining various elements in order to form a new element. For a precise definition of a heterogeneous algebra we refer the reader to Ritter [1]. Note that the collection of point sets, points, and scalars together with the operations described in the previous section form a heterogeneous algebra.

A *homogeneous algebra* is a heterogeneous algebra with only one set of operands. In other words, a homogeneous algebra is simply a set together with a finite number of operations. Homogeneous algebras will be referred to as *value sets* and will be denoted by capital blackboard font letters, e.g., $\mathbb{E}$, $\mathbb{F}$, and $\mathbb{G}$. There are several value sets that occur more often than others in digital image processing. These are the set of integers, real numbers (floating point numbers), the complex numbers, binary numbers of fixed length *k*, the extended real numbers (which include the symbols $+\infty$ and/or $-\infty$), and the extended non–negative real numbers. We denote these sets by $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{C}$, $\mathbb{Z}_{2^k}$, $\mathbb{R}_{+\infty} = \mathbb{R} \cup \{+\infty\}$, $\mathbb{R}_{-\infty} = \mathbb{R} \cup \{-\infty\}$, $\mathbb{R}_{\pm\infty} = \mathbb{R} \cup \{+\infty, -\infty\}$, and $\mathbb{R}^{\geqq 0}_{\infty} = \mathbb{R}^+ \cup \{0, +\infty\}$, respectively, where the symbol $\mathbb{R}^+$ denotes the set of positive real numbers.

Operations on Value Sets

The operations on and between elements of a given value set $\mathbb{F}$ are the usual elementary operations associated with $\mathbb{F}$. Thus, if $\mathbb{F} \in \{\mathbb{Z}, \mathbb{R}, \mathbb{Z}_{2^k}\}$, then the binary operations are the usual arithmetic and logic operations of addition, multiplication, and maximum, and the complementary operations of subtraction, division, and minimum. If $\mathbb{F} = \mathbb{C}$, then the binary operations are addition, subtraction, multiplication, and division. Similarly, we allow the usual elementary unary operations associated with these sets such as the absolute value, conjugation, as well as trigonometric, logarithmic and exponential functions as these are available in all higher-level scientific programming languages.

For the set $\mathbb{R}_{\pm\infty}$ we need to extend the arithmetic and logic operations of $\mathbb{R}$ as follows:

$$
\begin{aligned}
&a + (-\infty) = (-\infty) + a = -\infty && a \in \mathbb{R}_{-\infty} \\
&a + \infty = \infty + a = \infty && a \in \mathbb{R}_{\infty} \\
&(-\infty) + \infty = \infty + (-\infty) = -\infty && \\
&a \vee (-\infty) = (-\infty) \vee a = a && a \in \mathbb{R}_{\pm\infty}
\end{aligned}
$$

Note that the element $-\infty$ acts as a null element in the system $(\mathbb{R}_{\pm\infty}, \vee, +)$ if we view the operation + as multiplication and the operation $\vee$ as addition. The same cannot be said about the element ∞ in the system $(\mathbb{R}_{\pm\infty}, \wedge, +)$ since $(-\infty)+\infty = \infty+(-\infty) = -\infty$. In order to remedy this situation we define the *dual* structure $(\mathbb{R}_{\pm\infty}, \wedge, +')$ of $(\mathbb{R}_{\pm\infty}, \vee, +)$ as follows:

$$
\begin{aligned}
&a +' b = a + b && a, b \in \mathbb{R} \\
&a +' (-\infty) = (-\infty) +' a = -\infty && a \in \mathbb{R}_{-\infty} \\
&a +' \infty = \infty +' a = \infty && a \in \mathbb{R}_{\infty} \\
&(-\infty) +' \infty = \infty +' (-\infty) = \infty && \\
&a \wedge \infty = \infty \wedge a = a && a \in \mathbb{R}_{\pm\infty}
\end{aligned}
$$

Now the element $+\infty$ acts as a null element in the system $(\mathbb{R}_{\pm\infty}, \wedge, +')$. Observe, however, that the dual additions $+$ and $+'$ introduce an asymmetry between $-\infty$ and $+\infty$. The resultant structure $(\mathbb{R}_{\pm\infty}, \vee, \wedge, +, +')$ is known as a *bounded lattice ordered group* [1].

Dual structures provide for the notion of dual elements. For each $r \in \mathbb{R}_{\pm\infty}$ we define its *dual* or *conjugate* r^* by $r^* = -r$, where $-(-\infty) = \infty$. The following duality laws are a direct consequence of this definition:

(1) $(r^*)^* = r$

(2) $(r \wedge t)^* = r^* \vee t^*$ *and* $(r \vee t)^* = r^* \wedge t^*$.

Closely related to the *additive* bounded lattice ordered group described above is the *multiplicative* bounded lattice ordered group $\left(\mathbb{R}_{\infty}^{\geq 0}, \vee, \wedge, \times, \times'\right)$. Here the dual $\times'$ of ordinary multiplication is defined as

$$a \times' b = a \times b \;\; \forall\, a, b \in \mathbb{R}^{\geq 0} = \mathbb{R}^{+} \cup \{0\}$$

with both multiplicative operations extended as follows:

$$\begin{array}{ll} a \times \infty = \infty \times a = \infty & a \in \mathbb{R}_{\infty}^{+} \\ a \times' \infty = \infty \times' a = \infty & a \in \mathbb{R}_{\infty}^{+} \\ 0 \times \infty = \infty \times 0 = 0 & \\ 0 \times' \infty = \infty \times' 0 = \infty & \end{array}$$

Hence, the element 0 acts as a null element in the system $\left(\mathbb{R}_{\infty}^{\geq \mathbf{0}}, \vee, \times\right)$ and the element $+\infty$ acts as a null element in the system $\left(\mathbb{R}_{\infty}^{\geq \mathbf{0}}, \wedge, \times'\right)$. The conjugate r^* of an element $r \in \mathbb{R}_{\infty}^{\geq \mathbf{0}}$ of this value set is defined by

$$r^* \equiv \begin{cases} r^{-1} & \text{if } r \in \mathbb{R}^{+} \\ 0 & \text{if } r = +\infty \\ +\infty & \text{if } r = 0 \end{cases}.$$

Another algebraic structure with duality which is of interest in image algebra is the value set $\left(\mathbb{Z}_2^*, \vee, \wedge, \tilde{+}, \tilde{+}'\right)$, where $\mathbb{Z}_2^* = (\mathbb{Z}_2)_{\pm\infty} = \mathbb{Z}_2 \cup \{\infty, -\infty\} = \{0, 1, -\infty, \infty\}$. The logical operations $\vee$ and $\wedge$ are the usual binary operations of max (or) and min (and), respectively, while the dual additive operations $\tilde{+}$ and $\tilde{+}'$ are defined by the tables shown in Figure 1.3.1.

$\tilde{+}$	0	1	$-\infty$	∞
0	1	0	∞	$-\infty$
1	0	1	∞	$-\infty$
$-\infty$	∞	∞	∞	∞
∞	$-\infty$	$-\infty$	∞	$-\infty$

$\tilde{+}'$	0	1	$-\infty$	∞
0	1	0	∞	$-\infty$
1	0	1	∞	$-\infty$
$-\infty$	∞	∞	∞	$-\infty$
∞	$-\infty$	$-\infty$	$-\infty$	∞

Figure 1.3.1. The dual additive operations $\tilde{+}$ and $\tilde{+}'$.

Note that the addition $\tilde{+}$ (as well as $\tilde{+}'$) restricted to $\mathbb{Z}_2 = \{0, 1\}$ is the exclusive or operation *xor* and computes the values for the truth table of the biconditional statement $p \leftrightarrow q$ (i.e., p if and only if q).

The operations on the value set $\mathbb{Z}_2^*$ can be easily generalized to its *k*-fold Cartesian product $\mathbb{Z}_{2^k}^* = \mathbb{Z}_2^* \times \mathbb{Z}_2^* \times \cdots \times \mathbb{Z}_2^*$. Specifically, if $m = (m_1, \ldots, m_k) \in \mathbb{Z}_{2^k}^*$ and $n = (n_1, \ldots, n_k) \in \mathbb{Z}_{2^k}^*$, where $m_i, n_i \in \mathbb{Z}_2^*$ for $i = 1, \ldots, k$, then $m\tilde{+}n = \big(m_1\tilde{+}n_1, \ldots, m_k\tilde{+}n_k\big)$.

The addition $\tilde{+}$ should not be confused with the usual addition $mod2^k$ on $\mathbb{Z}_{2^k}$. In fact, for $m, n \in \mathbb{Z}_{2^k}$ $m\tilde{+}n = \big((m_1 + n_1)', \ldots, (m_k + n_k)'\big)$, where

$$(m_i + n_i)' = \begin{cases} 0 & if \ (m_i + n_i)mod2 = 1 \\ 1 & if \ (m_i + n_i)mod2 = 0\,. \end{cases}$$

Many point sets are also value sets. For example, the point set $\mathbf{X} = \mathbb{R}^n$ is a metric space as well as a vector space with the usual operation of vector addition. Thus, $(\mathbb{R}^n, +)$, where the symbol "$+$" denotes vector addition, will at various times be used both as a point set and as a value set. Confusion as to usage will not arise as usage should be clear from the discussion.

Summary of Pertinent Numeric Value Sets

In order to focus attention on the value sets most often used in this treatise we provide a listing of their algebraic structures:

(a) $(\mathbb{R}, \vee, \wedge, +, \cdot)$

(b) $(\mathbb{C}, +, \cdot)$

(c) $(\mathbb{Z}, \vee, \wedge, +, \cdot)$

(d) $(\mathbb{Z}_{2^k}, \vee, \wedge, +, \cdot)$

(e) $(\mathbb{R}_{\pm\infty}, \vee, \wedge, +, +')$

(f) $\big(\mathbb{R}_{\infty}^{\geq 0}, \vee, \wedge, \times, \times'\big)$

(g) $\Big(\mathbb{Z}_2^*, \vee, \wedge, \tilde{+}, \tilde{+}'\Big)$

In contrast to structure c, the addition and multiplication in structure d is addition and multiplication $mod2^k$.

These listed structures represent the pertinent global structures. In various applications only certain subalgebras of these algebras are used. For example, the subalgebras $(\mathbb{R}_{-\infty}, \vee, +)$ and $(\mathbb{R}_{+\infty}, \wedge, +')$ of $(\mathbb{R}_{\pm\infty}, \vee, \wedge, +, +')$ play special roles in morphological processing. Similarly, the subalgebra $(\mathbb{N}, \vee, \wedge, +)$ of $(\mathbb{Z}, \vee, \wedge, +, \cdot)$, where $\mathbb{N} = \{0, 1, 2, \ldots, n, \ldots\}$, is the only pertinent applicable algebra in certain cases.

The complementary binary operations, whenever they exist, are assumed to be part of the structures. Thus, for example, subtraction and division which can be defined in terms of addition and multiplication, respectively, are assumed to be part of $(\mathbb{R}, \vee, \wedge, +, \cdot)$.

Value Set Operators

As for point sets, given a value set $\mathbb{F}$, the operations on $2^{\mathbb{F}}$ are again the usual operations of union, intersection, set difference, etc. If, in addition, $\mathbb{F}$ is a lattice, then the operations of infimum and supremum are also included. A brief summary of value set operators is given below.

For the following operations assume that A, $B \in 2^{\mathbb{F}}$ for some value set $\mathbb{F}$.

union	$A \cup B = \{c : c \in A \; or \; c \in B\}$
intersection	$A \cap B = \{c : c \in A \; and \; c \in B\}$
set difference	$A \backslash B = \{c : c \in A \; and \; c \notin B\}$
symmetric difference	$A \triangle B = \{c : c \in A \cup B \; and \; c \notin A \cap B\}$
Cartesian product	$A \times B = \{(a, b) : a \in A \; and \; b \in B\}$
choice function	$choice(A) \in A$
cardinality	$card(A) = cardinality \; of \; A$
supremum	$sup(A) = supremum \; of \; A$
infimum	$inf(A) = infimum \; of \; A$

1.4. Images

The primary operands in image algebra are images, templates, and neighborhoods. Of these three classes of operands, images are the most fundamental since templates and neighborhoods can be viewed as special cases of the general concept of an image. In order to provide a mathematically rigorous definition of an image that covers the plethora of objects called an “image” in signal processing and image understanding, we define an image in general terms, with a minimum of specification. In the following we use the notation A^B to denote the set of all functions $B \to A$ (i.e., $A^B = \{f : f \; is \; a \; function \; from \; B \; to \; A\}$).

Definition: Let $\mathbb{F}$ be a value set and $\mathbf{X}$ a point set. An $\mathbb{F}$-*valued image on* $\mathbf{X}$ is any element of $\mathbb{F}^{\mathbf{X}}$. Given an $\mathbb{F}$–valued image $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$ (i.e., $\mathbf{a} : \mathbf{X} \to \mathbb{F}$), then $\mathbb{F}$ is called the *set of possible range values* of **a** and $\mathbf{X}$ the *spatial domain* of **a**.

It is often convenient to let the graph of an image $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$ represent **a**. The graph of an image is also referred to as the *data structure representation* of the image. Given the data structure representation $\mathbf{a} = \{(\mathbf{x}, \mathbf{a}(\mathbf{x})) : \mathbf{x} \in \mathbf{X}\}$, then an element $(\mathbf{x}, \mathbf{a}(\mathbf{x}))$ of the data structure is called a *picture element* or *pixel*. The first coordinate **x** of a pixel is called the *pixel location* or *image point*, and the second coordinate **a**(**x**) is called the *pixel value* of **a** at location **x**.

The above definition of an image covers all mathematical images on topological spaces with range in an algebraic system. Requiring **X** to be a topological space provides us with the notion of nearness of pixels. Since **X** is not directly specified we may substitute any space required for the analysis of an image or imposed by a particular sensor and scene. For example, $\mathbf{X}$ could be a subset of $\mathbb{Z}^3$ or $\mathbb{R}^3$ with $\mathbf{x} \in \mathbf{X}$ of form $\mathbf{x} = (x, y, t)$, where the first coordinates (x, y) denote spatial location and *t* a time variable.

Similarly, replacing the unspecified value set $\mathbb{F}$ with $\mathbb{Z}_{2^k}$ or $\mathbb{F} = (\mathbb{Z}_{2^k}, \mathbb{Z}_{2^m}, \mathbb{Z}_{2^n})$ provides us with digital integer-valued and digital vector-valued images, respectively. An implication of these observations is that our image definition also characterizes any type of discrete or continuous *physical image*.

Induced Operations on Images

Operations on and between $\mathbb{F}$-valued images are the natural induced operations of the algebraic system $\mathbb{F}$. For example, if γ is a binary operation on $\mathbb{F}$, then γ induces a binary operation — again denoted by γ — on $\mathbb{F}^{\mathbf{X}}$ defined as follows:

Let $\mathbf{a}, \mathbf{b} \in \mathbb{F}^{\mathbf{X}}$. Then

$$\mathbf{a}\gamma\mathbf{b} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x})\gamma\mathbf{b}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}.$$

For example, suppose $\mathbf{a}$, $\mathbf{b} \in \mathbb{R}^{\mathbf{X}}$ and our value set is the algebraic structure of the real numbers $(\mathbb{R}, +, \cdot, \vee, \wedge)$. Replacing γ by the binary operations $+$, $\cdot$, $\vee$, and $\wedge$ we obtain the *basic* binary operations

$$\begin{aligned}
\mathbf{a} + \mathbf{b} &= \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) + \mathbf{b}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}, \\
\mathbf{a} \cdot \mathbf{b} &= \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \cdot \mathbf{b}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}, \\
\mathbf{a} \vee \mathbf{b} &= \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \vee \mathbf{b}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\},
\end{aligned}$$

and

$$\mathbf{a} \wedge \mathbf{b} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \wedge \mathbf{b}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}$$

on real-valued images. Obviously, all four operations are commutative and associative.

In addition to the binary operation between images, the binary operation γ on $\mathbb{F}$ also induces the following scalar operations on images:

For $k \in \mathbb{F}$ and $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$,

$$k\gamma\mathbf{a} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = k\gamma\mathbf{a}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}$$

and

$$\mathbf{a}\gamma k = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x})\gamma k\,,\ \mathbf{x} \in \mathbf{X}\}.$$

Thus, for $k \in \mathbb{R}$, we obtain the following scalar multiplication and addition of real-valued images:

$$k \cdot \mathbf{a} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = k \cdot \mathbf{a}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}$$

and

$$k + \mathbf{a} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = k + \mathbf{a}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}.$$

It follows from the commutativity of real numbers that,

$$k \cdot \mathbf{a} = \mathbf{a} \cdot k \text{ and } k + \mathbf{a} = \mathbf{a} + k.$$

Although much of image processing is accomplished using real-, integer-, binary-, or complex-valued images, many higher-level vision tasks require manipulation of vector and set-valued images. A set-valued image is of form $\mathbf{a} : \mathbf{X} \to 2^{\mathbb{F}}$. Here the underlying value set is $\left(2^{\mathbb{F}}, \cup, \cap, \tilde{\ }\right)$, where the tilde symbol denotes complementation. Hence, the operations on set-valued images are those induced by the Boolean algebra of the value set. For example, if $\mathbf{a}, \mathbf{b} \in \left(2^{\mathbb{F}}\right)^{\mathbf{X}}$, then

$$\begin{aligned}
\mathbf{a} \cup \mathbf{b} &= \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \cup \mathbf{b}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}\,, \\
\mathbf{a} \cap \mathbf{b} &= \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \cap \mathbf{b}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}\,,
\end{aligned}$$

and

$$\tilde{\mathbf{a}} = \left\{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \widetilde{\mathbf{a}(\mathbf{x})},\ \mathbf{x} \in \mathbf{X}\right\},$$

where $\widetilde{\mathbf{a}(\mathbf{x})} = \mathbb{F} \backslash \mathbf{a}(\mathbf{x})$.

The operation of complementation is, of course, a unary operation. A particularly useful unary operation on images which is induced by a binary operation on a value set is known as the *global reduce operation*. More precisely, if γ is an associative and commutative binary operation on $\mathbb{F}$ and $\mathbf{X}$ is finite, say $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, then γ induces a unary operation

$$\Gamma : \mathbb{F}^{\mathbf{X}} \to \mathbb{F}$$

called the *global reduce operation induced by* γ, which is defined as

$$\Gamma \mathbf{a} = \Gamma_{\mathbf{x} \in \mathbf{X}} \mathbf{a}(\mathbf{x}) = \Gamma_{k=1}^{n} \mathbf{a}(\mathbf{x}_k) = \mathbf{a}(\mathbf{x}_1) \gamma \mathbf{a}(\mathbf{x}_2) \gamma \cdots \gamma \mathbf{a}(\mathbf{x}_n) .$$

Thus, for example, if $\mathbb{F} = \mathbb{R}$ and γ is the operation of addition ($\gamma = +$), then $\Gamma = \Sigma$ and

$$\sum \mathbf{a} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{a}(\mathbf{x}) = \mathbf{a}(\mathbf{x}_1) + \mathbf{a}(\mathbf{x}_2) + \cdots + \mathbf{a}(\mathbf{x}_n).$$

In all, the value set $(\mathbb{R}, +, \cdot, \vee, \wedge)$ provides for four basic global reduce operations, namely $\sum \mathbf{a}$, $\prod \mathbf{a}$, $\bigvee \mathbf{a}$, and $\bigwedge \mathbf{a}$.

Induced Unary Operations and Functional Composition

In the previous section we discussed unary operations on elements of $\mathbb{F}^{\mathbf{X}}$ induced by a binary operation γ on $\mathbb{F}$. Typically, however, unary image operations are induced directly by unary operations on $\mathbb{F}$. Given a unary operation $f : \mathbb{F} \to \mathbb{F}$, then the induced unary operation $\mathbb{F}^{\mathbf{X}} \to \mathbb{F}^{\mathbf{X}}$ is again denoted by *f* and is defined by

$$f(\mathbf{a}) = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = f(\mathbf{a}(\mathbf{x})), \ \mathbf{x} \in \mathbf{X}\}.$$

Note that in this definition we view the composition $f \circ \mathbf{a}$ as a unary operation on $\mathbb{F}^{\mathbf{X}}$ with operand **a**. This subtle distinction has the important consequence that *f* is viewed as a unary operation — namely a function from $\mathbb{F}^{\mathbf{X}}$ to $\mathbb{F}^{\mathbf{X}}$ — and **a** as an *argument* of *f*. For example, substituting $\mathbb{R}$ for $\mathbb{F}$ and the sine function $sin : \mathbb{R} \to \mathbb{R}$ for *f*, we obtain the induced operation $sin : \mathbb{R}^{\mathbf{X}} \to \mathbb{R}^{\mathbf{X}}$, where

$$sin(\mathbf{a}) = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = sin(\mathbf{a}(\mathbf{x})), \ \mathbf{x} \in \mathbf{X}\}.$$

As another example, consider the characteristic function

$$\chi_{\geq k}(r) = \begin{cases} 1 & \text{if } r \geq k \\ 0 & \text{otherwise.} \end{cases}$$

Then for any $\mathbf{a} \in \mathbb{R}^{\mathbf{X}}$, $\chi_{\geq k}(\mathbf{a})$ is the Boolean (two-valued) image on $\mathbf{X}$ with value 1 at location $\mathbf{x}$ if $\mathbf{a}(\mathbf{x}) \geq k$ and value 0 if $\mathbf{a}(\mathbf{x}) < k$. An obvious application of this operation is the thresholding of an image. Given a floating point image **a** and using the characteristic function

$$\chi_{[j,k]}(r) = \begin{cases} 1 & \text{if } j \leq r \leq k \\ 0 & \text{otherwise,} \end{cases}$$

then the image **b** in the image algebra expression

$$\mathbf{b} := \mathbf{a} \cdot \chi_{[j,k]}(\mathbf{a})$$

is given by

$$\mathbf{b} = \{(\mathbf{x}, \mathbf{b}(\mathbf{x})) : \mathbf{b}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \text{ if } j \leq \mathbf{a}(\mathbf{x}) \leq k, \text{ otherwise } \mathbf{b}(\mathbf{x}) = 0\}.$$

The unary operations on an image $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$ discussed thus far have resulted either in a scalar (an element of $\mathbb{F}$) by use of the global reduction operation, or another $\mathbb{F}$-valued image by use of the composition $f \circ \mathbf{a} = f(\mathbf{a})$. More generally, given a function $f : \mathbb{F} \to \mathbb{G}$, then the composition $f \circ \mathbf{a}$ provides for a unary operation which changes an $\mathbb{F}$-valued image into a $\mathbb{G}$-valued image $f(\mathbf{a})$. Taking the same viewpoint, but using a function *f* between spatial domains instead, provides a scheme for realizing naturally induced operations for spatial manipulation of image data. In particular, if $f : \mathbf{Y} \to \mathbf{X}$ and $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$, then we define the induced image $\mathbf{a} \circ f \in \mathbb{F}^{\mathbf{Y}}$ by

$$\mathbf{a} \circ f = \{(\mathbf{y}, \mathbf{a}(f(\mathbf{y}))) : \mathbf{y} \in \mathbf{Y}\}.$$

Thus, the operation defined by the above equation transforms an $\mathbb{F}$-valued image defined over the space $\mathbf{X}$ into an $\mathbb{F}$-valued image defined over the space $\mathbf{Y}$.

Examples of spatial based image transformations are affine and perspective transforms. For instance, suppose $\mathbf{a} \in \mathbb{R}^{\mathbf{X}}$, where $\mathbf{X} \subset \mathbb{Z}^2$ is a rectangular $m \times n$ array. If $1 \leq k \leq \frac{m}{2}$ and $f : \mathbf{X} \to \mathbf{X}$ is defined as

$$f(x, y) = \begin{cases} (x, y) & \text{if } k \leq x \\ (2k - x, y) & \text{if } x < k \end{cases},$$

then $\mathbf{a} \circ f$ is a one sided reflection of **a** across the line $x = k$. Further examples are provided by several of the algorithms presented in this text.

Simple shifts of an image can be achieved by using either a spatial transformation or *point addition*. In particular, given $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$, $\mathbf{X} \subset \mathbb{Z}^2$, and $\mathbf{y} \in \mathbb{Z}^2$, we define a shift of **a** by **y** as

$$\mathbf{a} + \mathbf{y} = \{(\mathbf{z}, \mathbf{b}(\mathbf{z})) : \mathbf{b}(\mathbf{z}) = \mathbf{a}(\mathbf{z} - \mathbf{y}),\ \mathbf{z} - \mathbf{y} \in \mathbf{X}\}.$$

Note that $\mathbf{a} + \mathbf{y}$ is an image on $\mathbf{X} + \mathbf{y}$ since $\mathbf{z} - \mathbf{y} \in \mathbf{X} \iff \mathbf{z} \in \mathbf{X} + \mathbf{y}$, which provides for the equivalent formulation

$$\mathbf{a} + \mathbf{y} = \{(\mathbf{z}, \mathbf{b}(\mathbf{z})) : \mathbf{b}(\mathbf{z}) = \mathbf{a}(\mathbf{z} - \mathbf{y}),\ \mathbf{z} \in \mathbf{X} + \mathbf{y}\}.$$

Of course, one could just as well define a spatial transformation $f : \mathbf{X} + \mathbf{y} \to \mathbf{X}$ by $f(\mathbf{z}) = \mathbf{z} - \mathbf{y}$ in order to obtain the identical shifted image $\mathbf{a} + \mathbf{y} = \mathbf{a} \circ f$.

Another simple unary image operation that can be defined in terms of a spatial map is *image transposition*. Given an image $\mathbf{a} \in \mathbb{F}^{\mathbb{Z}_m \times \mathbb{Z}_n}$, then the *transpose* of **a**, denoted by $\mathbf{a}'$, is defined as $\mathbf{a}' \equiv \mathbf{a} \circ f$, where $f : \mathbb{Z}_n \times \mathbb{Z}_m \to \mathbb{Z}_m \times \mathbb{Z}_n$ is given by $f(x, y) = (y, x)$.

Binary Operations Induced by Unary Operations

Various unary operations image operations induced by functions $f : \mathbb{F} \to \mathbb{F}$ can be generalized to binary operations on $\mathbb{F}^{\mathbf{X}}$. As a simple illustration, consider the exponentiation function $f : \mathbb{R}^{\geq 0} \to \mathbb{R}$ defined by $f(r) = r^k$, where *k* denotes some non-negative real number. Then *f* induces the exponentiation operation

$$\mathbf{a}^k = \Big\{(\mathbf{x}, \mathbf{b}(\mathbf{x})) : \mathbf{b}(\mathbf{x}) = [\mathbf{a}(\mathbf{x})]^k,\ \mathbf{x} \in \mathbf{X}\Big\},$$

where **a** is a non-negative real-valued image on **X**. We may extend this operation to a binary image operation as follows: if $\mathbf{a}$, $\mathbf{b} \in \left(\mathbb{R}^{\geq 0}\right)^{\mathbf{X}}$, then

$$\mathbf{a}^{\mathbf{b}} = \left\{(\mathbf{x}, \mathbf{c}(\mathbf{x})) \,:\, \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x})^{\mathbf{b}(\mathbf{x})},\ \mathbf{x} \in \mathbf{X}\right\}.$$

The notion of exponentiation can be extended to negative valued images as long as we follow the rules of arithmetic and restrict this binary operation to those pairs of real-valued images for which $\mathbf{a}(\mathbf{x})^{\mathbf{b}(\mathbf{x})} \in \mathbb{R}\ \forall \mathbf{x} \in \mathbf{X}$. This avoids creation of complex, undefined, and indeterminate pixel values such as $(-1)^{\frac{1}{2}}$, $\frac{1}{0^2}$, and 0^0, respectively. However, there is one exception to these rules of standard arithmetic. The algebra of images provides for the existence of *pseudo inverses*. For $\mathbf{a} \in \mathbb{R}^{\mathbf{X}}$, the *pseudo inverse* of **a**, which for reason of simplicity is denoted by $\mathbf{a}^{-1}$ is defined as

$$\mathbf{a}^{-1} = \left\{(\mathbf{x}, \mathbf{b}(\mathbf{x})) \,:\, \mathbf{b}(\mathbf{x}) = \frac{1}{\mathbf{a}(\mathbf{x})} \text{ if } \mathbf{a}(\mathbf{x}) \neq 0 \text{ otherwise } \mathbf{b}(\mathbf{x}) = 0\right\}.$$

Note that if some pixel values of **a** are zero, then $\mathbf{a} \cdot \mathbf{a}^{-1} \neq \mathbf{1}$, where **1** denotes unit image all of whose pixel values are 1. However, the equality $\mathbf{a} \cdot \mathbf{a}^{-1} \cdot \mathbf{a} = \mathbf{a}$ always holds. Hence the name "pseudo inverse."

The inverse of exponentiation is defined in the usual way by taking logarithms. Specifically,

$$log_{\mathbf{b}}\mathbf{a} = \left\{(\mathbf{x}, \mathbf{c}(\mathbf{x})) \,:\, \mathbf{c}(\mathbf{x}) = log_{\mathbf{b}(\mathbf{x})}\mathbf{a}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\right\}.$$

As for real numbers, $log_{\mathbf{b}}\mathbf{a}$ is defined only for positive images; i.e., $\mathbf{a}$, $\mathbf{b} \in (\mathbb{R}^{+})^{\mathbf{X}}$.

Another set of examples of binary operations induced by unary operations are the characteristic functions for comparing two images. For $\mathbf{a}$, $\mathbf{b} \in \mathbb{R}^{\mathbf{X}}$ we define

$$\begin{aligned}
\chi_{\leq \mathbf{b}}(\mathbf{a}) &= \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = 1 \text{ if } \mathbf{a}(\mathbf{x}) \leq \mathbf{b}(\mathbf{x}), \text{ otherwise } \mathbf{c}(\mathbf{x}) = 0\} \\
\chi_{< \mathbf{b}}(\mathbf{a}) &= \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = 1 \text{ if } \mathbf{a}(\mathbf{x}) < \mathbf{b}(\mathbf{x}), \text{ otherwise } \mathbf{c}(\mathbf{x}) = 0\} \\
\chi_{= \mathbf{b}}(\mathbf{a}) &= \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = 1 \text{ if } \mathbf{a}(\mathbf{x}) = \mathbf{b}(\mathbf{x}), \text{ otherwise } \mathbf{c}(\mathbf{x}) = 0\} \\
\chi_{\geq \mathbf{b}}(\mathbf{a}) &= \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = 1 \text{ if } \mathbf{a}(\mathbf{x}) \geq \mathbf{b}(\mathbf{x}), \text{ otherwise } \mathbf{c}(\mathbf{x}) = 0\} \\
\chi_{> \mathbf{b}}(\mathbf{a}) &= \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = 1 \text{ if } \mathbf{a}(\mathbf{x}) > \mathbf{b}(\mathbf{x}), \text{ otherwise } \mathbf{c}(\mathbf{x}) = 0\} \\
\chi_{\neq \mathbf{b}}(\mathbf{a}) &= \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = 1 \text{ if } \mathbf{a}(\mathbf{x}) \neq \mathbf{b}(\mathbf{x}), \text{ otherwise } \mathbf{c}(\mathbf{x}) = 0\}.
\end{aligned}$$

Functional Specification of Image Operations

The basic concepts of elementary function theory provide the underlying foundation of a functional specification of image processing techniques. This is a direct consequence of viewing images as functions. The most elementary concepts of function theory are the notions of domain, range, restriction, and extension of a function.

Image restrictions and extensions are used to restrict images to regions of particular interest and to embed images into larger images, respectively. Employing standard mathematical notation, the *restriction* of $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$ to a subset **Z** of **X** is denoted by $\mathbf{a}|_{\mathbf{Z}}$, and defined by

$$\mathbf{a}|_{\mathbf{Z}} \equiv \mathbf{a} \cap (\mathbf{Z} \times \mathbb{F}) = \{(\mathbf{x}, \mathbf{a}(\mathbf{x})) \,:\, \mathbf{x} \in \mathbf{Z}\}.$$

Thus, $\mathbf{a}|_{\mathbf{Z}} \in \mathbb{F}^{\mathbf{Z}}$. In practice, the user may specify **Z** explicitly by providing bounds for the coordinates of the points of **Z**.

There is nothing magical about restricting **a** to a subset **Z** of its domain **X**. We can just as well define restrictions of images to subsets of the range values. Specifically, if $S \subset \mathbb{F}$ and $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$, then the *restriction* of **a** to *S* is denoted by $\mathbf{a}||_S$ and defined as

$$\mathbf{a}||_S \equiv \mathbf{a} \cap (\mathbf{X} \times S).$$

In terms of the pixel representation of $\mathbf{a}||_S$ we have $\mathbf{a}||_S = \{(\mathbf{x}, \mathbf{a}(\mathbf{x})) : \mathbf{a}(\mathbf{x}) \in S\}$. The double-bar notation is used to focus attention on the fact that the restriction is applied to the second coordinate of $\mathbf{a} \subset \mathbf{X} \times \mathbb{F}$.

Image restrictions in terms of subsets of the value set $\mathbb{F}$ is an extremely useful concept in computer vision as many image processing tasks are restricted to image domains over which the image values satisfy certain properties. Of course, one can always write this type of restriction in terms of a first coordinate restriction by setting $\mathbf{Z} = \{\mathbf{x} \in \mathbf{X} : \mathbf{a}(\mathbf{x}) \in S\}$ so that $\mathbf{a}||_S = \mathbf{a}|_{\mathbf{Z}}$. However, writing a program statement such as $\mathbf{b} := \mathbf{a}|_{\mathbf{Z}}$ is of little value since **Z** is implicitly specified in terms of *S*; i.e., **Z** must be determined in terms of the property "$\mathbf{a}(\mathbf{x}) \in S$." Thus, **Z** would have to be precomputed, adding to the computational overhead as well as increased code. In contrast, direct restriction of the second coordinate values to an explicitly specified set *S* avoids these problems and provides for easier implementation.

As mentioned, restrictions to the range set provide a useful tool for expressing various algorithmic procedures. For instance, if $\mathbf{a} \in \mathbb{R}^{\mathbf{X}}$ and *S* is the interval $(k, \infty) \subset \mathbb{R}$, where *k* denotes some given threshold value, then $\mathbf{a}||_{(k,\infty)}$ denotes the image **a** restricted to all those points of **X** where **a**(**x**) exceeds the value *k*. In order to reduce notation, we define $\mathbf{a}||_{>k} \equiv \mathbf{a}||_{(k,\infty)}$. Similarly,

$$\mathbf{a}||_{\geq k} \equiv \mathbf{a}||_{[k,\infty)},\ \ \mathbf{a}||_{<k} \equiv \mathbf{a}||_{(-\infty,k)},\ \ \mathbf{a}||_{k} \equiv \mathbf{a}||_{\{k\}},\ \text{ and }\ \mathbf{a}||_{\leq k} \equiv \mathbf{a}||_{(-\infty,k]}.$$

As in the case of characteristic functions, a more general form of range restriction is given when *S* corresponds to a set-valued image $S \in \left(2^{\mathbb{F}}\right)^{\mathbf{X}}$; i.e., $S(\mathbf{x}) \subset \mathbb{F}\ \ \forall \mathbf{x} \in \mathbf{X}$. In this case we define

$$\mathbf{a}||_S = \{(\mathbf{x}, \mathbf{a}(\mathbf{x})) : \mathbf{a}(\mathbf{x}) \in S(\mathbf{x})\}.$$

For example, for $\mathbf{a}, \mathbf{b} \in \mathbb{R}^{\mathbf{X}}$ we define

$$\begin{aligned}
&\mathbf{a}||_{\leq \mathbf{b}} \equiv \{(\mathbf{x}, \mathbf{a}(\mathbf{x})) : \mathbf{a}(\mathbf{x}) \leq \mathbf{b}(\mathbf{x})\},\ \ \mathbf{a}||_{<\mathbf{b}} \equiv \{(\mathbf{x}, \mathbf{a}(\mathbf{x})) : \mathbf{a}(\mathbf{x}) < \mathbf{b}(\mathbf{x})\},\\
&\mathbf{a}||_{\geq \mathbf{b}} \equiv \{(\mathbf{x}, \mathbf{a}(\mathbf{x})) : \mathbf{a}(\mathbf{x}) \geq \mathbf{b}(\mathbf{x})\},\ \ \mathbf{a}||_{>\mathbf{b}} \equiv \{(\mathbf{x}, \mathbf{a}(\mathbf{x})) : \mathbf{a}(\mathbf{x}) > \mathbf{b}(\mathbf{x})\},\\
&\mathbf{a}||_{\mathbf{b}} \equiv \{(\mathbf{x}, \mathbf{a}(\mathbf{x})) : \mathbf{a}(\mathbf{x}) = \mathbf{b}(\mathbf{x})\},\ \ \mathbf{a}||_{\neq \mathbf{b}} \equiv \{(\mathbf{x}, \mathbf{a}(\mathbf{x})) : \mathbf{a}(\mathbf{x}) \neq \mathbf{b}(\mathbf{x})\}.
\end{aligned}$$

Combining the concepts of first and second coordinate (domain and range) restrictions provides the general definition of an image restriction. If $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$, $\mathbf{Z} \subset \mathbf{X}$, and $S \subset \mathbb{F}$, then the *restriction of* **a** *to* **Z** *and S* is defined as

$$\mathbf{a}|_{(\mathbf{Z},S)} = \mathbf{a} \cap (\mathbf{Z} \times S).$$

It follows that $\mathbf{a}|_{(\mathbf{Z},S)} = \{(\mathbf{x}, \mathbf{a}(\mathbf{x})) : \mathbf{x} \in \mathbf{Z} \text{ and } \mathbf{a}(\mathbf{x}) \in S\}$, $\mathbf{a}|_{(\mathbf{X},S)} = \mathbf{a}||_S$, and $\mathbf{a}|_{(\mathbf{Z},\mathbb{F})} = \mathbf{a}|_{\mathbf{Z}}$.

The *extension* of $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$ to $\mathbf{b} \in \mathbb{F}^{\mathbf{Y}}$ on **Y**, where **X** and **Y** are subsets of the same topological space, is denoted by $\mathbf{a}|^{\mathbf{b}}$ and defined by

$$\mathbf{a}|^{\mathbf{b}}(\mathbf{x}) = \begin{cases} \mathbf{a}(\mathbf{x}) & \text{if } \mathbf{x} \in \mathbf{X} \\ \mathbf{b}(\mathbf{x}) & \text{if } \mathbf{x} \in \mathbf{Y} \backslash \mathbf{X}. \end{cases}$$

In actual practice, the user will have to specify the function **b**.

Two of the most important concepts associated with a function are its domain and range. In the field of image understanding, it is convenient to view these concepts as functions that map images to sets associated with certain image properties. Specifically, we view the concept of range as a function

$$range : \mathbb{F}^{\mathbf{X}} \to 2^{\mathbb{F}}$$

defined by $range(\mathbf{a}) = \{r \in \mathbb{F} \,:\, r = \mathbf{a}(\mathbf{x}) \; for \; some \; \mathbf{x} \in \mathbf{X}\}$.

Similarly, the concept of domain is viewed as the function

$$domain : \mathbb{F}^{\mathbf{X}}|_{(2^{\mathbf{X}} \times 2^{\mathbb{F}})} \to 2^{\mathbf{X}},$$

where

$$\mathbb{F}^{\mathbf{X}}|_{(2^{\mathbf{X}} \times 2^{\mathbb{F}})} = \left\{\mathbf{b} \,:\, \mathbf{b} = \mathbf{a}|_{(\mathbf{Z},S)},\ \mathbf{a} \in \mathbb{F}^{\mathbf{X}},\ \mathbf{Z} \in 2^{\mathbf{X}},\ S \in 2^{\mathbb{F}}\right\}$$

and *domain* is defined by

$$domain(\mathbf{b}) = \left\{\mathbf{x} \in \mathbf{X} \,:\, \mathbf{a}|_{(\mathbf{Z},S)}(\mathbf{x}) = \mathbf{b}(\mathbf{x}) = r \; for \; some \; r \in \mathbb{F}\right\}.$$

These mapping can be used to extract point sets and value sets from regions of images of particular interest. For example, the statement

$$s := domain(\mathbf{a}||_{>k})$$

yields the *set* of all points (pixel locations) where **a**(**x**) exceeds *k*, namely $s = \{\mathbf{x} \in \mathbf{X} \,:\, \mathbf{a}(\mathbf{x}) > k\}$. The statement

$$s := range(\mathbf{a}||_{>k})$$

on the other hand, results in a subset of $\mathbb{R}$ instead of **X**.

Closely related to spatial transformations and functional composition is the notion of *image concatenation*. Concatenation serves as a tool for simplifying algorithm code, adding translucency to code, and to provide a link to the usual block notion used in matrix algebra. Given $\mathbf{a} \in \mathbb{F}^{\mathbb{Z}_m \times \mathbb{Z}_k}$ and $\mathbf{b} \in \mathbb{F}^{\mathbb{Z}_m \times \mathbb{Z}_n}$, then the *row-order concatenation of* **a** *with* **b** is denoted by $(\mathbf{a} \mid \mathbf{b})$ and is defined as

$$(\mathbf{a} \mid \mathbf{b}) \equiv \mathbf{a}|^{\mathbf{b}+(0,k)}.$$

Note that $(\mathbf{a} \mid \mathbf{b}) \in \mathbb{F}^{\mathbb{Z}_m \times \mathbb{Z}_{n+k}}$.

Assuming the correct dimensionality in the first coordinate, concatenation of any number of images is defined inductively using the formula $(\mathbf{a} \mid \mathbf{b}|\mathbf{c}) = ((\mathbf{a} \mid \mathbf{b})|\mathbf{c})$ so that in general we have

$$(\mathbf{a}_1|\mathbf{a}_2| \cdots |\mathbf{a}_l) = ((\mathbf{a}_1|\mathbf{a}_2| \cdots |\mathbf{a}_{l-1})|\mathbf{a}_l).$$

Column-order concatenation can be defined in a similar manner or by simple transposition; i.e.,

$$\begin{pmatrix} \mathbf{a}_1 \\ - \\ \mathbf{a}_2 \\ - \\ \vdots \\ - \\ \mathbf{a}_l \end{pmatrix} = (\mathbf{a}_1|\mathbf{a}_2| \cdots |\mathbf{a}_l)'.$$

Multi-Valued Image Operations

Although general image operations described in the previous sections apply to both single and multi-valued images as long as there is no specific value type associated with the generic value set $\mathbb{F}$, there exist a large number of multi-valued image operations that are quite distinct from single-valued image operations. As the general theory of multi-valued image operations is beyond the scope of this treatise, we shall restrict our attention to some specific operations on vector-valued images while referring the reader interested in more intricate details to Ritter [1]. However, it is important to realize that vector-valued images are a special cases of multi-valued images.

If $\mathbb{F} = \mathbb{R}^n$ and $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$, then **a(x)** is a vector of form $\mathbf{a}(\mathbf{x}) = (\mathbf{a}_1(\mathbf{x}), \ldots, \mathbf{a}_n(\mathbf{x}))$ where for each $i = 1, \ldots, n$, $\mathbf{a}_i(\mathbf{x}) \in \mathbb{R}$. Thus, an image $\mathbf{a} \in (\mathbb{R}^n)^{\mathbf{X}}$ is of form $\mathbf{a} = (\mathbf{a}_1, \ldots, \mathbf{a}_n)$ and with each *vector value* **a(x)** there are associated *n* real values $\mathbf{a}_i(\mathbf{x})$.

Real-valued image operations generalize to the usual vector operations on $(\mathbb{R}^n)^{\mathbf{X}}$. In particular, if $\mathbf{a}, \mathbf{b} \in (\mathbb{R}^n)^{\mathbf{X}}$, then

$$\begin{aligned}
\mathbf{a} + \mathbf{b} &= (\mathbf{a}_1 + \mathbf{b}_1, \ldots, \mathbf{a}_n + \mathbf{b}_n) \\
\mathbf{a} \cdot \mathbf{b} &= (\mathbf{a}_1 \cdot \mathbf{b}_1, \ldots, \mathbf{a}_n \cdot \mathbf{b}_n) \\
\mathbf{a} \vee \mathbf{b} &= (\mathbf{a}_1 \vee \mathbf{b}_1, \ldots, \mathbf{a}_n \vee \mathbf{b}_n) \\
\mathbf{a} \wedge \mathbf{b} &= (\mathbf{a}_1 \wedge \mathbf{b}_1, \ldots, \mathbf{a}_n \wedge \mathbf{b}_n).
\end{aligned}$$

If $\mathbf{r} = (r_1, \ldots, r_n) \in \mathbb{R}^n$, then we also have

$$\begin{aligned}
\mathbf{r} + \mathbf{a} &= (r_1 + \mathbf{a}_1, \ldots, r_n + \mathbf{a}_n)\,, \\
\mathbf{r} \cdot \mathbf{a} &= (r_1 \cdot \mathbf{a}_1, \ldots, r_n \cdot \mathbf{a}_n)\,,
\end{aligned}$$

etc. In the special case where $\mathbf{r} = (r, r, \ldots, r)$, we simply use the scalar $r \in \mathbb{R}$ and define $r + \mathbf{a} \equiv \mathbf{r} + \mathbf{a}$, $r \cdot \mathbf{a} \equiv \mathbf{r} \cdot \mathbf{a}$, and so on.

As before, binary operations on multi-valued images are induced by the corresponding binary operation $\gamma : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ on the value set $\mathbb{R}^n$. It turns out to be useful to generalize this concept by replacing the binary operation γ by a sequence of binary operations $\gamma_j : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, where $j = 1, \ldots, n$, and defining

$$\mathbf{a}\gamma\mathbf{b} \equiv (\mathbf{a}\gamma_1\mathbf{b}, \mathbf{a}\gamma_2\mathbf{b}, \ldots, \mathbf{a}\gamma_n\mathbf{b})\,.$$

For example, if $\gamma_j : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is defined by

$$(x_1, \ldots, x_n)\gamma_j(y_1, \ldots, y_n) = max\{x_i \vee y_j \,:\, 1 \le i \le j\}\,,$$

then for $\mathbf{a}, \mathbf{b} \in (\mathbb{R}^n)^{\mathbf{X}}$ and $\mathbf{c} = \mathbf{a}\gamma\mathbf{b}$, the components of $\mathbf{c}(\mathbf{x}) = (\mathbf{c}_1(\mathbf{x}), \ldots, \mathbf{c}_n(\mathbf{x}))$ have values

$$\mathbf{c}_j(\mathbf{x}) = \mathbf{a}(\mathbf{x})\gamma_j\mathbf{b}(\mathbf{x}) = max\{\mathbf{a}_i(\mathbf{x}) \vee \mathbf{a}_j(\mathbf{x}) \,:\, 1 \le i \le j\}$$

for $j = 1, \ldots, n$.

As another example, suppose γ_1 and γ_2 are two binary operations $\mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ defined by

$$(x_1, x_2)\gamma_1(y_1, y_2) = x_1 y_1 - x_2 y_2$$

and

$$(x_1, x_2)\gamma_2(y_1, y_2) = x_1 y_2 + x_2 y_1\,,$$

respectively. Now if $\mathbf{a}, \mathbf{b} \in \left(\mathbb{R}^2\right)^{\mathbf{X}}$ represent two complex-valued images, then the product$\mathbf{c} = \mathbf{a}\gamma\mathbf{b}$ represents pointwise complex multiplication, namely

$$\mathbf{c}(\mathbf{x}) = (\mathbf{a}_1(\mathbf{x})\mathbf{b}_1(\mathbf{x}) - \mathbf{a}_2(\mathbf{x})\mathbf{b}_2(\mathbf{x}), \, \mathbf{a}_1(\mathbf{x})\mathbf{b}_2(\mathbf{x}) + \mathbf{a}_2(\mathbf{x})\mathbf{b}_1(\mathbf{x}))\,.$$

Basic operations on single and multi-valued images can be combined to form image processing operations of arbitrary complexity. Two such operations that have proven to be extremely useful in processing real vector-valued images are the *winner take all j*th-coordinate maximum and minimum of two images. Specifically, if $\mathbf{a}, \mathbf{b} \in (\mathbb{R}^n)^{\mathbf{X}}$, then the *jth-coordinate maximum* of $\mathbf{a}$ and $\mathbf{b}$ is defined as

$$\mathbf{a} \vee |_j \mathbf{b} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) \, : \, \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \text{ if } \mathbf{a}_j(\mathbf{x}) \geq \mathbf{b}_j(\mathbf{x}), \, \text{otherwise } \mathbf{c}(\mathbf{x}) = \mathbf{b}(\mathbf{x})\},$$

while the *jth-coordinate minimum* is defined as

$$\mathbf{a} \wedge |_j \mathbf{b} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) \, : \, \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \text{ if } \mathbf{a}_j(\mathbf{x}) \leq \mathbf{b}_j(\mathbf{x}), \, \text{otherwise } \mathbf{c}(\mathbf{x}) = \mathbf{b}(\mathbf{x})\}.$$

Unary operations on vector-valued images are defined in a similar componentwise fashion. Given a function $f : \mathbb{R} \to \mathbb{R}$, then *f* induces a function $\mathbb{R}^n \to \mathbb{R}^n$, again denoted by *f*, which is defined by

$$f(x_1, \, x_2, \, \ldots \, , \, x_n) \equiv (f(x_1), \, f(x_2), \, \ldots \, , \, f(x_n)).$$

These functions provide for one type of unary operations on vector-valued images. In particular, if $\mathbf{a} = (\mathbf{a}_1, \, \mathbf{a}_2, \, \ldots \, , \, \mathbf{a}_n) \in (\mathbb{R}^n)^{\mathbf{X}}$, then

$$f(\mathbf{a}) \equiv f \circ \mathbf{a} = (f(\mathbf{a}_1), \, f(\mathbf{a}_2), \, \ldots \, , \, f(\mathbf{a}_n)).$$

Thus, if $f = sin : \mathbb{R} \to \mathbb{R}$, then

$$sin(\mathbf{a}) = (sin(\mathbf{a}_1), \, \ldots \, , \, sin(\mathbf{a}_n))\,.$$

Similarly, if $f = \chi_{\geq k}$, then

$$\chi_{\geq k}(\mathbf{a}) = \left(\chi_{\geq k}(\mathbf{a}_1), \, \ldots \, , \, \chi_{\geq k}(\mathbf{a}_n)\right)\,.$$

Any function $f : \mathbb{R}^n \to \mathbb{R}^n$ gives rise to a sequence of functions $f_j = p_j \circ f : \mathbb{R}^n \to \mathbb{R}$, where $j = 1, \, \ldots \, , \, n$. Conversely, given a sequence of functions $f_j : \mathbb{R}^n \to \mathbb{R}$, where $j = 1, \, \ldots \, , \, n$, then we can define a function $f : \mathbb{R}^n \to \mathbb{R}^n$ by

$$f(\mathbf{x}) \equiv (f_1(\mathbf{x}), \, f_2(\mathbf{x}), \, \ldots \, , \, f_n(\mathbf{x})),$$

where $\mathbf{x} = (x_1, \, \ldots \, , \, x_n) \in \mathbb{R}^n$. Such functions provide for a more complex type of unary image operations since by definition

$$f(\mathbf{a}) = (f_1(\mathbf{a}), \, \ldots \, , f_m(\mathbf{a})) = \{(\mathbf{x}, \mathbf{b}(\mathbf{x})) \, : \, \mathbf{b}(\mathbf{x}) = (f_1(\mathbf{a}(\mathbf{x})), \, \ldots \, , f_m(\mathbf{a}(\mathbf{x})))\},$$

which means that the construction of each new coordinate depends on *all* the original coordinates. To provide a specific example, define $f_1 : \mathbb{R}^2 \to \mathbb{R}$ by $f_1(x, y) = sin(x) + cosh(y)$ and $f_2 : \mathbb{R}^2 \to \mathbb{R}$ by $f_2(x, y) = cos(x) + sinh(y)$. Then the induced function

$f : (\mathbb{R}^2)^{\mathbf{X}} \to (\mathbb{R}^2)^{\mathbf{X}}$ given by $f = (f_1, f_2)$. Applying f to an image $\mathbf{a} \in (\mathbb{R}^2)^{\mathbf{X}}$ results in the image

$$f(\mathbf{a}) = \{(\mathbf{x}, \mathbf{b}(\mathbf{x})) \,:\, \mathbf{b}(\mathbf{x}) = (sin(\mathbf{a}_1(\mathbf{x})) + cosh(\mathbf{a}_2(\mathbf{x})),\, cos(\mathbf{a}_1(\mathbf{x})) + sinh(\mathbf{a}_2(\mathbf{x}))),\ \mathbf{x} \in \mathbf{X}\,\}.$$

Thus, if we represent complex numbers as points in $\mathbb{R}^2$ and $\mathbf{a}$ denotes a complex-valued image, then $f(\mathbf{a})$ is a pointwise application of the complex sine function.

Global reduce operations are also applied componentwise. For example, if $\mathbf{a} \in (\mathbb{R}^n)^{\mathbf{X}}$, and $k = card(\mathbf{X})$, then

$$\begin{aligned}\Sigma\mathbf{a} &= (\Sigma\mathbf{a}_1, \ldots, \Sigma\mathbf{a}_n) \\ &= \left(\sum_{j=1}^{k} \mathbf{a}_1(\mathbf{x}_j), \ldots, \sum_{j=1}^{k} \mathbf{a}_n(\mathbf{x}_j)\right) \in \mathbb{R}^n .\end{aligned}$$

In contrast, the summation $\sum_{i=1}^{n} \mathbf{a}_i = \sum_{i=1}^{n} p_i(\mathbf{a}) \in \mathbb{R}^{\mathbf{X}}$ since each $\mathbf{a}_i \in \mathbb{R}^{\mathbf{X}}$. Note that the projection function p_i is a unary operation $(\mathbb{R}^n)^{\mathbf{X}} \to \mathbb{R}^{\mathbf{X}}$.

Similarly,

$$\begin{aligned}\vee\,\mathbf{a} &= (\vee\mathbf{a}_1, \ldots, \vee\mathbf{a}_n), \\ \wedge\,\mathbf{a} &= (\wedge\mathbf{a}_1, \ldots, \wedge\mathbf{a}_n),\end{aligned}$$

and

$$\Pi\mathbf{a} = (\Pi\mathbf{a}_1, \ldots, \Pi\mathbf{a}_n).$$

Summary of Image Operations

The lists below summarize some of the more significant image operations.

Binary image operations.

It is assumed that only appropriately valued images are employed for the operations listed below. Thus, for the operations of maximum and minimum apply to real- or integer-valued images but not complex-valued images. Similarly, union and intersection apply only to set-valued images.

generic	$\mathbf{a}\gamma\mathbf{b} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x})\gamma\mathbf{b}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}$
addition	$\mathbf{a} + \mathbf{b} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) + \mathbf{b}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}$
multiplication	$\mathbf{a} \cdot \mathbf{b} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \cdot \mathbf{b}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}$
maximum	$\mathbf{a} \vee \mathbf{b} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \vee \mathbf{b}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}$
minimum	$\mathbf{a} \wedge \mathbf{b} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \wedge \mathbf{b}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}$
scalar addition	$k + \mathbf{a} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = k + \mathbf{a}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}$
scalar multiplication	$k \cdot \mathbf{a} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = k \cdot \mathbf{a}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}$
point addition	$\mathbf{a} + \mathbf{y} = \{(\mathbf{z}, \mathbf{b}(\mathbf{z})) : \mathbf{b}(\mathbf{z}) = \mathbf{a}(\mathbf{z} - \mathbf{y}),\ \mathbf{z} \in \mathbf{X} + \mathbf{y}\,\}$
union	$\mathbf{a} \cup \mathbf{b} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \cup \mathbf{b}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}$
intersection	$\mathbf{a} \cap \mathbf{b} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \cap \mathbf{b}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}$
exponentiation	$\mathbf{a}^{\mathbf{b}} = \left\{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \mathbf{a}(\mathbf{x})^{\mathbf{b}(\mathbf{x})},\ \mathbf{x} \in \mathbf{X}\right\}$
logarithm	$log_{\mathbf{b}}\mathbf{a} = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = log_{\mathbf{b}(\mathbf{x})}\mathbf{a}(\mathbf{x}),\ \mathbf{x} \in \mathbf{X}\}$

concatenation	$(\mathbf{a}\vert\mathbf{b}) = \mathbf{a}\vert^{\mathbf{b}+(0,k)},\ \mathbf{a} \in \mathbb{F}^{\mathbb{Z}_m \times \mathbb{Z}_k},\ \mathbf{a} \in \mathbb{F}^{\mathbb{Z}_m \times \mathbb{Z}_n}$
concatenation	$\begin{pmatrix} \mathbf{a} \\ - \\ \mathbf{b} \end{pmatrix} = (\mathbf{a}\vert\mathbf{b})'$
characteristics	

$$\chi_{\leq \mathbf{b}}(\mathbf{a}) = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \ \mathbf{c}(\mathbf{x}) = 1 \text{ if } \mathbf{a}(\mathbf{x}) \leq \mathbf{b}(\mathbf{x}), \text{ otherwise } \mathbf{c}(\mathbf{x}) = 0\}$$
$$\chi_{< \mathbf{b}}(\mathbf{a}) = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \ \mathbf{c}(\mathbf{x}) = 1 \text{ if } \mathbf{a}(\mathbf{x}) < \mathbf{b}(\mathbf{x}), \text{ otherwise } \mathbf{c}(\mathbf{x}) = 0\}$$
$$\chi_{= \mathbf{b}}(\mathbf{a}) = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \ \mathbf{c}(\mathbf{x}) = 1 \text{ if } \mathbf{a}(\mathbf{x}) = \mathbf{b}(\mathbf{x}), \text{ otherwise } \mathbf{c}(\mathbf{x}) = 0\}$$
$$\chi_{\geq \mathbf{b}}(\mathbf{a}) = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \ \mathbf{c}(\mathbf{x}) = 1 \text{ if } \mathbf{a}(\mathbf{x}) \geq \mathbf{b}(\mathbf{x}), \text{ otherwise } \mathbf{c}(\mathbf{x}) = 0\}$$
$$\chi_{> \mathbf{b}}(\mathbf{a}) = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \ \mathbf{c}(\mathbf{x}) = 1 \text{ if } \mathbf{a}(\mathbf{x}) > \mathbf{b}(\mathbf{x}), \text{ otherwise } \mathbf{c}(\mathbf{x}) = 0\}$$
$$\chi_{\neq \mathbf{b}}(\mathbf{a}) = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \ \mathbf{c}(\mathbf{x}) = 1 \text{ if } \mathbf{a}(\mathbf{x}) \neq \mathbf{b}(\mathbf{x}), \text{ otherwise } \mathbf{c}(\mathbf{x}) = 0\}$$

Whenever $\mathbf{b}$ is a constant image, say $\mathbf{b} = k$ (i.e., $\mathbf{b}(\mathbf{x}) = k \ \ \forall \mathbf{x} \in \mathbf{X}$), then we simply write $\mathbf{a}^k$ for $\mathbf{a}^{\mathbf{b}}$ and $log_k\mathbf{a}$ for $log_{\mathbf{b}}\mathbf{a}$. Similarly, we have $k+\mathbf{a}$, $\chi_{\leq k}(\mathbf{a})$, $\chi_{<k}(\mathbf{a})$, etc.

Unary image operations.

As in the case of binary operations, we again assume that only appropriately valued images are employed for the operations listed below.

value transform	$f \circ \mathbf{a} = f(\mathbf{a}) = \{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = f(\mathbf{a}(\mathbf{x})),\ \mathbf{x} \in \mathbf{X}\}$
spatial transform	$\mathbf{a} \circ f = \{(\mathbf{y}, \mathbf{a}(f(\mathbf{y}))) : \mathbf{y} \in \mathbf{Y}\}$
domain restriction	$\mathbf{a}\vert_{\mathbf{Z}} = \{(\mathbf{x}, \mathbf{a}(\mathbf{x})) : \mathbf{x} \in \mathbf{Z}\}$
range restriction	$\mathbf{a}\Vert_S = \{(\mathbf{x}, \mathbf{a}(\mathbf{x})) : \mathbf{a}(\mathbf{x}) \in S\}$
extension	$\mathbf{a}\vert^{\mathbf{b}} = \left\{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \begin{cases} \mathbf{a}(\mathbf{x}) & \text{if } \mathbf{x} \in \mathbf{X} \\ \mathbf{b}(\mathbf{x}) & \text{if } \mathbf{x} \in \mathbf{Y} \backslash \mathbf{X} \end{cases}\right\}$
domain	$domain(\mathbf{a}) = \{\mathbf{x} \in \mathbf{X} : \exists\, r \in \mathbb{F}\ s.t.\ \mathbf{a}(\mathbf{x}) = r\}$
range	$range(\mathbf{a}) = \{r \in \mathbb{F} : \exists\, \mathbf{x} \in \mathbf{X}\ s.t.\ r = \mathbf{a}(\mathbf{x})\}$
generic reduction	$\Gamma\, \mathbf{a} = \mathbf{a}(\mathbf{x}_1)\gamma\mathbf{a}(\mathbf{x}_2)\gamma \cdots \gamma\mathbf{a}(\mathbf{x}_n)$
image sum	$\sum \mathbf{a} = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{a}(\mathbf{x}) = \mathbf{a}(\mathbf{x}_1) + \mathbf{a}(\mathbf{x}_2) + \cdots + \mathbf{a}(\mathbf{x}_n)$
image product	$\prod \mathbf{a} = \prod_{\mathbf{x} \in \mathbf{X}} \mathbf{a}(\mathbf{x}) = \mathbf{a}(\mathbf{x}_1) \cdot \mathbf{a}(\mathbf{x}_2) \cdot \cdots \cdot \mathbf{a}(\mathbf{x}_n)$
image maximum	$\bigvee \mathbf{a} = \bigvee_{\mathbf{x} \in \mathbf{X}} \mathbf{a}(\mathbf{x}) = \mathbf{a}(\mathbf{x}_1) \vee \mathbf{a}(\mathbf{x}_2) \vee \cdots \vee \mathbf{a}(\mathbf{x}_n)$
image minimum	$\bigwedge \mathbf{a} = \bigwedge_{\mathbf{x} \in \mathbf{X}} \mathbf{a}(\mathbf{x}) = \mathbf{a}(\mathbf{x}_1) \wedge \mathbf{a}(\mathbf{x}_2) \wedge \cdots \wedge \mathbf{a}(\mathbf{x}_n)$
image complement	$\tilde{\mathbf{a}} = \left\{(\mathbf{x}, \mathbf{c}(\mathbf{x})) : \mathbf{c}(\mathbf{x}) = \widetilde{\mathbf{a}(\mathbf{x})},\ \mathbf{x} \in \mathbf{X}\right\}$
pseudo inverse	$\mathbf{a}^{-1} = \left\{(\mathbf{x}, \mathbf{b}(\mathbf{x})) : \mathbf{b}(\mathbf{x}) = \begin{cases} \frac{1}{\mathbf{a}(\mathbf{x})} & \text{if } \mathbf{a}(\mathbf{x}) \neq 0 \\ 0 & \text{otherwise} \end{cases}\right\}$
image transpose	$\mathbf{a}' = \{((x, y), \mathbf{a}'(x, y)) : \mathbf{a}'(x, y) = \mathbf{a}(y, x),\ (y, x) \in \mathbf{X}\}$

1.5. Templates

Templates are images whose *values* are images. The notion of a template, as used in image algebra, unifies and generalizes the usual concepts of templates, masks, windows, and neighborhood functions into one general mathematical entity. In addition, templates generalize the notion of structuring elements as used in mathematical morphology [26, 55].

Definition. A *template* is an image whose pixel values are images (functions). In particular, an $\mathbb{F}$-*valued template from* $\mathbf{Y}$ *to* $\mathbf{X}$ is a function $\mathbf{t} : \mathbf{Y} \to \mathbb{F}^{\mathbf{X}}$. Thus, $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$ and $\mathbf{t}$ is an $\mathbb{F}^{\mathbf{X}}$-valued image on $\mathbf{Y}$.

For notational convenience we define $\mathbf{t_y} \equiv \mathbf{t}(\mathbf{y})\ \forall \mathbf{y} \in \mathbf{Y}$. The image $\mathbf{t_y}$ has representation

$$\mathbf{t_y} = \{(\mathbf{x}, \mathbf{t_y}(\mathbf{x})) \,:\, \mathbf{x} \in \mathbf{X}\}\,.$$

The pixel values $\mathbf{t_y}(\mathbf{x})$ of this image are called the *weights* of the template at point $\mathbf{y}$.

If $\mathbf{t}$ is a real- or complex-valued template from $\mathbf{Y}$ to $\mathbf{X}$, then the *support* of $\mathbf{t_y}$ is denoted by $S(\mathbf{t_y})$ and is defined as

$$S(\mathbf{t_y}) = \{\mathbf{x} \in \mathbf{X} \,:\, \mathbf{t_y}(\mathbf{x}) \neq 0\}\,.$$

More generally, if $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$ and $\mathbb{F}$ is an algebraic structure with a zero element 0, then the support of $\mathbf{t_y}$ will be defined as $S(\mathbf{t_y}) = \{\mathbf{x} \in \mathbf{X} \,:\, \mathbf{t_y}(\mathbf{x}) \neq 0\}$.

For extended real-valued templates we also define the following supports *at infinity*:

$$S_\infty(\mathbf{t_y}) = \{\mathbf{x} \in \mathbf{X} \,:\, \mathbf{t_y}(\mathbf{x}) \neq \infty\}$$

and

$$S_{-\infty}(\mathbf{t_y}) = \{\mathbf{x} \in \mathbf{X} \,:\, \mathbf{t_y}(\mathbf{x}) \neq -\infty\}\,.$$

If $\mathbf{X}$ is a space with an operation $+$ such that $(\mathbf{X}, +)$ is a group, then a template $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{X}}$ is said to be *translation invariant* (with respect to the operation $+$) if and only if for each triple $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbf{X}$ we have that $\mathbf{t_y}(\mathbf{x}) = \mathbf{t_{y+z}}(\mathbf{x} + \mathbf{z})$. Templates that are not translation invariant are called *translation variant* or, simply, *variant* templates. A large class of translation invariant templates with finite support have the nice property that they can be defined pictorially. For example, let $\mathbf{X} = \mathbb{Z}^2$ and $\mathbf{y} = (x,y)$ be an arbitrary point of $\mathbf{X}$. Set $\mathbf{x}_1 = (x, y-1)$, $\mathbf{x}_2 = (x+1, y)$, and $\mathbf{x}_3 = (x+1, y-1)$. Define $\mathbf{t} \in \left(\mathbb{R}^{\mathbf{X}}\right)^{\mathbf{X}}$ by defining the weights $\mathbf{t_y}(\mathbf{y}) = 1$, $\mathbf{t_y}(\mathbf{x_1}) = 3$, $\mathbf{t_y}(\mathbf{x}_2) = 2$, $\mathbf{t_y}(\mathbf{x}_3) = 4$, and $\mathbf{t_y}(\mathbf{x}) = 0$ whenever $\mathbf{x}$ is not an element of $\{\mathbf{y}, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$. Note that it follows from the definition of $\mathbf{t}$ that $S(\mathbf{t_y}) = \{\mathbf{y}, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$. Thus, at any arbitrary point $\mathbf{y}$, the configuration of the support and weights of $\mathbf{t_y}$ is as shown in Figure 1.5.1. The shaded cell in the pictorial representation of $\mathbf{t_y}$ indicates the location of the point $\mathbf{y}$.

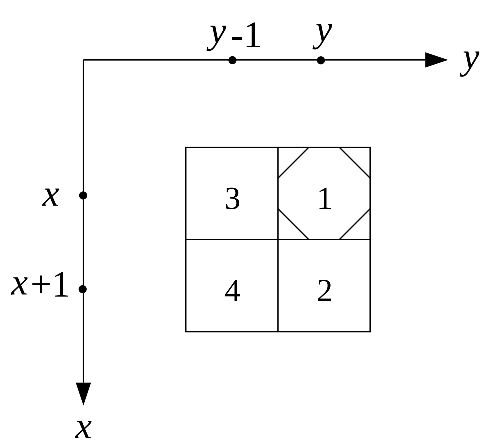

Figure 1.5.1. Pictorial representation of a translation invariant template.

There are certain collections of templates that can be defined explicitly in terms of parameters. These parameterized templates are of great practical importance.

Definition. A *parameterized* $\mathbb{F}$*-valued template from* $\mathbf{Y}$ *to* $\mathbf{X}$ *with parameters in* P is a function of form $\mathbf{t} : P \to \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$. The set P is called the *set of parameters* and each $p \in P$ is called a *parameter* of $\mathbf{t}$.

Thus, a parameterized $\mathbb{F}$-valued template from $\mathbf{Y}$ to $\mathbf{X}$ gives rise to a family of regular $\mathbb{F}$-valued templates from $\mathbf{Y}$ to $\mathbf{X}$, namely $\left\{\mathbf{t}(p) \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}} : \; p \in P\right\}$.

Image-Template Products

The definition of an image-template product provides the rules for combining images with templates and templates with templates. The definition of this product includes the usual correlation and convolution products used in digital image processing. Suppose $\mathbb{F}$ is a value set with two binary operations $\bigcirc$ and γ, where $\bigcirc$ distributes over γ, and γ is associative and commutative. If $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{Z}}\right)^{\mathbf{Y}}$, then for each $\mathbf{y} \in \mathbf{Y}$, $\mathbf{t_y} \in \mathbb{F}^{\mathbf{X}}$. Thus, if $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$, where $\mathbf{X}$ is finite, then $\mathbf{a} \bigcirc \mathbf{t_y} \in \mathbb{F}^{\mathbf{X}}$ and $\Gamma\left(\mathbf{a} \bigcirc \mathbf{t_y}\right) \in \mathbb{F}$. It follows that the binary operations $\bigcirc$ and γ induce a binary operation

$$\textcircled{\gamma} : \mathbb{F}^{\mathbf{X}} \times \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}} \to \mathbb{F}^{\mathbf{Y}},$$

where

$$\mathbf{b} = \mathbf{a} \,\textcircled{\gamma}\, \mathbf{t} \in \mathbb{F}^{\mathbf{Y}}$$

is defined by

$$\mathbf{b}(\mathbf{y}) = \Gamma(\mathbf{a} \bigcirc \mathbf{t_y}) = \mathop{\Gamma}_{\mathbf{x} \in \mathbf{X}} \left(\mathbf{a}(\mathbf{x}) \bigcirc \mathbf{t_y}(\mathbf{x})\right).$$

Therefore, if $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, then

$$\mathbf{b}(\mathbf{y}) = (\mathbf{a}(\mathbf{x}_1) \bigcirc \mathbf{t_y}(\mathbf{x}_1))\gamma(\mathbf{a}(\mathbf{x}_2) \bigcirc \mathbf{t_y}(\mathbf{x}_2))\gamma \cdots \gamma(\mathbf{a}(\mathbf{x}_n) \bigcirc \mathbf{t_y}(\mathbf{x}_n)).$$

The expression $\mathbf{a} \,\textcircled{\gamma}\, \mathbf{t}$ is called the *right product of* $\mathbf{a}$ *with* $\mathbf{t}$. Note that while $\mathbf{a}$ is an image on $\mathbf{X}$, the product $\mathbf{a} \,\textcircled{\gamma}\, \mathbf{t}$ is an image on $\mathbf{Y}$. Thus, templates allow for the transformation of an image from one type of domain to an entirely different domain type.

Replacing $(\mathbb{F}, \gamma, \bigcirc)$ by $(\mathbb{R}, +, \cdot)$ changes $\mathbf{b} = \mathbf{a} \,\textcircled{\gamma}\, \mathbf{t}$ into

$$\mathbf{b} = \mathbf{a} \oplus \mathbf{t},$$

the *linear image-template product*, where

$$\mathbf{b}(\mathbf{y}) = \sum_{\mathbf{x} \in \mathbf{X}} (\mathbf{a}(\mathbf{x}) \cdot \mathbf{t_y}(\mathbf{x})),$$

$\mathbf{a} \in \mathbb{R}^{\mathbf{X}}$, and $\mathbf{t} \in \left(\mathbb{R}^{\mathbf{X}}\right)^{\mathbf{Y}}$.

Every template $\mathbf{s} \in \left(\mathbb{F}^{\mathbf{Y}}\right)^{\mathbf{X}}$ has a transpose $\mathbf{s}' \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$ which is defined $\mathbf{s}'_{\mathbf{y}}(\mathbf{x}) = \mathbf{s_x}(\mathbf{y})$. Obviously, $(\mathbf{s}')' = \mathbf{s}$ and $\mathbf{s}'$ reverses the mapping order from $\mathbf{X} \to \mathbb{F}^{\mathbf{Y}}$

to $\mathbf{Y} \to \mathbb{F}^{\mathbf{X}}$. By definition, $\mathbf{s}'_{\mathbf{y}} \bigcirc \mathbf{a} \in \mathbb{F}^{\mathbf{X}}$ and $\Gamma\left(\mathbf{s}'_{\mathbf{y}} \bigcirc \mathbf{a}\right) \in \mathbb{F}$, whenever $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$ and $\mathbf{s} \in \left(\mathbb{F}^{\mathbf{Y}}\right)^{\mathbf{X}}$. Hence the binary operations $\bigcirc$ and γ induce another product operation

$$\circledgamma : \left(\mathbb{F}^{\mathbf{Y}}\right)^{\mathbf{X}} \times \mathbb{F}^{\mathbf{X}} \to \mathbb{F}^{\mathbf{Y}},$$

where

$$\mathbf{b} = \mathbf{s} \circledgamma \mathbf{a} \in \mathbb{F}^{\mathbf{Y}}$$

is defined by

$$\mathbf{b}(\mathbf{y}) = \Gamma\left(\mathbf{s}'_{\mathbf{y}} \bigcirc \mathbf{a}\right) = \Gamma_{\mathbf{x}\in\mathbf{X}}\left(\mathbf{s}'_{\mathbf{y}}(\mathbf{x}) \bigcirc \mathbf{a}(\mathbf{x})\right).$$

The expression $\mathbf{s} \circledgamma \mathbf{a}$ is called the *left product of* $\mathbf{a}$ *with* $\mathbf{s}$.

When computing $\mathbf{s} \circledgamma \mathbf{a}$, it is not necessary to use the transpose $\mathbf{s}'$ since

$$\Gamma_{\mathbf{x}\in\mathbf{X}}\left(\mathbf{s}'_{\mathbf{y}}(\mathbf{x}) \bigcirc \mathbf{a}(\mathbf{x})\right) = \Gamma_{\mathbf{x}\in\mathbf{X}}\left(\mathbf{s}_{\mathbf{x}}(\mathbf{y}) \bigcirc \mathbf{a}(\mathbf{x})\right).$$

This allows us to redefine the transformation $\mathbf{b} = \mathbf{s} \circledgamma \mathbf{a}$ as

$$\mathbf{b}(\mathbf{y}) = \Gamma_{\mathbf{x}\in\mathbf{X}}\left(\mathbf{s}_{\mathbf{x}}(\mathbf{y}) \bigcirc \mathbf{a}(\mathbf{x})\right).$$

For the remainder of this section we assume that $(\mathbb{F}, \gamma)$ is a monoid and let 0 denote the zero of $\mathbb{F}$ under the operation γ. Suppose $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$ and $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{Z}}\right)^{\mathbf{Y}}$, where $\mathbf{X}$ and $\mathbf{Z}$ are subsets of the same space. Since $\mathbb{F}$ is a monoid, the operator $\circledgamma$ can be extended to a mapping

$$\circledgamma : \mathbb{F}^{\mathbf{X}} \times \left(\mathbb{F}^{\mathbf{Z}}\right)^{\mathbf{Y}} \to \mathbb{F}^{\mathbf{Y}},$$

where $\mathbf{b} = \mathbf{a} \circledgamma \mathbf{t}$ is defined by is defined by

$$\mathbf{b}(\mathbf{y}) = \begin{cases} \Gamma_{\mathbf{x}\in\mathbf{X}\cap\mathbf{Z}}\left(\mathbf{a}(\mathbf{x}) \bigcirc \mathbf{t}_{\mathbf{y}}(\mathbf{x})\right) & \text{if } \mathbf{X} \cap \mathbf{Z} \neq \varnothing \\ 0 & \text{if } \mathbf{X} \cap \mathbf{Z} = \varnothing. \end{cases}$$

The left product $\mathbf{s} \circledgamma \mathbf{a}$ is defined in a similar fashion. Subsequent examples will demonstrate that the ability of replacing **X** with **Z** greatly simplifies the issue of template implementation and the use of templates in algorithm development.

Significant reduction in the number of computations involved the image-template product can be achieved if $(\mathbb{F}, \gamma, \bigcirc)$ is a commutative semiring. Recall that if $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{Z}}\right)^{\mathbf{Y}}$, then the support of $\mathbf{t}$ at a point $\mathbf{y} \in \mathbf{Y}$ with respect to the operation γ is defined as $S(\mathbf{t}_{\mathbf{y}}) = \{\mathbf{x} \in \mathbf{Z} : \mathbf{t}_{\mathbf{y}}(\mathbf{x}) \neq 0\}$. Since $\mathbf{t}_{\mathbf{y}}(\mathbf{x}) = 0$ whenever $\mathbf{x} \notin S(\mathbf{t}_{\mathbf{y}})$, we have that $\mathbf{a}(\mathbf{x}) \bigcirc \mathbf{t}_{\mathbf{y}}(\mathbf{x}) = 0$ whenever $\mathbf{x} \notin S(\mathbf{t}_{\mathbf{y}})$ and, therefore,

$$\Gamma_{\mathbf{x}\in\mathbf{X}\cap\mathbf{Z}}\left(\mathbf{a}(\mathbf{x}) \bigcirc \mathbf{t}_{\mathbf{y}}(\mathbf{x})\right) = \Gamma_{\mathbf{x}\in\mathbf{X}\cap S(\mathbf{t}_{\mathbf{y}})}\left(\mathbf{a}(\mathbf{x}) \bigcirc \mathbf{t}_{\mathbf{y}}(\mathbf{x})\right).$$

It follows that the computation of the new pixel value **b(y)** does not depend on the size of **X**, but on the size of $S(\mathbf{t}_{\mathbf{y}})$. Therefore, if $k = card(\mathbf{X} \cap S(\mathbf{t}_{\mathbf{y}}))$, then the computation of **b(y)** requires a total of $2k^2 - 1$ operations of type γ and $\bigcirc$.

As pointed out earlier, substitution of different value sets and specific binary operations for γ and $\bigcirc$ results in a wide variety of different image transforms. Our prime examples are the ring $(\mathbb{R}, +, \cdot)$ and the value sets $(\mathbb{R}_{\pm\infty}, \vee, \wedge, +, +')$ and $\left(\mathbb{R}_{\infty}^{\geq 0}, \vee, \wedge, \times, \times'\right)$. The structure $(\mathbb{R}_{\pm\infty}, \vee, \wedge, +, +')$ provides for two lattice products:

$$\mathbf{b} = \mathbf{a} \boxed{\vee}\, \mathbf{t},$$

where

$$\mathbf{b}(\mathbf{y}) = \bigvee_{\mathbf{x} \in \mathbf{X} \cap S_{-\infty}(\mathbf{t}_{\mathbf{y}})} [\mathbf{a}(\mathbf{x}) + \mathbf{t}_{\mathbf{y}}(\mathbf{x})]\,,$$

and

$$\mathbf{b} = \mathbf{a} \boxed{\wedge}\, \mathbf{t}\,,$$

where

$$\mathbf{b}(\mathbf{y}) = \bigwedge_{\mathbf{x} \in \mathbf{X} \cap S_{\infty}(\mathbf{t}_{\mathbf{y}})} [\mathbf{a}(\mathbf{x}) +' \mathbf{t}_{\mathbf{y}}(\mathbf{x})]\,.$$

In order to distinguish between these two types of lattice transforms, we call the operator $\boxed{\vee}$ the *additive maximum* and $\boxed{\wedge}$ the *additive minimum*. It follows from our earlier discussion that if $\mathbf{X} \cap S_{-\infty}(\mathbf{t}_{\mathbf{y}}) = \varnothing$, then the value of $\mathbf{b}(\mathbf{y})$ is $-\infty$, the *zero* of $\mathbb{R}_{\pm\infty}$ under the operation of $\vee$. Similarly, if $\mathbf{X} \cap S_{\infty}(\mathbf{t}_{\mathbf{y}}) = \varnothing$, then $\mathbf{b}(\mathbf{y}) = \infty$.

The left additive max and min operations are defined by

$$\mathbf{t} \boxed{\vee}\, \mathbf{a} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) \,:\, \mathbf{b}(\mathbf{y}) = \bigvee_{\mathbf{x} \in \mathbf{X} \cap S_{-\infty}(\mathbf{t}_{\mathbf{y}})} [\mathbf{t}_{\mathbf{x}}(\mathbf{y}) + \mathbf{a}(\mathbf{x})]\,,\ \mathbf{y} \in \mathbf{Y} \right\}$$

and

$$\mathbf{t} \boxed{\wedge}\, \mathbf{a} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) \,:\, \mathbf{b}(\mathbf{y}) = \bigwedge_{\mathbf{x} \in \mathbf{X} \cap S_{\infty}(\mathbf{t}_{\mathbf{y}})} [\mathbf{t}_{\mathbf{x}}(\mathbf{y}) +' \mathbf{a}(\mathbf{x})]\,,\ \mathbf{y} \in \mathbf{Y} \right\},$$

respectively. The relationship between the additive max and min is given in terms of lattice duality by

$$\mathbf{a} \boxed{\wedge}\, \mathbf{t} = (\mathbf{t}^* \boxed{\vee}\, \mathbf{a}^*)^*\,,$$

where the image $\mathbf{a}^*$ is defined by $\mathbf{a}^*(\mathbf{x}) = [\mathbf{a}(\mathbf{x})]^*$, and the conjugate (or dual) of $\mathbf{t} \in \left(\mathbb{R}^{\mathbf{X}}_{\pm\infty}\right)^{\mathbf{Y}}$ is the template $\mathbf{t}^* \in \left(\mathbb{R}^{\mathbf{Y}}_{\pm\infty}\right)^{\mathbf{X}}$ defined by $\mathbf{t}^*_{\mathbf{x}}(\mathbf{y}) = [\mathbf{t}_{\mathbf{y}}(\mathbf{x})]^*$. It follows that $\mathbf{t}^*_{\mathbf{x}}(\mathbf{y}) = -\mathbf{t}'_{\mathbf{y}}(\mathbf{x})$.

The value set $\left(\mathbb{R}^{\geq 0}_{\infty}, \vee, \wedge, \times, \times'\right)$ also provides for two lattice products. Specifically, we have

$$\mathbf{b} = \mathbf{a} \circledvee \mathbf{t}\,,$$

where

$$\mathbf{b}(\mathbf{y}) = \bigvee_{\mathbf{x} \in \mathbf{X} \cap S(\mathbf{t}_{\mathbf{y}})} [\mathbf{a}(\mathbf{x}) \times \mathbf{t}_{\mathbf{y}}(\mathbf{x})]\,,$$

and

$$\mathbf{b} = \mathbf{a} \circledwedge \mathbf{t}\,,$$

where

$$\mathbf{b}(\mathbf{y}) = \bigwedge_{\mathbf{x} \in \mathbf{X} \cap S_{\infty}(\mathbf{t}_{\mathbf{y}})} [\mathbf{a}(\mathbf{x}) \times' \mathbf{t}_{\mathbf{y}}(\mathbf{x})]\,.$$

Here 0 is the zero of $\mathbb{R}^{\geq 0}_{\infty}$ under the operation of $\vee$, so that $\mathbf{b}(\mathbf{y}) = 0$ whenever $\mathbf{X} \cap S(\mathbf{t}_{\mathbf{y}}) = \varnothing$. Similarly, $\mathbf{b}(\mathbf{y}) = \infty$ whenever $\mathbf{X} \cap S_{\infty}(\mathbf{t}_{\mathbf{y}}) = \varnothing$.

The lattice products $\textcircled{\vee}$ and $\textcircled{\wedge}$ are called the *multiplicative maximum* and *multiplicative minimum*, respectively. The *left multiplicative max* and *left multiplicative min* are defined as

$$\mathbf{t} \textcircled{\vee} \mathbf{a} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) \,:\, \mathbf{b}(\mathbf{y}) = \bigvee_{\mathbf{x} \in \mathbf{X} \cap S_\infty(\mathbf{t}'_\mathbf{y})} [\mathbf{t}_\mathbf{x}(\mathbf{y}) \times \mathbf{a}(\mathbf{x})]\,,\ \mathbf{y} \in \mathbf{Y} \right\}$$

and

$$\mathbf{t} \textcircled{\wedge} \mathbf{a} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) \,:\, \mathbf{b}(\mathbf{y}) = \bigwedge_{\mathbf{x} \in \mathbf{X} \cap S_\infty(\mathbf{t}'_\mathbf{y})} [\mathbf{t}_\mathbf{x}(\mathbf{y}) \times' \mathbf{a}(\mathbf{x})]\,,\ \mathbf{y} \in \mathbf{Y} \right\},$$

respectively. The duality relation between the multiplicative max and min is given by

$$\mathbf{a} \textcircled{\wedge} \mathbf{t} = (\mathbf{t}^* \textcircled{\vee} \mathbf{a}^*)^*,$$

where $\mathbf{a}^*(\mathbf{x}) = (\mathbf{a}(\mathbf{x}))^*$ and $\mathbf{t}^*_\mathbf{x}(\mathbf{y}) = [\mathbf{t}_\mathbf{y}(\mathbf{x})]^*$. Here r^* denotes the conjugate of r in $\mathbb{R}^{\gtrless 0}_{\infty}$.

Summary of Image-Template Products

In the following list of pertinent image-template products $\mathbf{a} \in \mathbb{F}^\mathbf{X}$ and $\mathbf{t} \in (\mathbb{F}^\mathbf{X})^\mathbf{Y}$. Again, for each operation we assume the appropriate value set $\mathbb{F}$.

right generic product

$$\mathbf{a} \textcircled{\gamma} \mathbf{t} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) \,:\, \mathbf{b}(\mathbf{y}) = \Gamma_{\mathbf{x} \in \mathbf{X}} (\mathbf{a}(\mathbf{x}) \bigcirc \mathbf{t}_\mathbf{y}(\mathbf{x}))\,,\ \mathbf{y} \in \mathbf{Y} \right\}$$

right linear product

$$\mathbf{a} \oplus \mathbf{t} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) \,:\, \mathbf{b}(\mathbf{y}) = \sum_{\mathbf{x} \in \mathbf{X}} (\mathbf{a}(\mathbf{x}) \cdot \mathbf{t}_\mathbf{y}(\mathbf{x}))\,,\ \mathbf{y} \in \mathbf{Y} \right\}$$

right additive max

$$\mathbf{a} \boxed{\vee} \mathbf{t} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) \,:\, \mathbf{b}(\mathbf{y}) = \bigvee_{\mathbf{x} \in \mathbf{X}} [\mathbf{a}(\mathbf{x}) + \mathbf{t}_\mathbf{y}(\mathbf{x})]\,,\ \mathbf{y} \in \mathbf{Y} \right\}$$

right additive min

$$\mathbf{a} \boxed{\wedge} \mathbf{t} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) \,:\, \mathbf{b}(\mathbf{y}) = \bigwedge_{\mathbf{x} \in \mathbf{X}} [\mathbf{a}(\mathbf{x}) +' \mathbf{t}_\mathbf{y}(\mathbf{x})]\,,\ \mathbf{y} \in \mathbf{Y} \right\}$$

right multiplicative max

$$\mathbf{a} \textcircled{\vee} \mathbf{t} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) \,:\, \mathbf{b}(\mathbf{y}) = \bigvee_{\mathbf{x} \in \mathbf{X}} [\mathbf{a}(\mathbf{x}) \times \mathbf{t}_\mathbf{y}(\mathbf{x})]\,,\ \mathbf{y} \in \mathbf{Y} \right\}$$

right multiplicative min

$$\mathbf{a} \textcircled{\wedge} \mathbf{t} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) \,:\, \mathbf{b}(\mathbf{y}) = \bigwedge_{\mathbf{x} \in \mathbf{X}} [\mathbf{a}(\mathbf{x}) \times' \mathbf{t}_\mathbf{y}(\mathbf{x})]\,,\ \mathbf{y} \in \mathbf{Y} \right\}$$

right xor max

$$\mathbf{a} \tilde{\boxed{\vee}} \mathbf{t} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) \,:\, \mathbf{b}(\mathbf{y}) = \bigvee_{\mathbf{x} \in \mathbf{X}} \left[\mathbf{a}(\mathbf{x}) \tilde{+} \mathbf{t}_\mathbf{y}(\mathbf{x})\right]\,,\ \mathbf{y} \in \mathbf{Y} \right\}$$

right xor min

$$\mathbf{a} \tilde{\boxed{\wedge}} \mathbf{t} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) \,:\, \mathbf{b}(\mathbf{y}) = \bigwedge_{\mathbf{x} \in \mathbf{X}} \left[\mathbf{a}(\mathbf{x}) \tilde{+}' \mathbf{t}_\mathbf{y}(\mathbf{x})\right]\,,\ \mathbf{y} \in \mathbf{Y} \right\}$$

In the next set of operations, $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{Y}}\right)^{\mathbf{X}}$.

left generic product

$$\mathbf{t} \circledgamma \mathbf{a} = \left\{(\mathbf{y}, \mathbf{b}(\mathbf{y})) \;:\; \mathbf{b}(\mathbf{y}) = \Gamma_{\mathbf{x}\in\mathbf{X}} \left(\mathbf{t}_{\mathbf{x}}(\mathbf{y}) \bigcirc \mathbf{a}(\mathbf{x})\right), \; \mathbf{y} \in \mathbf{Y}\right\}$$

left linear product

$$\mathbf{t} \oplus \mathbf{a} = \left\{(\mathbf{y}, \mathbf{b}(\mathbf{y})) \;:\; \mathbf{b}(\mathbf{y}) = \sum_{\mathbf{x}\in\mathbf{X}} (\mathbf{t}_{\mathbf{x}}(\mathbf{y}) \cdot \mathbf{a}(\mathbf{x})), \; \mathbf{y} \in \mathbf{Y}\right\}$$

left additive max

$$\mathbf{t} \boxed{\vee} \mathbf{a} = \left\{(\mathbf{y}, \mathbf{b}(\mathbf{y})) \;:\; \mathbf{b}(\mathbf{y}) = \bigvee_{\mathbf{x}\in\mathbf{X}} \left[\mathbf{t}_{\mathbf{x}}(\mathbf{y}) + \mathbf{a}(\mathbf{x})\right], \; \mathbf{y} \in \mathbf{Y}\right\}$$

left additive min

$$\mathbf{t} \boxed{\wedge} \mathbf{a} = \left\{(\mathbf{y}, \mathbf{b}(\mathbf{y})) \;:\; \mathbf{b}(\mathbf{y}) = \bigwedge_{\mathbf{x}\in\mathbf{X}} \left[\mathbf{t}_{\mathbf{x}}(\mathbf{y}) +' \mathbf{a}(\mathbf{x})\right], \; \mathbf{y} \in \mathbf{Y}\right\}$$

left multiplicative max

$$\mathbf{t} \circledvee \mathbf{a} = \left\{(\mathbf{y}, \mathbf{b}(\mathbf{y})) \;:\; \mathbf{b}(\mathbf{y}) = \bigvee_{\mathbf{x}\in\mathbf{X}} \left[\mathbf{t}_{\mathbf{x}}(\mathbf{y}) \times \mathbf{a}(\mathbf{x})\right], \; \mathbf{y} \in \mathbf{Y}\right\}$$

left multiplicative min

$$\mathbf{t} \circledwedge \mathbf{a} = \left\{(\mathbf{y}, \mathbf{b}(\mathbf{y})) \;:\; \mathbf{b}(\mathbf{y}) = \bigwedge_{\mathbf{x}\in\mathbf{X}} \left[\mathbf{t}_{\mathbf{x}}(\mathbf{y}) \times' \mathbf{a}(\mathbf{x})\right], \; \mathbf{y} \in \mathbf{Y}\right\}$$

Binary and Unary Template Operations

Since templates are images, all unary and binary image operations discussed earlier apply to templates as well. Any binary γ on $\mathbb{F}$ induces a binary operation (again denoted by γ) on $\left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$ as follows: for each pair $\mathbf{s}, \mathbf{t} \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$ the induced operation $\mathbf{s}\gamma\mathbf{t}$ is defined in terms of the induced binary image operation on $\mathbb{F}^{\mathbf{X}}$, namely $(\mathbf{s}\gamma\mathbf{t})_{\mathbf{y}} \equiv \mathbf{s}_{\mathbf{y}}\gamma\mathbf{t}_{\mathbf{y}} \quad \forall \mathbf{y} \in \mathbf{Y}$. Thus, if $\mathbb{F} = \mathbb{R}$, $\mathbf{s}, \mathbf{t} \in \left(\mathbb{R}^{\mathbf{X}}\right)^{\mathbf{Y}}$, and $\gamma = +$, then $(\mathbf{s} + \mathbf{t})_{\mathbf{y}} = \mathbf{s}_{\mathbf{y}} + \mathbf{t}_{\mathbf{y}}$, where $\mathbf{s}_{\mathbf{y}} + \mathbf{t}_{\mathbf{y}}$ denotes the pointwise sum of the two images $\mathbf{s}_{\mathbf{y}} \in \mathbb{R}^{\mathbf{X}}$ and $\mathbf{t}_{\mathbf{y}} \in \mathbb{R}^{\mathbf{X}}$.

The unary template operations of prime importance are the global reduce operations. Suppose $\mathbf{Y}$ is a finite point set, say $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}$, and $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$. Any binary semigroup operation γ on $\mathbb{F}$ induces a global reduce operation

$$\Gamma : \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}} \to \mathbb{F}^{\mathbf{X}}$$

which is defined by

$$\Gamma\mathbf{t} = \Gamma_{\mathbf{y}\in\mathbf{Y}} \mathbf{t}_{\mathbf{y}} = \Gamma_{k=1}^{n} \mathbf{t}_{\mathbf{y}_k} = \mathbf{t}_{\mathbf{y}_1}\gamma\mathbf{t}_{\mathbf{y}_2}\gamma \cdots \gamma\mathbf{t}_{\mathbf{y}_n}.$$

Thus, for example, if $\mathbb{F} = \mathbb{R}$ and γ is the operation of addition ($\gamma = +$), then $\Gamma = \Sigma$ and

$$\sum \mathbf{t} = \sum_{\mathbf{y} \in \mathbf{Y}} \mathbf{t_y} = \mathbf{t}_{\mathbf{y}_1} + \mathbf{t}_{\mathbf{y}_2} + \cdots + \mathbf{t}_{\mathbf{y}_n}.$$

Therefore, $\sum \mathbf{t}$ is an image, namely the sum of a finite number of images.

In all, the value set $(\mathbb{R}, +, \cdot, \vee, \wedge)$ provides for four basic global reduce operations, namely $\sum \mathbf{t}$, $\prod \mathbf{t}$, $\bigvee \mathbf{t}$, and $\bigwedge \mathbf{t}$.

If the value set $\mathbb{F}$ has two binary operations γ and $\bigcirc$ so that $(\mathbb{F}, \gamma, \bigcirc)$ is a ring (or semiring), then under the induced operations $\left(\left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}, \gamma, \bigcirc\right)$ is also a ring (or semiring). Analogous to the image-template product, the binary operations $\bigcirc$ and γ induce a template convolution product

$$\text{\textcircled{γ}} : \left(\mathbb{F}^{\mathbf{Z}}\right)^{\mathbf{X}} \times \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}} \to \left(\mathbb{F}^{\mathbf{Z}}\right)^{\mathbf{Y}}$$

defined as follows. Suppose $\mathbf{s} \in \left(\mathbb{F}^{\mathbf{Z}}\right)^{\mathbf{X}}$, $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$, and $\mathbf{X}$ a finite point set. Then the template product $\mathbf{r} = \mathbf{s}\,\text{\textcircled{γ}}\,\mathbf{t}$, where $\mathbf{r} \in \left(\mathbb{F}^{\mathbf{Z}}\right)^{\mathbf{Y}}$, is defined as

$$\mathbf{r_y}(\mathbf{z}) = \Gamma_{\mathbf{x} \in \mathbf{X}} \left(\mathbf{s_x}(\mathbf{z}) \bigcirc \mathbf{t_y}(\mathbf{x})\right) \qquad \forall \mathbf{y} \in \mathbf{Y} \text{ and } \forall \mathbf{z} \in \mathbf{Z}\,.$$

Thus, if $\mathbf{s} \in \left(\mathbb{R}^{\mathbf{Z}}\right)^{\mathbf{X}}$ and $\mathbf{t} \in \left(\mathbb{R}^{\mathbf{X}}\right)^{\mathbf{Y}}$, then $\mathbf{r} = \mathbf{s} \oplus \mathbf{t}$ is given by the formula

$$\mathbf{r_y}(\mathbf{z}) = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{s_x}(\mathbf{z}) \cdot \mathbf{t_y}(\mathbf{x})\,.$$

The lattice product $\mathbf{r} = \mathbf{s} \boxed{\vee} \mathbf{t}$ is defined in a similar manner. For $\mathbf{s} \in \left(\mathbb{R}_{\pm\infty}^{\mathbf{Z}}\right)^{\mathbf{X}}$ and $\mathbf{t} \in \left(\mathbb{R}_{\pm\infty}^{\mathbf{X}}\right)^{\mathbf{Y}}$, the product template $\mathbf{r}$ is given by

$$\mathbf{r_y}(\mathbf{z}) = \bigvee_{\mathbf{x} \in \mathbf{X}} \left[\mathbf{s_x}(\mathbf{z}) + \mathbf{t_y}(\mathbf{x})\right].$$

The following example provides a specific instance of the above product formulation.

Example: Suppose $\mathbf{s}, \mathbf{t} \in \left(\mathbb{R}^{\mathbb{Z}^2}\right)^{\mathbb{Z}^2}$ are the following translation invariant templates:

$$\mathbf{s_y} = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array} \qquad \mathbf{t_y} = \begin{array}{|c|}\hline 1 \\ \hline 3 \\ \hline -1 \\ \hline \end{array}$$

Then the template product $\mathbf{r} = \mathbf{s} \oplus \mathbf{t}$ is the template defined by

$\mathbf{r_y} =$

1	2	1
3	6	3
-1	-2	-1

If $\mathbf{s}, \mathbf{t} \in \left(\mathbb{R}_{\pm\infty}^{\mathbb{Z}^2}\right)^{\mathbb{Z}^2}$ are defined as above with values $-\infty$ outside the support, then the template product $\mathbf{r} = \mathbf{s} \boxed{\vee} \mathbf{t}$ is the template defined by

$\mathbf{r_y} =$

2	3	2
4	5	4
0	1	0

The template $\mathbf{t}$ is not an $\mathbb{R}_{\infty}^{\geq 0}$-valued template. To provide an example of the template product $\mathbf{s} \circledvee \mathbf{t}$, we redefine $\mathbf{t}$ as

$\mathbf{t_y} =$

1
3
1

Then $\mathbf{r} = \mathbf{s} \circledvee \mathbf{t}$ is given by

$\mathbf{r_y} =$

1	2	1
3	6	3
1	2	1

The utility of template products stems from the fact that in semirings the equation

$$\mathbf{a} \textcircled{γ} (\mathbf{s} \textcircled{γ} \mathbf{t}) = (\mathbf{a} \textcircled{γ} \mathbf{s}) \textcircled{γ} \mathbf{t}$$

holds [1]. This equation can be utilized in order to reduce the computational burden associated with typical convolution problems. For example, if $\mathbf{r} \in \left(\mathbb{R}^{\mathbb{Z}^2}\right)^{\mathbb{Z}^2}$ is defined by

$\forall \mathbf{y} \in \mathbb{Z}^2$, then

$$\mathbf{r_y} = \begin{array}{|c|c|c|} \hline 4 & 6 & -4 \\ \hline 6 & 9 & -6 \\ \hline -4 & -6 & 4 \\ \hline \end{array}$$

$$\mathbf{a} \oplus \mathbf{r} = \mathbf{a} \oplus (\mathbf{s} \oplus \mathbf{t}) = (\mathbf{a} \oplus \mathbf{s}) \oplus \mathbf{t}\,,$$

where

$$\mathbf{s_y} = \begin{array}{|c|c|c|} \hline 2 & 3 & -2 \\ \hline \end{array} \qquad\qquad \mathbf{t_y} = \begin{array}{|c|} \hline 2 \\ \hline 3 \\ \hline -2 \\ \hline \end{array}$$

The construction of the new image $\mathbf{b} := \mathbf{a} \oplus \mathbf{r}$ requires nine multiplications and eight additions per pixel (if we ignore boundary pixels). In contrast, the computation of the image $\mathbf{b} := (\mathbf{a} \oplus \mathbf{s}) \oplus \mathbf{t}$ requires only six multiplications and four additions per pixel. For large images (e.g., size 1024×1024) this amounts to significant savings in computation.

Summary of Unary and Binary Template Operations

In the following $\mathbf{s}, \mathbf{t} \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$ and $\mathbb{F}$ denotes the appropriate value set.

generic binary operation	$\mathbf{s}\gamma\mathbf{t}: \quad (\mathbf{s}\gamma\mathbf{t})_{\mathbf{y}} \equiv \mathbf{s_y}\gamma\mathbf{t_y}$
template sum	$\mathbf{s}+\mathbf{t}: \quad (\mathbf{s}+\mathbf{t})_{\mathbf{y}} \equiv \mathbf{s_y}+\mathbf{t_y}$
max of two templates	$\mathbf{s}\vee\mathbf{t}: \quad (\mathbf{s}\vee\mathbf{t})_{\mathbf{y}} \equiv \mathbf{s_y}\vee\mathbf{t_y}$
min of two templates	$\mathbf{s}\wedge\mathbf{t}: \quad (\mathbf{s}\wedge\mathbf{t})_{\mathbf{y}} \equiv \mathbf{s_y}\wedge\mathbf{t_y}$
generic reduce operation	$\Gamma\mathbf{t} \equiv \Gamma_{\mathbf{y}\in\mathbf{Y}} \mathbf{t_y} = \Gamma_{k=1}^{n} \mathbf{t}_{\mathbf{y}_k} = \mathbf{t}_{\mathbf{y}_1}\gamma\mathbf{t}_{\mathbf{y}_2}\gamma\cdots\gamma\mathbf{t}_{\mathbf{y}_n}$
sum reduce	$\sum\mathbf{t} \equiv \sum_{\mathbf{y}\in\mathbf{Y}} \mathbf{t_y} = \mathbf{t}_{\mathbf{y}_1}+\mathbf{t}_{\mathbf{y}_2}+\cdots+\mathbf{t}_{\mathbf{y}_n}$
product reduce	$\prod\mathbf{t} \equiv \prod_{\mathbf{y}\in\mathbf{Y}} \mathbf{t_y} = \mathbf{t}_{\mathbf{y}_1}\cdot\mathbf{t}_{\mathbf{y}_2}\cdot\cdots\cdot\mathbf{t}_{\mathbf{y}_n}$
max reduce	$\bigvee\mathbf{t} \equiv \bigvee_{\mathbf{y}\in\mathbf{Y}} \mathbf{t_y} = \mathbf{t}_{\mathbf{y}_1}\vee\mathbf{t}_{\mathbf{y}_2}\vee\cdots\vee\mathbf{t}_{\mathbf{y}_n}$
min reduce	$\bigwedge\mathbf{t} \equiv \bigwedge_{\mathbf{y}\in\mathbf{Y}} \mathbf{t_y} = \mathbf{t}_{\mathbf{y}_1}\wedge\mathbf{t}_{\mathbf{y}_2}\wedge\cdots\wedge\mathbf{t}_{\mathbf{y}_n}$

In the next list, $\mathbf{s} \in \left(\mathbb{F}^{\mathbf{Z}}\right)^{\mathbf{X}}$, $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$, $\mathbf{X}$ is a finite point set, and $\mathbb{F}$ denotes the appropriate value set.

generic template product	$\mathbf{r} = \mathbf{s} \circledgamma \mathbf{t}$:	$\mathbf{r_y}(\mathbf{z}) = \Gamma_{\mathbf{x}\in\mathbf{X}} (\mathbf{s_x}(\mathbf{z}) \bigcirc \mathbf{t_y}(\mathbf{x}))$
linear template product	$\mathbf{r} = \mathbf{s} \oplus \mathbf{t}$:	$\mathbf{r_y}(\mathbf{z}) = \sum_{\mathbf{x}\in\mathbf{X}} \mathbf{s_x}(\mathbf{z}) \cdot \mathbf{t_y}(\mathbf{x})$
additive max product	$\mathbf{r} = \mathbf{s} \boxed{\vee} \mathbf{t}$:	$\mathbf{r_y}(\mathbf{z}) = \bigvee_{\mathbf{x}\in\mathbf{X}} \mathbf{s_x}(\mathbf{z}) + \mathbf{t_y}(\mathbf{x})$
additive min product	$\mathbf{r} = \mathbf{s} \boxed{\wedge} \mathbf{t}$:	$\mathbf{r_y}(\mathbf{z}) = \bigwedge_{\mathbf{x}\in\mathbf{X}} \mathbf{s_x}(\mathbf{z}) + \mathbf{t_y}(\mathbf{x})$
multiplicative max product	$\mathbf{r} = \mathbf{s} \circledvee \mathbf{t}$:	$\mathbf{r_y}(\mathbf{z}) = \bigvee_{\mathbf{x}\in\mathbf{X}} \mathbf{s_x}(\mathbf{z}) \cdot \mathbf{t_y}(\mathbf{x})$
multiplicative min product	$\mathbf{r} = \mathbf{s} \circledwedge \mathbf{t}$:	$\mathbf{r_y}(\mathbf{z}) = \bigwedge_{\mathbf{x}\in\mathbf{X}} \mathbf{s_x}(\mathbf{z}) \cdot \mathbf{t_y}(\mathbf{x})$

1.6. Recursive Templates

In this section we introduce the notions of recursive templates and recursive template operations, which are direct extensions of the notions of templates and the corresponding template operations discussed in the preceding section.

A recursive template is defined in terms of a regular template from some point set $\mathbf{X}$ to another point set $\mathbf{Y}$ with some partial order imposed on $\mathbf{Y}$.

> **Definition.** A *partially ordered set* $(P, \prec)$ (or *poset*) is a set P together with a binary relation $\prec$, satisfying the following three axioms for arbitrary $x, y, z \in P$:
>
> (i) $x \prec x$ (reflexive)
>
> (ii) $x \prec y$ and $y \prec x \Rightarrow x = y$ (antisymmetric)
>
> (iii) $x \prec y$ and $y \prec z \Rightarrow x \prec z$ (transitive)

Now suppose that $\mathbf{X}$ is a point set, $\mathbf{Y}$ is a partially ordered point set with partial order $\prec$, and $\mathbb{F}$ a monoid. An $\mathbb{F}$-*valued recursive template* $\mathbf{t}$ *from* $\mathbf{Y}$ *to* $\mathbf{X}$ is a function $\mathbf{t} = (\mathbf{t}_{\not\prec}, \mathbf{t}_{\prec}) : \mathbf{Y} \to \left(\mathbb{F}^{\mathbf{X}}, \mathbb{F}^{\mathbf{Y}}\right)$, where $\mathbf{t}_{\not\prec} : \mathbf{Y} \to \mathbb{F}^{\mathbf{X}}$ and $\mathbf{t}_{\prec} : \mathbf{Y} \to \mathbb{F}^{\mathbf{Y}}$, such that

1. $\mathbf{y} \notin S(\mathbf{t}_{\prec}(\mathbf{y}))$ and
2. for each $\mathbf{z} \in S(t_{\prec}(\mathbf{y}))$, $\mathbf{z} \prec \mathbf{y}$.

Thus, for each $\mathbf{y} \in \mathbf{Y}$, $\mathbf{t}_{\not\prec}(\mathbf{y})$ is an $\mathbb{F}$-valued image on $\mathbf{X}$ and $\mathbf{t}_{\prec}(\mathbf{y})$ is an $\mathbb{F}$-valued image on $\mathbf{Y}$.

In most applications, the relation $\mathbf{X} \subset \mathbf{Y}$ or $\mathbf{X} = \mathbf{Y}$ usually holds. Also, for consistency of notation and for notational convenience, we define $\mathbf{t}_{\not\prec\mathbf{y}} \equiv \mathbf{t}_{\not\prec}(\mathbf{y})$ and $\mathbf{t}_{\prec\mathbf{y}} \equiv \mathbf{t}_{\prec}(\mathbf{y})$ so that $\mathbf{t_y} = (\mathbf{t}_{\not\prec\mathbf{y}}, \mathbf{t}_{\prec\mathbf{y}})$. The support of $\mathbf{t}$ at a point $\mathbf{y}$ is defined as $S(\mathbf{t_y}) = (S(\mathbf{t}_{\not\prec\mathbf{y}}), S(\mathbf{t}_{\prec\mathbf{y}}))$. The set of all $\mathbb{F}$-valued recursive templates from $\mathbf{Y}$ to $\mathbf{X}$ will be denoted by $\left(\mathbb{F}^{\mathbf{X}}, \mathbb{F}^{\mathbf{Y}}\right)^{(\mathbf{Y},\prec)}$.

In analogy to our previous definition of translation invariant templates, if $\mathbf{X}$ is closed under the operation $+$, then a recursive template $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{X}}, \mathbb{F}^{\mathbf{X}}\right)^{(\mathbf{X},\prec)}$ is called *translation invariant* if for each triple $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbf{X}$, we have $\mathbf{t}_{\mathbf{y}}(\mathbf{x}) = \mathbf{t}_{\mathbf{y}+\mathbf{z}}(\mathbf{x}+\mathbf{z})$, or equivalently, $\mathbf{t}_{\not\prec\mathbf{y}}(\mathbf{x}) = \mathbf{t}_{\not\prec\mathbf{y}+\mathbf{z}}(\mathbf{x}+\mathbf{z})$ and $\mathbf{t}_{\prec\mathbf{y}}(\mathbf{x}) = \mathbf{t}_{\prec\mathbf{y}+\mathbf{z}}(\mathbf{x}+\mathbf{z})$. An example of an invariant recursive template is shown in Figure 1.6.1.

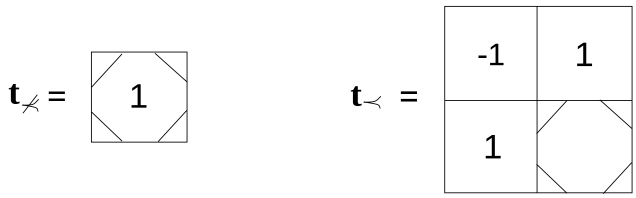

Figure 1.6.1. An example of an integer-valued invariant recursive template from $\mathbb{Z}^2$ to $\mathbb{Z}^2$.

If $\mathbf{t}$ is an invariant recursive template and has only one pixel defined on the target point of its nonrecursive support $S(\mathbf{t}_{\not\prec\mathbf{y}})$, then $\mathbf{t}$ is called a *simplified* recursive template. Pictorially, a simplified recursive template can be drawn the same way as a nonrecursive template since the recursive part and the nonrecursive part do not overlap. In particular, the recursive template shown in Figure 1.6.1 can be redrawn as illustrated in Figure 1.6.2

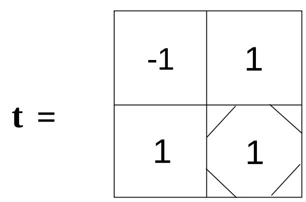

Figure 1.6.2. An example of an integer-valued simplified recursive template.

The notions of transpose and dual of a recursive template are defined in terms of those for nonrecursive templates. In particular, the *transpose* $\mathbf{t}'$ of a recursive template $\mathbf{t}$ is defined as $\mathbf{t}' = \left(\mathbf{t}'_{\not\prec}, \mathbf{t}'_{\prec}\right)$. Similarly, if $\mathbf{t} \in \left(\mathbb{R}^{\mathbf{X}}_{\pm\infty}, \mathbb{R}^{\mathbf{X}}_{\pm\infty}\right)^{(\mathbf{X},\prec)}$, then the *additive dual* of $\mathbf{t}$ is defined by $\mathbf{t}^* = \left(\mathbf{t}^*_{\not\prec}, \mathbf{t}^*_{\prec}\right)$. The *multiplicative dual* for recursive $\mathbb{R}^{\geq 0}_{\infty}$-valued templates is defined in a likewise fashion.

Operations between Images and Recursive Templates

In order to facilitate the discussion on recursive templates operations, we begin by extending the notions of the linear product $\oplus$, the additive maximum $\boxed{\vee}$, and the multiplicative maximum $\textcircled{\vee}$ to the corresponding recursive operations $\oplus_{\prec}$, $\boxed{\vee}_{\prec}$, and $\textcircled{\vee}_{\prec}$, respectively.

Let $\mathbf{X}$ and $\mathbf{Y}$ be finite subsets of $\mathbb{R}^n$ with $\mathbf{Y}$ partially ordered by $\prec$. If $\mathbf{a} \in \mathbb{R}^{\mathbf{X}}$ and $\mathbf{t} \in \left(\mathbb{R}^{\mathbf{X}}, \mathbb{R}^{\mathbf{Y}}\right)^{(\mathbf{Y},\prec)}$, then the *recursive linear image-template product* $\mathbf{a} \oplus_{\prec} \mathbf{t}$ is defined by

$$\mathbf{a} \oplus_{\prec} \mathbf{t} = \Bigg\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) : \mathbf{y} \in \mathbf{Y},\ \ \mathbf{b}(\mathbf{y}) = \sum_{\mathbf{x} \in S(\mathbf{t}_{\not\prec\mathbf{y}})} (\mathbf{a}(\mathbf{x}) \cdot \mathbf{t}_{\not\prec\mathbf{y}}(\mathbf{x})) + \sum_{\mathbf{z} \in S(\mathbf{t}_{\prec\mathbf{y}})} (\mathbf{b}(\mathbf{z}) \cdot \mathbf{t}_{\prec\mathbf{y}}(\mathbf{z})) \Bigg\}.$$

The recursive template operation $\oplus_{\prec}$ computes a new pixel value $\mathbf{b}(\mathbf{y})$ based on both the pixel values $\mathbf{a}(\mathbf{x})$ of the source image and some *previously* calculated new pixel values $\mathbf{b}(\mathbf{z})$ which are determined by the partial order $\prec$ and the region of support of the participating template. By definition of a recursive template, $\mathbf{z} \prec \mathbf{y}$ for every $\mathbf{z} \in S(\mathbf{t}_{\prec \mathbf{y}})$ and $\mathbf{y} \notin S(\mathbf{t}_{\prec \mathbf{y}})$. Therefore, $\mathbf{b}(\mathbf{y})$ is always recursively computable. Some partial orders that are commonly used in two-dimensional recursive transforms are forward and backward raster scanning and serpentine scanning.

It follows from the definition of $\oplus_{\prec}$ that the computation of a new pixel $\mathbf{b}(\mathbf{y})$ can be done only after all its predecessors (ordered by $\prec$) have been computed. Thus, in contrast to nonrecursive template operations, recursive template operations are not computed in a globally parallel fashion.

Note that if the recursive template $\mathbf{t}$ is defined such that $S(\mathbf{t}_{\prec \mathbf{y}}) = \varnothing$ for all $\mathbf{y} \in \mathbf{Y}$, then one obtains the usual nonrecursive template operation

$$\mathbf{a} \oplus_{\prec} \mathbf{t} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) \,:\, \mathbf{b}(\mathbf{y}) = \sum_{\mathbf{x} \in S(\mathbf{t}_{\not\prec \mathbf{y}})} (\mathbf{a}(\mathbf{x}) \cdot \mathbf{t}_{\not\prec \mathbf{y}}(\mathbf{x})),\ \mathbf{y} \in \mathbf{Y} \right\}.$$

Hence, recursive template operations are natural extensions of nonrecursive template operations.

Recursive additive maximum and multiplicative minimum are defined in a similar fashion. Specifically, if $\mathbf{a} \in \mathbb{R}_{\pm\infty}^{\mathbf{X}}$ and $\mathbf{t} \in \left(\mathbb{R}_{\pm\infty}^{\mathbf{X}}, \mathbb{R}_{\pm\infty}^{\mathbf{X}}\right)^{(\mathbf{X}, \prec)}$, then

$$\mathbf{b} = \mathbf{a} \boxed{\vee}_{\prec} \mathbf{t}$$

is defined by

$$\mathbf{b}(\mathbf{y}) = \bigvee_{\mathbf{x} \in S_{-\infty}(\mathbf{t}_{\not\prec \mathbf{y}})} [\mathbf{a}(\mathbf{x}) + \mathbf{t}_{\not\prec \mathbf{y}}(\mathbf{x})] \vee \bigvee_{\mathbf{z} \in S_{-\infty}(\mathbf{t}_{\prec \mathbf{y}})} [\mathbf{b}(\mathbf{z}) + \mathbf{t}_{\prec \mathbf{y}}(\mathbf{z})].$$

For $\mathbf{a} \in \left(\mathbb{R}_{\infty}^{\geq 0}\right)^{\mathbf{X}}$ and $\mathbf{t} \in \left(\left(\mathbb{R}_{\infty}^{\geq 0}\right)^{\mathbf{X}}, \left(\mathbb{R}_{\infty}^{\geq 0}\right)^{\mathbf{Y}}\right)^{(\mathbf{Y}, \prec)}$,

$$\mathbf{b} = \mathbf{a} \circledvee_{\prec} \mathbf{t}$$

is defined by

$$\mathbf{b}(\mathbf{y}) = \bigvee_{\mathbf{x} \in S(\mathbf{t}_{\not\prec \mathbf{y}})} [\mathbf{a}(\mathbf{x}) \times \mathbf{t}_{\not\prec \mathbf{y}}(\mathbf{x})] \vee \bigvee_{\mathbf{z} \in S(\mathbf{t}_{\prec \mathbf{y}})} [\mathbf{b}(\mathbf{z}) \times \mathbf{t}_{\prec \mathbf{y}}(\mathbf{z})].$$

The operations of the recursive additive minimum and multiplicative minimum ($\boxed{\wedge}_{\prec}$ and $\circledwedge_{\prec}$) are defined in the same straightforward fashion.

Recursive additive maximum, minimum as well as recursive multiplicative maximum and minimum are nonlinear operations. However, the recursive linear product remains a linear operation.

The basic recursive template operations described above can be easily generalized to the generic recursive image-template product by simple substitution of the specific operations, such as multiplication and addition, by the generic operations $\bigcirc$ and γ. More precisely, given a semiring $(\mathbb{F}, \gamma, \bigcirc)$ with identity, then one can define the generic recursive product

$$\circledgamma_{\prec} : \mathbb{F}^{\mathbf{X}} \times \left(\mathbb{F}^{\mathbf{X}}, \mathbb{F}^{\mathbf{Y}}\right)^{(\mathbf{Y}, \prec)} \to \mathbb{F}^{\mathbf{Y}}$$

by defining $\mathbf{b} = \mathbf{a} \circledgamma_{\prec} \mathbf{t}$ by

$$\mathbf{b}(\mathbf{y}) = \mathop{\Gamma}_{\mathbf{z} \in S\left(\mathbf{t}_{\nprec \mathbf{y}}\right)} \left[\mathbf{a}(\mathbf{x}) \bigcirc \mathbf{t}_{\nprec \mathbf{y}}(\mathbf{x})\right] \gamma \mathop{\Gamma}_{\mathbf{z} \in S(\mathbf{t}_{\prec \mathbf{y}})} \left[\mathbf{b}(\mathbf{z}) \bigcirc \mathbf{t}_{\prec \mathbf{y}}(\mathbf{z})\right].$$

Again, in addition to the basic recursive template operations discussed earlier, a wide variety of recursive template operations can be derived from the generalized recursive rule by substituting different binary operations for $\bigcirc$ and γ. Additionally, parameterized recursive templates are defined in the same manner as parametrized nonrecursive templates; namely as functions

$$\mathbf{t} : P \to \left(\mathbb{F}^{\mathbf{X}}, \mathbb{F}^{\mathbf{Y}}\right)^{(\mathbf{Y}, \prec)},$$

where P denotes the set of parameters, and $\mathbf{t}(p) = \left(\mathbf{t}(p)_{\nprec}, \mathbf{t}(p)_{\prec}\right)$ with $\mathbf{t}(p)_{\nprec} \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$ and $\mathbf{t}(p)_{\prec} \in \left(\mathbb{F}^{\mathbf{Y}}\right)^{(\mathbf{Y}, \prec)}$.

Summary of Recursive Template Operations

In the following list of pertinent recursive image-template products $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$ and $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{X}}, \mathbb{F}^{\mathbf{Y}}\right)^{(\mathbf{Y}, \prec)}$. As before, for each operation we assume the appropriate value set $\mathbb{F}$.

recursive generic product

$$\mathbf{a} \circledgamma_{\prec} \mathbf{t} = \left\{ \begin{array}{l} (\mathbf{y}, \mathbf{b}(\mathbf{y})) \; : \; \mathbf{b}(\mathbf{y}) = \mathbf{y} \in \mathbf{Y}, \mathop{\Gamma}_{\mathbf{z} \in S\left(\mathbf{t}_{\nprec \mathbf{y}}\right)} \left[\mathbf{a}(\mathbf{x}) \bigcirc \mathbf{t}_{\nprec \mathbf{y}}(\mathbf{x})\right] \gamma \\ \qquad\qquad\qquad\qquad\qquad\qquad \mathop{\Gamma}_{\mathbf{z} \in S(\mathbf{t}_{\prec \mathbf{y}})} \left[\mathbf{b}(\mathbf{z}) \bigcirc \mathbf{t}_{\prec \mathbf{y}}(\mathbf{z})\right] \end{array} \right\}$$

recursive linear product

$$\mathbf{a} \oplus_{\prec} \mathbf{t} = \left\{ \begin{array}{l} (\mathbf{y}, \mathbf{b}(\mathbf{y})) \; : \; \mathbf{b}(\mathbf{y}) = \mathbf{y} \in \mathbf{Y}, \sum_{\mathbf{x} \in S(\mathbf{t}_{\nprec \mathbf{y}})} (\mathbf{a}(\mathbf{x}) \cdot \mathbf{t}_{\nprec \mathbf{y}}(\mathbf{x})) + \\ \qquad\qquad\qquad\qquad\qquad\qquad \sum_{\mathbf{z} \in S(\mathbf{t}_{\prec \mathbf{y}})} (\mathbf{b}(\mathbf{z}) \cdot \mathbf{t}_{\prec \mathbf{y}}(\mathbf{z})) \end{array} \right\}$$

recursive additive max

$$\mathbf{a} \boxed{\vee}_{\prec} \mathbf{t} = \left\{ \begin{array}{l} (\mathbf{y}, \mathbf{b}(\mathbf{y})) \; : \; \mathbf{b}(\mathbf{y}) = \mathbf{y} \in \mathbf{Y}, \bigvee_{\mathbf{x} \in S_{-\infty}(\mathbf{t}_{\nprec \mathbf{y}})} \left[\mathbf{a}(\mathbf{x}) + \mathbf{t}_{\nprec \mathbf{y}}(\mathbf{x})\right] \vee \\ \qquad\qquad\qquad\qquad\qquad\qquad \bigvee_{\mathbf{z} \in S_{-\infty}(\mathbf{t}_{\prec \mathbf{y}})} \left[\mathbf{b}(\mathbf{z}) + \mathbf{t}_{\prec \mathbf{y}}(\mathbf{z})\right] \end{array} \right\}$$

recursive additive min

$$\mathbf{a} \boxed{\wedge}_{\prec} \mathbf{t} = \left\{ \begin{array}{l} (\mathbf{y}, \mathbf{b}(\mathbf{y})) \; : \; \mathbf{b}(\mathbf{y}) = \mathbf{y} \in \mathbf{Y}, \bigwedge_{\mathbf{x} \in S_{\infty}(\mathbf{t}_{\nprec \mathbf{y}})} \left[\mathbf{a}(\mathbf{x}) +' \mathbf{t}_{\nprec \mathbf{y}}(\mathbf{x})\right] \wedge \\ \qquad\qquad\qquad\qquad\qquad\qquad \bigwedge_{\mathbf{z} \in S_{\infty}(\mathbf{t}_{\prec \mathbf{y}})} \left[\mathbf{b}(\mathbf{z}) +' \mathbf{t}_{\prec \mathbf{y}}(\mathbf{z})\right] \end{array} \right\}$$

recursive multiplicative max

$$\mathbf{a} \circledvee_{\prec} \mathbf{t} = \left\{ \begin{array}{l} (\mathbf{y}, \mathbf{b}(\mathbf{y})) \; : \; \mathbf{b}(\mathbf{y}) = \mathbf{y} \in \mathbf{Y}, \bigvee_{\mathbf{x} \in S(\mathbf{t}_{\nprec \mathbf{y}})} \left[\mathbf{a}(\mathbf{x}) \times \mathbf{t}_{\nprec \mathbf{y}}(\mathbf{x})\right] \vee \\ \qquad\qquad\qquad\qquad\qquad\qquad \bigvee_{\mathbf{z} \in S(\mathbf{t}_{\prec \mathbf{y}})} \left[\mathbf{b}(\mathbf{z}) \times \mathbf{t}_{\prec \mathbf{y}}(\mathbf{z})\right] \end{array} \right\}$$

right multiplicative min

$$\mathbf{a} \mathbin{\textcircled{\scriptsize$\wedge$}}_{\prec} \mathbf{t} = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) \,:\, \mathbf{b}(\mathbf{y}) = \mathbf{y} \in \mathbf{Y}, \bigwedge_{\mathbf{x} \in S_{\infty}(\mathbf{t}_{\not\prec \mathbf{y}})} [\mathbf{a}(\mathbf{x}) \times' \mathbf{t}_{\not\prec \mathbf{y}}(\mathbf{x})] \wedge \bigwedge_{\mathbf{z} \in S_{\infty}(\mathbf{t}_{\prec \mathbf{y}})} [\mathbf{b}(\mathbf{z}) \times' \mathbf{t}_{\prec \mathbf{y}}(\mathbf{z})] \right\}$$

The definition of the left recursive product $\mathbf{t} \mathbin{\textcircled{\scriptsize$\gamma$}}_{\prec} \mathbf{a}$ is also straightforward. However, for sake of brevity and since the different left products are not required for the remainder of this text, we dispense with their formulation. Additional facts about recursive products, their properties and applications can be found in [1, 56, 57].

1.7. Neighborhoods

There are several types of template operations that are more easily implemented in terms of neighborhood operations. Typically, neighborhood operations replace template operations whenever the values in the support of a template consist only of the unit elements of the value set associated with the template. A template $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{X}}\right)^{\mathbf{Y}}$ with the property that for each $\mathbf{y} \in \mathbf{Y}$, the values in the support of $\mathbf{t}_{\mathbf{y}}$ consist only of the unit of $\mathbb{F}$ is called a *unit template*.

For example, the invariant template $\mathbf{t} \in \left(\mathbb{R}^{\mathbb{Z}^2}\right)^{\mathbb{Z}^2}$ shown in Figure 1.7.1 is a unit template with respect to the value set $(\mathbb{R}, +, \cdot)$ since the value 1 is the unit with respect to multiplication.

t =

1	1	1
1	1	1
1	1	1

Figure 1.7.1. The unit Moore template for the value set $(\mathbb{R}, +, \cdot)$.

Similarly, the template $\mathbf{r} \in \left(\mathbb{R}_{-\infty}^{\mathbb{Z}^2}\right)^{\mathbb{Z}^2}$ shown in Figure 1.7.2 is a unit template with respect to the value set $(\mathbb{R}_{-\infty}, \vee, +)$ since the value 0 is the unit with respect to the operation $+$.

r =

	0	
0	0	0
	0	

Figure 1.7.2. The unit von Neumann template for the value set $(\mathbb{R}_{-\infty}, \vee, +)$.

If $\mathbf{X} \subset \mathbb{Z}^2$ is an $m \times n$ array of points, $\mathbf{a} \in \mathbb{R}^{\mathbf{X}}$, and $\mathbf{t} \in \left(\mathbb{R}^{\mathbb{Z}^2}\right)^{\mathbb{Z}^2}$ is the 3×3 unit Moore template, then the values of the $m \times n$ image $\mathbf{b}$ obtained from the statement $\mathbf{b} := \mathbf{a} \bigoplus \mathbf{t}$ are computed by using the equation

$$\mathbf{b}(\mathbf{y}) = \sum_{\mathbf{x} \in \mathbf{X} \cap S(\mathbf{t_y})} \mathbf{a}(\mathbf{x}) \cdot \mathbf{t_y}(\mathbf{x}) = \sum_{\mathbf{x} \in \mathbf{X} \cap S(\mathbf{t_y})} \mathbf{a}(\mathbf{x}) \cdot 1 \, .$$

We need to point out that the difference between the mathematical equality $\mathbf{b} = \mathbf{a} \bigoplus \mathbf{t}$ and the pseudocode statement $\mathbf{b} := \mathbf{a} \bigoplus \mathbf{t}$ is that in the latter the new image is computed only for those points $\mathbf{y}$ for which $\mathbf{X} \cap S(\mathbf{t_y}) \neq \varnothing$. Observe that since $\mathbf{a}(\mathbf{x}) \cdot 1 = \mathbf{a}(\mathbf{x})$ and $M(\mathbf{y}) = S(\mathbf{t_y})$, where $M(\mathbf{y})$ denotes the Moore neighborhood of $\mathbf{y}$ (see Figure 1.2.2), it follows that

$$\mathbf{b}(\mathbf{y}) = \sum_{\mathbf{x} \in \mathbf{X} \cap M(\mathbf{y})} \mathbf{a}(\mathbf{x}) \, .$$

This observation leads to the notion of *neighborhood reduction*. In implementation, neighborhood reduction avoids unnecessary multiplication by the unit element and, as we shall shortly demonstrate, neighborhood reduction also avoids some standard boundary problems associated with image-template products.

To precisely define the notion of neighborhood reduction we need a more general notion of the reduce operation $\Gamma : \mathbb{F}^{\mathbf{X}} \to \mathbb{F}$, which was defined in terms of a binary operation γ on $\mathbb{F}$. The more general form of Γ is a function

$$\Gamma : \mathbb{F}^{\mathbf{X}}|_{2^{\mathbf{X}}} \to \mathbb{F} \, ,$$

where $\mathbb{F}^{\mathbf{X}}|_{2^{\mathbf{X}}} = \{\mathbf{a}|_{\mathbf{W}} : \mathbf{a} \in \mathbb{F}^{\mathbf{X}}, \ \mathbf{W} \subset \mathbf{X}\}$.

For example, if $\mathbb{F}^{\mathbf{X}} = \mathbb{R}^{\mathbf{X}}$, where $\mathbf{X} \subset \mathbb{Z}^2$ is an $m \times n$ array of points, then one such function could be defined as

$$\Sigma : \mathbb{R}^{\mathbf{X}}|_{2^{\mathbf{X}}} \to \mathbb{R} \, ,$$

where $\sum(\mathbf{a}|_{\mathbf{W}}) = \sum_{\mathbf{x} \in \mathbf{W}} \mathbf{a}(\mathbf{x})$. Another example would be to define

$$\Gamma : \mathbb{R}^{\mathbf{X}}|_{2^{\mathbf{X}}} \to \mathbb{R}$$

as $\Gamma(\mathbf{a}|_{\mathbf{W}}) = \frac{1}{card(\mathbf{W})} \sum_{\mathbf{x} \in \mathbf{W}} \mathbf{a}(\mathbf{x})$, then Γ implements the averaging function, which we shall denote by *average*. Similarly, for integer-valued images, the median reduction

$$median : \mathbb{N}^{\mathbf{X}}|_{2^{\mathbf{X}}} \to \mathbb{N}$$

is defined as $median(\mathbf{a}|_{\mathbf{W}}) = median\{\mathbf{a}(\mathbf{x}_{i_1}), \mathbf{a}(\mathbf{x}_{i_2}), \ldots, \mathbf{a}(\mathbf{x}_{i_k})\}$, where $\{\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_k}\} = \mathbf{W}$.

Now suppose $\mathbf{X} \subset \mathbf{Z}$, $\mathbf{t} \in \left(\mathbb{F}^{\mathbf{Z}}\right)^{\mathbf{Y}}$ is a unit template with respect to the operation $\bigcirc$ of the semiring $(\mathbb{F}, \gamma, \bigcirc)$, $N : \mathbf{Y} \to 2^{\mathbf{Z}}$ is a neighborhood system defined by $N(\mathbf{y}) = S(\mathbf{t_y})$, and $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$. It then follows that $\mathbf{b} := \mathbf{a} \textcircled{\gamma} \mathbf{t}$ is given by

$$\mathbf{b}(\mathbf{y}) = \Gamma_{\mathbf{x} \in \mathbf{X} \cap S(\mathbf{t_y})} (\mathbf{a}(\mathbf{x}) \bigcirc \mathbf{t_y}(\mathbf{x})) = \Gamma_{\mathbf{x} \in \mathbf{X} \cap N(\mathbf{y})} \mathbf{a}(\mathbf{x}) \, .$$

This observation leads to the following definition of an *image-neighborhood product*. Given $\mathbf{X} \subset \mathbf{Z}$, $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$, a reduction function $\Gamma : \mathbb{F}^{\mathbf{X}}|_{2^{\mathbf{X}}} \to \mathbb{F}$, and a neighborhood system $N : \mathbf{Y} \to 2^{\mathbf{Z}}$ (i.e., $N \in \left(2^{\mathbf{Z}}\right)^{\mathbf{Y}}$), then the *image-neighborhood product* $\mathbf{b} := \mathbf{a} \textcircled{Γ} N$ is defined by

$$\mathbf{b}(\mathbf{y}) = \Gamma\left(\mathbf{a}|_{\mathbf{X}\cap N(\mathbf{y})}\right)$$

for each $\mathbf{y} \in \mathbf{Y}$. Note that the product $\textcircled{$\Gamma$}$ is similar to the image template product $\textcircled{$\gamma$}$ in that $\textcircled{$\Gamma$}$ is a function

$$\textcircled{Γ} : \mathbb{F}^{\mathbf{X}} \times \left(2^{\mathbf{Z}}\right)^{\mathbf{Y}} \to \mathbb{F}^{\mathbf{Y}}.$$

In particular, if $\mathbf{a} \in \mathbb{R}^{\mathbf{X}}$, $M : \mathbb{Z}^2 \to 2^{\mathbb{Z}^2}$ is the Moore neighborhood, and $\mathbf{t} \in \left(\mathbb{R}^{\mathbb{Z}^2}\right)^{\mathbb{Z}^2}$ is the 3×3 unit Moore template defined earlier, then $\mathbf{a} \oplus \mathbf{t} = \mathbf{a} \oplus M$. Likewise, $\mathbf{a} \boxed{\vee}\, \mathbf{r} = \mathbf{a} \boxed{\vee}\, N$, where $\mathbf{r} \in \left(\mathbb{R}_{-\infty}^{\mathbb{Z}^2}\right)^{\mathbb{Z}^2}$ denotes the von Neumann unit template (Figure 1.7.2) and *N* denotes the von Neumann neighborhood (1.2.2). The latter equality stems from the fact that if $\mathbf{b} := \mathbf{a} \boxed{\vee}\, \mathbf{r}$ and $\mathbf{c} := \mathbf{a} \boxed{\vee}\, N$, then since $\mathbf{r}_{\mathbf{y}}(\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathbf{X} \cap S_{-\infty}(\mathbf{r}_{\mathbf{y}})$ and $S_{-\infty}(\mathbf{r}_{\mathbf{y}}) = N(\mathbf{y})$ for all points $\mathbf{y} \in \mathbb{Z}^2$, we have that

$$\mathbf{b}(\mathbf{y}) = \bigvee_{\mathbf{x}\in\mathbf{X}\cap S_{-\infty}(\mathbf{r}_{\mathbf{y}})} \mathbf{a}(\mathbf{x}) + \mathbf{r}_{\mathbf{y}}(\mathbf{x}) = \bigvee_{\mathbf{x}\in\mathbf{X}\cap N(\mathbf{y})} \mathbf{a}(\mathbf{x}) = \mathbf{c}(\mathbf{y})\,.$$

Unit templates act like *characteristic* functions in that they do not weigh a pixel, but simply note which pixels are in their support and which are not. When employed in the image-template operations of their semiring, they only serve to collect a number of values that need to be reduced by the gamma operation. For this reason, unit templates are also referred to as *characteristic templates*. Now suppose that we wish to describe a translation invariant unit template with a specific support such as the 3×3 support of the Moore template **t** shown in Figure 1.7.1. Suppose further that we would like this template to be used with a variety of reduction operations, for instance, summation and maximum. In fact, we *cannot* describe such an operand without regard of the image-template operation $\textcircled{$\gamma$}$ by which it will be used. For us to derive the expected results, the template must map all points in its support to the unitary value with respect to the combining operation $\bigcirc$. Thus, for the reduce operation of summation $\sum$, the unit values in the support must be 1, while for the maximum reduce operation $\bigvee$, the values in the support must all be 0. Therefore, we cannot define a single template operand to characterize a neighborhood for reduction without regard to the image-template operation to be used to reduce the values within the neighborhood. However, we can capture exactly the information of interest in unit templates with the simple notion of neighborhood function. Thus, for example, the Moore neighborhood *M* can be used to add the values in every 3×3 neighborhood as well as to find the maximum or minimum in such a neighborhood by using the statements $\mathbf{a} \oplus M$, $\mathbf{a} \boxed{\vee}\, M$, and $\mathbf{a} \boxed{\wedge}\, M$, respectively. This is one advantage for replacing unit templates with neighborhoods.

Another advantage of using neighborhoods instead of templates can be seen by considering the simple example of image smoothing by local averaging. Suppose $\mathbf{a} \in \mathbb{R}^{\mathbf{X}}$, where $\mathbf{X} \subset \mathbb{Z}^2$ is an $m \times n$ array of points, and $\mathbf{t} \in \left(\mathbb{R}^{\mathbb{Z}^2}\right)^{\mathbb{Z}^2}$ is the 3×3 unit Moore template with unit values 1. The image **b** obtained from the statement $\mathbf{b} := \frac{1}{9}(\mathbf{a} \oplus \mathbf{t})$ represents the image obtained from **a** by local averaging since the new pixel value $\mathbf{b}(\mathbf{y})$ is given by

$$\mathbf{b}(\mathbf{y}) = \frac{1}{9} \sum_{\mathbf{x}\in\mathbf{X}\cap S(\mathbf{t}_{\mathbf{y}})} \mathbf{a}(\mathbf{x}) \cdot \mathbf{t}_{\mathbf{y}}(\mathbf{x}) = \frac{1}{9} \sum_{\mathbf{x}\in\mathbf{X}\cap S(\mathbf{t}_{\mathbf{y}})} \mathbf{a}(\mathbf{x})\,.$$

Of course, there will be a boundary effect. In particular, if $\mathbf{X} = \{(i,j) : 1 \le i \le m,\ 1 \le j \le n\}$, then

$$\mathbf{b}(1,1) = \frac{1}{9}(\mathbf{a}(1,1) + \mathbf{a}(1,2) + \mathbf{a}(2,1) + \mathbf{a}(2,2)),$$

which is not the average of four points. One may either ignore this boundary effect (the most common choice), or one may one of several schemes to prevent it [1]. However, each of these schemes adds to the computational burden. A simpler and more elegant way is to use the Moore neighborhood function *M* combined with the averaging reduction $a \equiv average$. The simple statement $\mathbf{b} := \mathbf{a} \circledast M$ provides for the desired locally averaged image without boundary effect.

Neighborhood composition plays an important role in algorithm optimization and simplification of algebraic expressions. Given two neighborhood functions $N_1, N_2 : \mathbb{R}^n \to 2^{\mathbb{R}^n}$, then the *dilation* of N_1 by N_2, denoted by $N_1 \oplus N_2$, is a neighborhood function $N : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ which is defined as

$$N(\mathbf{y}) = \bigcup_{\mathbf{p} \in N_2(\mathbf{y})} (N_1(\mathbf{y}) + (\mathbf{p} - \mathbf{y})),$$

where $N(\mathbf{y}) + \mathbf{q} \equiv \{\mathbf{x} + \mathbf{q} : \mathbf{x} \in N(\mathbf{y})\}$. Just as for template composition, algorithm optimization can be achieved by use of the equation $\mathbf{a} \textcircled{\Gamma} (N_1 \oplus N_2) = (\mathbf{a} \textcircled{\Gamma} N_1) \textcircled{\Gamma} N_2$ for appropriate neighborhood functions and neighborhood reduction functions Γ. For $k \in \mathbb{N}$, the *k*th *iterate* of a neighborhood $N : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ is defined inductively as $N^k = N^{k-1} \oplus N$, where $N^0(\mathbf{y}) = \{\mathbf{y}\}\ \forall \mathbf{y} \in \mathbb{R}^n$.

Most neighborhood functions used in image processing are translation invariant subsets of $\mathbb{R}^n$ (in particular, subsets of $\mathbb{Z}^2 \subset \mathbb{R}^2$). A neighborhood function $N : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ is said to be translation invariant if $N(\mathbf{y} + \mathbf{p}) = N(\mathbf{y}) + \mathbf{p}$ for every point $\mathbf{p} \in \mathbb{R}^n$. Given a translation invariant neighborhood *N*, we define its *reflection* or *conjugate* N^* by $N^*(\mathbf{y}) = N^*(\mathbf{0}) + \mathbf{y}$, where $N^*(\mathbf{0}) = \{-\mathbf{x} : \mathbf{x} \in N(\mathbf{0})\}$ and $\mathbf{0} = (0, 0, \ldots, 0) \in \mathbb{R}^n$ denotes the origin. Conjugate neighborhoods play an important role in morphological image processing.

Note also that for a translation invariant neighborhood *N*, the *k*th iterate of *N* can be expressed in terms of the *sum* of sets

$$N^k(\mathbf{y}) = N^{k-1}(\mathbf{y}) + N(\mathbf{0}).$$

Furthermore, since $N^{k-1}(\mathbf{y}) + N(\mathbf{0}) = \bigcup_{\mathbf{q} \in N(\mathbf{0})} \left(N^{k-1}(\mathbf{y}) + \mathbf{q}\right)$ and $\bigcup_{\mathbf{q} \in N(\mathbf{0})} \left(N^{k-1}(\mathbf{y}) + \mathbf{q}\right) = \bigcup_{\mathbf{p} \in N(\mathbf{y})} \left(N^{k-1}(\mathbf{0}) + \mathbf{p}\right)$, we have the symmetric relation $N^k(\mathbf{y}) = N^{k-1}(\mathbf{0}) + N(\mathbf{y})$.

Summary of Image-Neighborhood Products

In the following list of pertinent image-neigborhood products $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$, $\mathbf{X} \subset \mathbf{Z}$, and $N \in \left(2^{\mathbf{Z}}\right)^{\mathbf{Y}}$. Again, for each operation we assume the appropriate value set $\mathbb{F}$.

generic neighborhood reduction

$$\mathbf{a} \textcircled{\Gamma} N = \left\{(\mathbf{y}, \mathbf{b}(\mathbf{y})) : \mathbf{b}(\mathbf{y}) = \Gamma \left(\mathbf{a}|_{N(\mathbf{y})}\right),\ \mathbf{y} \in \mathbf{Y}\right\}$$

neigborhood sum

$$\mathbf{a} \oplus N = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) \,:\, \mathbf{b}(\mathbf{y}) = \sum_{\mathbf{x} \in \mathbf{X} \cap N(\mathbf{y})} \mathbf{a}(\mathbf{x}) \,,\ \mathbf{y} \in \mathbf{Y} \right\}$$

neighborhood maximum

$$\mathbf{a} \boxtimes N = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) \,:\, \mathbf{b}(\mathbf{y}) = \bigvee_{\mathbf{x} \in \mathbf{X} \cap N(\mathbf{y})} a(\mathbf{x}) \,,\ \mathbf{y} \in \mathbf{Y} \right\}$$

neighborhood minimum

$$\mathbf{a} \boxtimes N = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) \,:\, \mathbf{b}(\mathbf{y}) = \bigwedge_{\mathbf{x} \in \mathbf{X} \cap N(\mathbf{y})} \mathbf{a}(\mathbf{x}) \,,\ \mathbf{y} \in \mathbf{Y} \right\}$$

Note that

$$\mathbf{a} \circledvee N = \left\{ (\mathbf{y}, \mathbf{b}(\mathbf{y})) \,:\, \mathbf{b}(\mathbf{y}) = \bigvee_{\mathbf{x} \in \mathbf{X} \cap N(\mathbf{y})} \mathbf{a}(\mathbf{x}) \,,\ \mathbf{y} \in \mathbf{Y} \right\}$$

and, therefore, $\mathbf{a} \circledvee N = \mathbf{a} \boxtimes N$. Similarly, $\mathbf{a} \circledwedge N = \mathbf{a} \boxtimes N$.

Although we did not address the issues of parameterized neighborhoods and recursive neighborhood operations, it should be clear that these are defined in the usual way by simple substitution of the appropriate neighborhood function for the corresponding Boolean template. For example, a parameterized neighborhood with parameters in the set *P* is a function $N : P \to \left(2^{\mathbf{Z}}\right)^{\mathbf{X}}$. Thus, for each parameter $p \in P$, $N(p)$ is a neighborhood system for $\mathbf{X}$ in $\mathbf{Z}$ since $N(p) : \mathbf{X} \to 2^{\mathbf{Z}}$. Similarly, a recursive neighborhood system for a partially ordered set $(\mathbf{X}, \prec)$ is a function $N = (N_{\not\prec}, N_{\prec}) : \mathbf{X} \to \left(2^{\mathbf{Z}}, 2^{\mathbf{X}}\right)$ satisfying the conditions that for each $\mathbf{x} \in \mathbf{X}$, $\mathbf{x} \notin N_{\prec}(\mathbf{x})$, and for each $\mathbf{z} \in N_{\prec}(\mathbf{x})$, $\mathbf{z} \prec \mathbf{x}$.

1.8. The *p*-Product

It is well known that in the linear domain template convolution products and image-template products are equivalent to matrix products and vector-matrix products, respectively [58, 1]. The notion of a generalized matrix product was developed in order to provide a general matrix theory approach to image-template products and template convolution products in both the linear and non-linear domains. This generalized matrix or *p*-product was first defined in Ritter [59]. This new matrix operation includes the matrix and vector products of linear algebra, the matrix product of minimax algebra [60], as well as generalized convolutions as special cases [59]. It provides for a transformation that combines the same or different types of values (or objects) into values of a possibly different type from those initially used in the combining operation. It has been shown that the *p*-product can be applied to express various image processing transforms in computing form [61, 62, 63]. In this document, however, we consider only products between matrices having the same type of values. In the subsequent discussion, $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}$ and the set of all $m \times n$ matrices with entries from $\mathbb{F}$ will be denoted by $\mathbb{F}_{m \times n}$. We will follow the usual convention of setting $\mathbb{F}^n = \mathbb{F}_{1 \times n}$ and view $\mathbb{F}^n$ as the set of all *n*-dimensional row vectors with entries from $\mathbb{F}$. Similarly, the set of all *m*-dimensional column vectors with entries from $\mathbb{F}$ is given by $(\mathbb{F}^m)' = [\mathbb{F}_{1 \times m}]' = \mathbb{F}_{m \times 1}$.

Let m, n, and p be positive integers with p dividing both m and n. Define the following correspondences:

$$c_p : \mathbb{Z}_p^+ \times \mathbb{Z}_{n/p}^+ \to \mathbb{Z}_n^+$$
$$\text{by } c_p(k,j) = (k-1)\frac{n}{p} + j,$$
$$\text{where } 1 \leq j \leq \frac{n}{p}, \text{and } 1 \leq k \leq p,$$

and

$$r_p : \mathbb{Z}_{m/p}^+ \times \mathbb{Z}_p^+ \to \mathbb{Z}_m^+$$
$$\text{by } r_p(i,k) = (i-1)p + k,$$
$$\text{where } 1 \leq k \leq p, \text{and } 1 \leq i \leq \frac{m}{p}.$$

Since $r_p(i,k) < r_p(i',k') \Leftrightarrow i < i'$ or $i = i'$ and $k < k'$, r_p linearizes the array $\mathbb{Z}_{m/p}^+ \times \mathbb{Z}_p^+$ using the row scanning order as shown:

$$\begin{bmatrix}
\overset{1}{(1,1)} & \overset{2}{(1,2)} & \cdots & \overset{k}{(1,k)} & \cdots & \overset{p}{(1,p)} \\
\overset{p+1}{(2,1)} & \overset{p+2}{(2,2)} & \cdots & \overset{p+k}{(2,k)} & \cdots & \overset{2p}{(2,p)} \\
\vdots & \vdots & & \vdots & & \vdots \\
\overset{(i-1)p+1}{(i,1)} & (i,2) & \cdots & \overset{(i-1)p+k}{(i,k)} & \cdots & \overset{ip}{(i,p)} \\
\vdots & \vdots & & \vdots & & \vdots \\
\overset{((m/p)-1)p+1}{(m/p,1)} & (m/p,2) & \cdots & (m/p,k) & \cdots & \overset{(m/p)p=m}{(m/p,p)}
\end{bmatrix}$$

It follows that the *row-scanning order* on $\mathbb{Z}_{m/p}^+ \times \mathbb{Z}_p^+$ is given by

$$(i,k) \leq (i',k') \;\Leftrightarrow\; r_p(i,k) \leq r_p(i',k')$$

or, equivalently, by

$$(i,k) \leq (i',k') \Leftrightarrow i < i' \text{ or } i = i' \text{ and } k \leq k'.$$

We define the one-to-one correspondence

$$f_p : \mathbb{Z}_l^+ \times \mathbb{Z}_{m/p}^+ \times \mathbb{Z}_p^+ \;\to\; \mathbb{Z}_l^+ \times \mathbb{Z}_m^+$$
$$\text{by } f_p : (x,y,z) \;\rightarrowtail\; (x, r_p(y,z)).$$

The one-to-one correspondence allows us to re-index the entries of a matrix $A = (a_{s,t}) \in \mathbb{F}_{l \times m}$ in terms of a triple index $a_{s,(i,k)}$ by using the convention

$$a_{s,(i,k)} = a_{s,t} \;\Leftrightarrow\; r_p(i,k) = t,$$
$$\text{where } 1 \leq i \leq m/p \text{ and } 1 \leq k \leq p.$$

Example: Suppose $l = 2,\ m = 6$ and $p = 2$. Then $m/p = 3,\ 1 \le k \le p = 2,$ and $1 \le i \le m/p = 3$. Hence for $A = (a_{s,t}) \in \mathbb{F}_{2\times 6}$, we have

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \end{pmatrix}$$
$$= \begin{pmatrix} a_{1,(1,1)} & a_{1,(1,2)} & a_{1,(2,1)} & a_{1,(2,2)} & a_{1,(3,1)} & a_{1,(3,2)} \\ a_{2,(1,1)} & a_{2,(1,2)} & a_{2,(2,1)} & a_{2,(2,2)} & a_{2,(3,1)} & a_{2,(3.2)} \end{pmatrix}.$$

The factor $\mathbb{Z}_n^+$ of the Cartesian product $\mathbb{Z}_n^+ \times \mathbb{Z}_q^+$ is decomposed in a similar fashion. Here the row-scanning map is given by

$$c_p : \mathbb{Z}_p^+ \times \mathbb{Z}_{n/p}^+ \to \mathbb{Z}_n^+$$
$$\text{where } c_p(k, j) = (k-1)(n/p) + j,$$
$$1 \le j \le n/p, \text{ and } 1 \le k \le p\,.$$

This allows us to re-index the entries of a matrix $B = (b_{s,t}) \in M_{n\times q}(\mathbb{F})$ in terms of a triple index $b_{(k,j),t}$ by using the convention

$$b_{(k,j),t} = b_{s,t} \Leftrightarrow c_p(k,j) = s\,,$$
$$\text{where } 1 \le k \le p \text{ and } 1 \le j \le n/p\,.$$

Example: Suppose $n = 4,\ q = 3$ and $p = 2$. Then $n/p = 2,\ 1 \le k \le p = 2,$ and $1 \le j \le n/p = 2$. Hence for $B = (b_{s,t}) \in \mathbb{F}_{n\times q}$, we have

$$B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \end{pmatrix} = \begin{pmatrix} b_{(1,1),1} & b_{(1,1),2} & b_{(1,1),3} \\ b_{(1,2),1} & b_{(1,2),2} & b_{(1,2),3} \\ b_{(2,1),1} & b_{(2,1),2} & b_{(2,1),3} \\ b_{(2,2),1} & b_{(2,2),2} & b_{(2,2),3} \end{pmatrix}.$$

Now let $A = (a_{sj'}) \in \mathbb{F}_{l\times m}$ and $B = (b_{i',t}) \in \mathbb{F}_{n\times q}$. Using the maps r_p and c_p, *A* and *B* can be rewritten as

$$A = \left(a_{s,(i,k)}\right)_{l\times m}, \text{ where } 1 \le s \le l,\ 1 \le r_p(i,k) = j' \le m, \text{ and}$$
$$B = \left(b_{(k,j),t}\right)_{n\times q}, \text{ where } 1 \le c_p(k,j) = i' \le n \text{ and } 1 \le t \le q.$$

The *p-product* or *generalized matrix product* of *A* and *B* is denoted by $A \bigoplus_p B$, and is the matrix

$$C = A \bigoplus_p B \in \mathbb{F}_{l(n/p)\times(m/p)q}$$

defined by

$$c_{(s,j)(i,t)} = \sum_{k=1}^{p} \left(a_{s,(i,k)} b_{(k,j),t}\right) = \left(a_{s,(i,1)} b_{(1,j),t}\right) + \ldots + \left(a_{s,(i,p)} b_{(p,j),t}\right),$$

where $c_{(s,j)(i,t)}$ denotes the (s,j)th row and (i,t)th column entry of C. Here we use the lexicographical order $(s,j) < (s',j') \Leftrightarrow s < s'$ or if $s = s', j < j'$. Thus, the matrix C has the following form:

$$\begin{bmatrix}
c_{(1,1)(1,1)} & \cdots & c_{(1,1)(1,q)} & c_{(1,1)(2,1)} & \cdots & c_{(1,1)(2,q)} & \cdots & c_{(1,1)(i,t)} & \cdots & c_{(1,1)\left(\frac{m}{p},q\right)} \\
c_{(1,2)(1,1)} & \cdots & c_{(1,2)(1,q)} & c_{(1,2)(2,1)} & \cdots & c_{(1,2)(2,q)} & \cdots & c_{(1,2)(i,t)} & \cdots & c_{(1,2)\left(\frac{m}{p},q\right)} \\
\vdots & & \vdots & & & \vdots & & \vdots & & \vdots \\
c_{\left(1,\frac{n}{p}\right)(1,1)} & \cdots & c_{\left(1,\frac{n}{p}\right)(1,q)} & c_{\left(1,\frac{n}{p}\right)(2,1)} & \cdots & c_{\left(1,\frac{n}{p}\right)(2,q)} & \cdots & c_{\left(1,\frac{n}{p}\right)(i,t)} & \cdots & c_{\left(1,\frac{n}{p}\right)\left(\frac{m}{p},q\right)} \\
c_{(2,1)(1,1)} & \cdots & c_{(2,1)(1,q)} & c_{(2,1)(2,1)} & \cdots & c_{(2,1)(2,q)} & \cdots & c_{(2,1)(i,t)} & \cdots & c_{(2,1)\left(\frac{m}{p},q\right)} \\
\vdots & & \vdots & \vdots & & \vdots & & \vdots & & \vdots \\
c_{\left(2,\frac{n}{p}\right)(1,1)} & \cdots & c_{\left(2,\frac{n}{p}\right)(1,q)} & c_{\left(2,\frac{n}{p}\right)(2,1)} & \cdots & c_{\left(2,\frac{n}{p}\right)(2,q)} & \cdots & c_{\left(2,\frac{n}{p}\right)(i,t)} & \cdots & c_{\left(2,\frac{n}{p}\right)\left(\frac{m}{p},q\right)} \\
\vdots & & \vdots & \vdots & & \vdots & & \vdots & & \vdots \\
c_{(s,j)(1,1)} & \cdots & c_{(s,j)(1,q)} & c_{(s,j)(2,1)} & \cdots & c_{(s,j)(2,q)} & \cdots & \underline{c_{(s,j)(i,t)}} & \cdots & c_{(s,j)\left(\frac{m}{p},q\right)} \\
\vdots & & \vdots & \vdots & & \vdots & & \vdots & & \vdots \\
c_{(l,1)(1,1)} & \cdots & c_{(l,1)(1,q)} & c_{(l,1)(2,1)} & \cdots & c_{(l,1)(2,q)} & \cdots & c_{(l,1)(i,t)} & \cdots & c_{(l,1)\left(\frac{m}{p},q\right)} \\
\vdots & & \vdots & \vdots & & \vdots & & \vdots & & \vdots \\
c_{\left(l,\frac{n}{p}\right)(1,1)} & \cdots & c_{\left(l,\frac{n}{p}\right)(1,q)} & c_{\left(l,\frac{n}{p}\right)(2,1)} & \cdots & c_{\left(l,\frac{n}{p}\right)(2,q)} & \cdots & c_{\left(l,\frac{n}{p}\right)(i,t)} & \cdots & c_{\left(l,\frac{n}{p}\right)\left(\frac{m}{p},q\right)}
\end{bmatrix}$$

The entry $c_{(s,j)(i,t)}$ in the (*s*,*j*)-row and (*i*,*t*)-column is underlined for emphasis.

To provide an example, suppose that $l = 2,\ m = 6,\ n = 4,$ and $q = 3$. Then for p = 2, one obtains $m/p = 3,\ n/p = 2$ and $1 \leq k \leq 2$. Now let

$$\mathrm{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \end{pmatrix} \in \mathrm{M}_{2\times 6}(\mathbb{R})$$

and

$$B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \end{pmatrix} \in \mathbb{R}_{4\times 3}\,.$$

Then the (2,1)-row and (2,3)-column element $c_{(2,1)(2,3)}$ of the matrix

$$C = A \bigoplus\nolimits_2 B \in \mathbb{R}_{l(n/p)\times(m/p)q} = \mathbb{R}_{4\times 9}$$

is given by

$$\begin{aligned} c_{(2,1)(2,3)} &= \sum_{k=1}^{2} a_{2,r_2(2,k)} \cdot b_{c_2(k,1),3} \\ &= a_{2,r_2(2,1)} \cdot b_{c_2(1,1),3} + a_{2,r_2(2,2)} \cdot b_{c_2(2,1),3} \\ &= a_{23} \cdot b_{13} + a_{24} \cdot b_{33}\,. \end{aligned}$$

Thus, in order to compute $c_{(2,1)(2,3)}$, the two underlined elements of *A* are combined with the two underlined elements of *B* as illustrated:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & \underline{a_{23}} & \underline{a_{24}} & a_{25} & a_{26} \end{pmatrix} \oplus_2 \begin{pmatrix} b_{11} & b_{12} & \underline{b_{13}} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & \underline{b_{33}} \\ b_{41} & b_{42} & b_{43} \end{pmatrix}$$

$$= \begin{pmatrix} a_{1,r_2(1,1)} & a_{1,r_2(1,2)} & a_{1,r_2(2,1)} & a_{1,r_2(2,2)} & a_{1,r_2(3,1)} & a_{1,r_2(3,2)} \\ a_{2,r_2(1,1)} & a_{2,r_2(1,2)} & \underline{a_{2,r_2(2,1)}} & \underline{a_{2,r_2(2,2)}} & a_{2,r_2(3,1)} & a_{2,r_2(3.2)} \end{pmatrix} \oplus_2$$

$$\begin{pmatrix} b_{c_2(1,1),1} & b_{c_2(1,1),2} & \underline{b_{c_2(1,1),3}} \\ b_{c_2(1,2),1} & b_{c_2(1,2),2} & b_{c_2(1,2),3} \\ b_{c_2(2,1),1} & b_{c_2(2,1),2} & \underline{b_{c_2(2,1),3}} \\ b_{c_2(2,2),1} & b_{c_2(2,2),2} & b_{c_2(2,2),3} \end{pmatrix}$$

$$= \begin{pmatrix} c_{(1,1)(1,1)} & c_{(1,1)(1,2)} & \cdots & c_{(1,1)(2,3)} & \cdots & c_{(1,1)(3,3)} \\ c_{(1,2)(1,1)} & c_{(1,2)(1,2)} & \cdots & c_{(1,2)(2,3)} & \cdots & c_{(1,2)(3,3)} \\ c_{(2,1)(1,1)} & c_{(2,1)(1,2)} & \cdots & \underline{c_{(2,1)(2,3)}} & \cdots & c_{(2,1)(3,3)} \\ c_{(2,2)(1,1)} & c_{(2,2)(1,2)} & \cdots & c_{(2,2)(2,3)} & \cdots & c_{(2,2)(3,3)} \end{pmatrix}$$

$$= \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{16} & \cdots & c_{19} \\ c_{21} & c_{22} & \cdots & c_{26} & \cdots & c_{29} \\ c_{31} & c_{32} & \cdots & \underline{c_{36}} & \cdots & c_{39} \\ c_{41} & c_{42} & \cdots & c_{46} & \cdots & c_{49} \end{pmatrix}.$$

In particular,

$$\begin{pmatrix} 1 & 2 & 0 & 5 & 4 & 3 \\ 2 & 3 & 4 & 1 & 0 & 6 \end{pmatrix} \oplus_2 \begin{pmatrix} 2 & 6 & 1 \\ 1 & 3 & 2 \\ 2 & 2 & 5 \\ 3 & 0 & 4 \end{pmatrix} = \begin{pmatrix} 6 & 10 & 11 & 10 & 10 & 25 & 14 & 30 & 19 \\ 7 & 3 & 10 & 15 & 0 & 20 & 13 & 12 & 20 \\ 10 & 18 & 17 & 10 & 26 & 9 & 12 & 12 & 30 \\ 11 & 6 & 16 & 7 & 12 & 12 & 18 & 0 & 24 \end{pmatrix}.$$

If

$$A = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \text{ and } B = \begin{pmatrix} 4 \\ 2 \\ 6 \\ 3 \end{pmatrix},$$

then

$$(A \oplus_2 B)' = (4,\, 2,\, 2,\, 1) \neq (4,\, -2,\, 6,\, -3) = B' \oplus_2 A'.$$

This shows that the transpose property, which holds for the regular matrix product, is generally false for the *p*-product. The reason is that the *p*-product is not a dual operation in the transpose domain. In order to make the transpose property hold we define the *dual* operation $\oplus_p'$ of $\oplus_p$ by

$$A \oplus_p' B \equiv \left(B' \oplus_p A'\right)'.$$

It follows that

$$A \oplus_p B = \left(B' \oplus_p' A'\right)'$$

and the *p*-product is the dual operation of $\oplus_p'$. In particular, we now have the transpose property $\left(A \oplus_p B\right)' = B' \oplus_p' A'$.

Since the operation $\bigoplus'_p$ is defined in terms of matrix transposition, labeling of matrix indices are reversed. Specifically, if $A = (a_{st})$ is an $l \times m$ matrix, then A gets reindexed as $A = \big(a_{s,(kj)}\big)$, using the convention

$$\begin{gathered} a_{s,(k,j)} = a_{s,t} \;\Leftrightarrow\; c_p(k,j) = t\,, \\ \text{where } \; 1 \le j \le m/p \text{ and } 1 \le k \le p\,. \end{gathered}$$

Similarly, if $B = (b_{st})$ is an $n \times q$ matrix, then the entries of B are relabeled as $b_{(i,k),t}$, using the convention

$$\begin{gathered} b_{(i,k),t} = b_{s,t} \;\Leftrightarrow\; r_p(i,k) = s\,, \\ \text{where } \; 1 \le k \le p \text{ and } 1 \le i \le n/p\,. \end{gathered}$$

The product $A \bigoplus'_p B = C$ is then defined by the equation

$$c_{(i,s)(t,j)} = \sum_{k=1}^{p} \big(a_{s,(k,j)} b_{(i,k),t}\big) = \big(a_{s,(1,j)} b_{(i,1),t}\big) + \ldots + \big(a_{s,(p,j)} b_{(i,p),t}\big).$$

Note that the dimension of C is $l \cdot \frac{n}{p} \times \frac{m}{p} \cdot q$.

To provide a specific example of the dual operation $\bigoplus'_p$, suppose that

$$\mathrm{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix} \text{ and } B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \\ b_{41} & b_{42} \\ b_{51} & b_{52} \\ b_{61} & b_{62} \end{pmatrix}.$$

In this case we have $l = 3,\ m = 4,\ n = 6,$ and $q = 2$. Thus, for $p = 2$ and using the scheme described above, the reindexed matrices have form

$$A = \begin{pmatrix} a_{1,(1,1)} & a_{1,(1,2)} & a_{1,(2,1)} & a_{1,(2,2)} \\ a_{2,(1,1)} & a_{2,(1,2)} & a_{2,(2,1)} & a_{2,(2,2)} \\ a_{3,(1,1)} & a_{3,(1,2)} & a_{3,(2,1)} & a_{3,(2,2)} \end{pmatrix} \text{ and } B = \begin{pmatrix} b_{(1,1),1} & b_{(1,1),2} \\ b_{(1,2),1} & b_{(1,2),2} \\ b_{(2,1),1} & b_{(2,1),2} \\ b_{(2,2),1} & b_{(2,2),2} \\ b_{(3,1),1} & b_{(3,1),2} \\ b_{(3,2),1} & b_{(3,2),2} \end{pmatrix}.$$

According to the dual product definition, the matrix $A \bigoplus'_2 B = C$ is a 9×4 matrix given by

$$C = \begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ \vdots & \vdots & \vdots & \vdots \\ c_{61} & c_{62} & \underline{c_{63}} & c_{64} \\ \vdots & \vdots & \vdots & \vdots \\ c_{91} & c_{92} & c_{93} & c_{94} \end{pmatrix} = \begin{pmatrix} c_{(1,1)(1,1)} & c_{(1,1)(1,2)} & c_{(1,1)(2,1)} & c_{(1,1)(2,2)} \\ c_{(1,2)(1,1)} & c_{(1,2)(1,2)} & c_{(1,2)(2,1)} & c_{(1,2)(2,2)} \\ \vdots & \vdots & \vdots & \vdots \\ c_{(2,3)(1,1)} & c_{(2,3)(1,2)} & \underline{c_{(2,3)(2,1)}} & c_{(2,3)(2,2)} \\ \vdots & \vdots & \vdots & \vdots \\ c_{(3,3)(1,1)} & c_{(3,3)(1,2)} & c_{(3,3)(2,1)} & c_{(3,3)(2,2)} \end{pmatrix}.$$

The underlined element c_{63} is obtained by using the formula:

$$c_{63} = c_{(2,3)(2,1)} = \sum_{k=1}^{2} a_{3,(k,1)} b_{(2,k),2} = a_{3,(1,1)} b_{(2,1),2} + a_{3,(2,1)} b_{(2,2),2}\,.$$

Thus, in order to compute c_{63}, the two underlined elements of *A* are combined with the two underlined elements of *B* as illustrated:

$$\begin{pmatrix} \underline{a_{11}} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \underline{a_{31}} & a_{32} & \underline{a_{33}} & a_{34} \end{pmatrix} \oplus'_2 \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & \underline{b_{32}} \\ b_{41} & \underline{b_{42}} \\ b_{51} & b_{52} \\ b_{61} & b_{62} \end{pmatrix}.$$

As a final observation, note that the matrices *A*, *B*, and *C* in this example have the form of the transposes of the matrices *B*, *A*, and *C*, respectively, of the previous example.

1.9. References

[1] G. Ritter, “Image algebra with applications.” Unpublished manuscript, available via anonymous ftp from `ftp://ftp.cis.ufl.edu/pub/src/ia/documents`, 1994.

[2] S. Unger, “A computer oriented toward spatial problems,” *Proceedings of the IRE*, vol. 46, pp. 1144–1750, 1958.

[3] J. von Neumann, “The general logical theory of automata,” in *Cerebral Mechanism in Behavior: The Hixon Symposium*, New York, NY: John Wiley & Sons, 1951.

[4] J. von Neumann, *Theory of Self-Reproducing Automata*. Urbana, IL: University of Illinois Press, 1966.

[5] K. Batcher, “Design of a massively parallel processor,” *IEEE Transactions on Computers*, vol. 29, no. 9, pp. 836–840, 1980.

[6] M. Duff, D. Watson, T. Fountain, and G. Shaw, “A cellular logic array for image processing,” *Pattern Recognition*, vol. 5, pp. 229–247, Sept. 1973.

[7] M. J. B. Duff, “Review of CLIP image processing system,” in *Proceedings of the National Computing Conference*, pp. 1055–1060, AFIPS, 1978.

[8] M. Duff, “Clip4,” in *Special Computer Architectures for Pattern Processing* (K. Fu and T. Ichikawa, eds.), ch. 4, pp. 65–86, Boca Raton, FL: CRC Press, 1982.

[9] T. Fountain, K. Matthews, and M. Duff, “The CLIP7A image processor,” *IEEE Pattern Analysis and Machine Intelligence*, vol. 10, no. 3, pp. 310–319, 1988.

[10] L. Uhr, “Pyramid multi-computer structures, and augmented pyramids,” in *Computing Structures for Image Processing* (M. Duff, ed.), pp. 95–112, London: Academic Press, 1983.

[11] L. Uhr, *Algorithm-Structured Computer Arrays and Networks*. New York, NY: Academic Press, 1984.

[12] W. Hillis, *The Connection Machine*. Cambridge, MA: The MIT Press, 1985.

[13] J. Klein and J. Serra, “The texture analyzer,” *Journal of Microscopy*, vol. 95, 1972.

[14] R. Lougheed and D. McCubbrey, “The cytocomputer: A practical pipelined image processor,” in *Proceedings of the Seventh International Symposium on Computer Architecture*, pp. 411–418, 1980.

[15] S. Sternberg, "Biomedical image processing," *Computer*, vol. 16, Jan. 1983.

[16] R. Lougheed, "A high speed recirculating neighborhood processing architecture," in *Architectures and Algorithms for Digital Image Processing II*, vol. 534 of *Proceedings of SPIE*, pp. 22–33, 1985.

[17] E. Cloud and W. Holsztynski, "Higher efficiency for parallel processors," in *Proceedings IEEE Southcon 84*, (Orlando, FL), pp. 416–422, Mar. 1984.

[18] E. Cloud, "The geometric arithmetic parallel processor," in *Proceedings Frontiers of Massively Parallel Processing*, George Mason University, Oct. 1988.

[19] E. Cloud, "Geometric arithmetic parallel processor: Architecture and implementation," in *Parallel Architectures and Algorithms for Image Understanding* (V. Prasanna, ed.), (Boston, MA.), Academic Press, Inc., 1991.

[20] H. Minkowski, "Volumen und oberflache," *Mathematische Annalen*, vol. 57, pp. 447–495, 1903.

[21] H. Minkowski, *Gesammelte Abhandlungen*. Leipzig-Berlin: Teubner Verlag, 1911.

[22] H. Hadwiger, *Vorlesungen Über Inhalt, Oberflæche und Isoperimetrie*. Berlin: Springer-Verlag, 1957.

[23] G. Matheron, *Random Sets and Integral Geometry*. New York: Wiley, 1975.

[24] J. Serra, "Introduction a la morphologie mathematique," booklet no. 3, Cahiers du Centre de Morphologie Mathematique, Fontainebleau, France, 1969.

[25] J. Serra, "Morphologie pour les fonctions "a peu pres en tout ou rien"," technical report, Cahiers du Centre de Morphologie Mathematique, Fontainebleau, France, 1975.

[26] J. Serra, *Image Analysis and Mathematical Morphology*. London: Academic Press, 1982.

[27] T. Crimmins and W. Brown, "Image algebra and automatic shape recognition," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-21, pp. 60–69, Jan. 1985.

[28] R. Haralick, S. Sternberg, and X. Zhuang, "Image analysis using mathematical morphology: Part I," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, pp. 532–550, July 1987.

[29] R. Haralick, L. Shapiro, and J. Lee, "Morphological edge detection," *IEEE Journal of Robotics and Automation*, vol. RA-3, pp. 142–157, Apr. 1987.

[30] P. Maragos and R. Schafer, "Morphological skeleton representation and coding of binary images," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, pp. 1228–1244, Oct. 1986.

[31] P. Maragos and R. Schafer, "Morphological filters Part II : Their relations to median, order-statistic, and stack filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, pp. 1170–1184, Aug. 1987.

[32] P. Maragos and R. Schafer, "Morphological filters Part I: Their set-theoretic analysis and relations to linear shift-invariant filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, pp. 1153–1169, Aug. 1987.

[33] J. Davidson and A. Talukder, "Template identification using simulated annealing in morphology neural networks," in *Second Annual Midwest Electro-Technology Conference*, (Ames, IA), pp. 64–67, IEEE Central Iowa Section, Apr. 1993.

[34] J. Davidson and F. Hummer, "Morphology neural networks: An introduction with applications," *IEEE Systems Signal Processing*, vol. 12, no. 2, pp. 177–210, 1993.

[35] E. Dougherty, "Unification of nonlinear filtering in the context of binary logical calculus, part ii: Gray-scale filters," *Journal of Mathematical Imaging and Vision*, vol. 2, pp. 185–192, Nov. 1992.

[36] D. Schonfeld and J. Goutsias, “Optimal morphological pattern restoration from noisy binary images,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 14–29, Jan. 1991.

[37] J. Goutsias, “On the morphological analysis of discrete random shapes,” *Journal of Mathematical Imaging and Vision*, vol. 2, pp. 193–216, Nov. 1992.

[38] L. Koskinen and J. Astola, “Asymptotic behaviour of morphological filters,” *Journal of Mathematical Imaging and Vision*, vol. 2, pp. 117–136, Nov. 1992.

[39] S. R. Sternberg, “Language and architecture for parallel image processing,” in *Proceedings of the Conference on Pattern Recognition in Practice*, (Amsterdam), May 1980.

[40] S. Sternberg, “Overview of image algebra and related issues,” in *Integrated Technology for Parallel Image Processing* (S. Levialdi, ed.), London: Academic Press, 1985.

[41] P. Maragos, *A Unified Theory of Translation-Invariant Systems with Applications to Morphological Analysis and Coding of Images*. Ph.D. dissertation, Georgia Institute of Technology, Atlanta, 1985.

[42] J. Davidson, *Lattice Structures in the Image Algebra and Applications to Image Processing*. PhD thesis, University of Florida, Gainesville, FL, 1989.

[43] J. Davidson, “Classification of lattice transformations in image processing,” *Computer Vision, Graphics, and Image Processing: Image Understanding*, vol. 57, pp. 283–306, May 1993.

[44] P. Miller, “Development of a mathematical structure for image processing,” optical division tech. report, Perkin-Elmer, 1983.

[45] J. Wilson, D. Wilson, G. Ritter, and D. Langhorne, “Image algebra FORTRAN language, version 3.0,” Tech. Rep. TR-89–03, University of Florida CIS Department, Gainesville, 1989.

[46] J. Wilson, “An introduction to image algebra Ada,” in *Image Algebra and Morphological Image Processing II*, vol. 1568 of *Proceedings of SPIE*, (San Diego, CA), pp. 101–112, July 1991.

[47] J. Wilson, G. Fischer, and G. Ritter, “Implementation and use of an image processing algebra for programming massively parallel computers,” in *Frontiers ’88: The Second Symposium on the Frontiers of Massively Parallel Computation*, (Fairfax, VA), pp. 587–594, 1988.

[48] G. Fischer and M. Rowlee, “Computation of disparity in stereo images on the Connection Machine,” in *Image Algebra and Morphological Image Processing*, vol. 1350 of *Proceedings of SPIE*, pp. 328–334, 1990.

[49] D. Crookes, P. Morrow, and P. McParland, “An algebra-based language for image processing on transputers,” *IEE Third Annual Conference on Image Processing and its Applications*, vol. 307, pp. 457–461, July 1989.

[50] D. Crookes, P. Morrow, and P. McParland, “An implementation of image algebra on transputers,” tech. rep., Department of Computer Science, Queen’s University of Belfast, Northern Ireland, 1990.

[51] J. Wilson, “Supporting image algebra in the C++ language,” in *Image Algebra and Morphological Image Processing IV*, vol. 2030 of *Proceedings of SPIE*, (San Diego, CA), pp. 315–326, July 1993.

[52] University of Florida Center for Computer Vision and Visualization, *Software User Manual for the* `iac++` *Class Library*, version 1.0 ed., 1994. Avialable via anonymous ftp from `ftp.cis.ufl.edu` in `/pub/ia/documents`.

[53] G. Birkhoff and J. Lipson, "Heterogeneous algebras," *Journal of Combinatorial Theory*, vol. 8, pp. 115–133, 1970.

[54] University of Florida Center for Computer Vision and Visualization, *Software User Manual for the* `iac++` *Class Library*, version 2.0 ed., 1995. Avialable via anonymous ftp from `ftp.cis.ufl.edu` in `/pub/ia/documents`.

[55] G. Ritter, J. Davidson, and J. Wilson, "Beyond mathematical morphology," in *Visual Communication and Image Processing II*, vol. 845 of *Proceedings of SPIE*, (Cambridge, MA), pp. 260–269, Oct. 1987.

[56] D. Li, *Recursive Operations in Image Algebra and Their Applications to Image Processing*. PhD thesis, University of Florida, Gainesville, FL, 1990.

[57] D. Li and G. Ritter, "Recursive operations in image algebra," in *Image Algebra and Morphological Image Processing*, vol. 1350 of *Proceedings of SPIE*, (San Diego, CA), July 1990.

[58] R. Gonzalez and P. Wintz, *Digital Image Processing*. Reading, MA: Addison-Wesley, second ed., 1987.

[59] G. Ritter, "Heterogeneous matrix products," in *Image Algebra and Morphological Image Processing II*, vol. 1568 of *Proceedings of SPIE*, (San Diego, CA), pp. 92–100, July 1991.

[60] R. Cuninghame-Green, *Minimax Algebra: Lecture Notes in Economics and Mathematical Systems 166*. New York: Springer-Verlag, 1979.

[61] G. Ritter and H. Zhu, "The generalized matrix product and its applications," *Journal of Mathematical Imaging and Vision*, vol. 1(3), pp. 201–213, 1992.

[62] H. Zhu and G. X. Ritter, "The generalized matrix product and the wavelet transform," *Journal of Mathematical Imaging and Vision*, vol. 3, no. 1, pp. 95–104, 1993.

[63] H. Zhu and G. X. Ritter, "The *p*-product and its applications in signal processing," *SIAM Journal of Matrix Analysis and Applications*, vol. 16, no. 2, pp. 579–601, 1995.