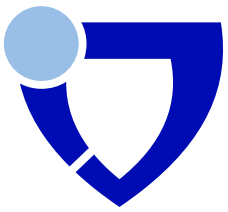


JACOBS
UNIVERSITY

Angelica Garcia Gutierrez, Peter Baumann

Modeling Fundamental Geo-Raster Operations with Array Algebra

Technical Report No. 7
June 2007

School of Engineering and Science

Modeling Fundamental Geo-Raster Operations with Array Algebra

Towards the improvement of query response times in Raster Image Databases

Angelica Garcia Gutierrez
Peter Baumann

School of Engineering and Science
Jacobs University Bremen gGmbH
Campus Ring 1
28759 Bremen
Germany
E-Mail: a.agarciagutierrez@jacobs-university.de, p.baumann@jacobs-university.de
`http://www.jacobs-university.de/`

Summary

Raster image data is the most voluminous data type encountered in remote sensing applications. With its multidimensional, gridded nature, the raster data structure is found to be very similar to the structures treated in OLAP (Online Analytical Processing); further, striking similarities can be observed in the mathematical modeling of operations in both domains. However, while in both remote sensing / imaging and OLAP there exists a host of research on analysis and extraction techniques, there is no cross-fertilization to the best of our knowledge.

One particular technique known in OLAP for speeding up complex database queries on very large datacubes is pre-aggregation, i.e., materialization of earlier results for subsequent use in same or similar requests. We feel that pre-aggregation can be advantageously transposed into the remote sensing / imaging area to speed up raster aggregation operations such as image scaling.

In this report we present fundamental operations on raster image data, model them using an algebraic framework, and classify them based on their structural properties. We then identify a set of operations requiring aggregation (summarization) of data. These operations are the potential candidates for support by OLAP pre-aggregation algorithms, which is our next research step.

Contents

1 Introduction

Satellites designed to document our planet over time provide valuable information about the impact of human activity, natural disasters and climate change. Typically, remote sensing applications form massive amounts of raster image data[1]. For example, the Landsat satellite program has accumulated over 1.7 million scenes and over 630 terabytes of data, at a growing rate of 320 gigabytes every day. Characteristic for the database technology supporting such data is the capability of dealing with huge amounts of multidimensional datasets. The raster data model maps an abstraction of the real world in a n-dimensional array of cells. The n-array is organized by continuous, evenly spaced rows and columns. Each cell is coded with an attribute that represents a data parameter e.g., temperature, height [1]. Similar data structures are found in other fields, particularly in business applications. For example, the two largest databases of Amazon combine together around of 42 terabytes of multidimensional data. Several approaches for intelligent analysis of the data are available, tried and tested. Online Analytical Processing (OLAP) and Data Mining represent two of the most important approaches.

As data sets grow larger, queries might need to operate over large amounts of data. Therefore, it is more efficient to extract general characterizations, that is, aggregate information of large subsets of the data. A typical solution is to precompute (materialize) the whole or parts of each query and save the results in a *materialized query table*. By querying the materialized query table instead of computing the results from the underlying base tables a database system can perform query processing more efficiently. However, full materialization may lead to an explosive growth of both storage and computation costs. To overcome this issue, substantial research as been conducted on the selection of views under different constraints [2, 3, 4, 5, 6]. But the problem of *view selection* is tightly correlated with another aspect, *query rewriting*, where we have also observed strong interest [7, 8, 9, 10, 11]. Query rewriting consists in reformulating the query in terms of the aggregates and then evaluating the rewriting over the aggregate extensions. The decision for answering a query from base or materialized tables lies on the estimated cost of answering the original and rewritten queries.

On the other hand, current research on raster databases encompass several aspects. In the area of *data modeling*, the focus is set on the construction of data models for spatial data changing over time, e.g., [12, 13]. Research is also conducted on the development of algebraic frameworks to improve spatial analysis capabilities, e.g., [14, 15, 16]. Another critical aspect is *query processing*, where research has focus on the use of parallel processing and indexing mechanisms to speed up access time of data stored in tertiary systems [17]. Some works have applied OLAP technology to support spatial data in GIS databases [18, 19], but they have focus on a particular data structure: vector data and to the best

[1]The terms *field* and *continuous data* are also used for some authors. In this report raster data refers to raster image data.

of our knowledge no works have addressed the issue of applying materialization techniques for aggregate queries with raster image data.

We consider the problem of selecting aggregates to minimize query response time for aggregate queries and the implementation of rewritings in raster image databases. In solving this problem, the first issue we need to address is what types of aggregates should be considered. Although there are several surveys describing GIS operations, most of them are introductory or lack mathematical foundations. In addition, extant classification schemes for GIS operations are not comprehensive: they either focus on a particular data structure or otherwise serve to a specific application. In this report we present a set of fundamental operations with geo-raster data and model their computation using an algebraic framework. The selection of these operations is derived from a thorough review of GIS classification schemes [1, 20, 21, 22, 23, 24, 25, 26, 27], and of international standards/best practices like *Open GIS Consortium*(OGC) [28]. We consider that such algebraic characterization is crucial for the identification of aggregates to be materialized and the application of rewriting algorithms.

The report is organized as follows. Section 2 motivates the choice of methodology and introduces the *Array Algebra* framework subsequently used. Section 3 describes the modeling of fundamental geo-raster operations. In Section 4 we identify potential operations to be treated with pre-aggregation algorithms. Finally, in Section 5 we present a brief summary and future work.

2 Choice of Methodology and Related Work

2.1 Geo-Raster Operations

In [23] a set of GIS operations is presented. By analyzing current GIS user interfaces and omitting all those operations that are due to either the development of a particular software package or are a result of the data model employed, the author derived a list of 20 universal GIS operations. A classification of GIS functions for natural resource management is presented in [29]. Such classification was adapted from [25], making clear distinction between vector and raster operations. In [24], raster operations are classified into a series of recognized domains that are common to many GIS. The description of raster operations in all these classification schemes is purely conceptual without any mathematical foundations. A more comprehensive description of raster operations is presented in [20], however, the computation of the examples does not follow an uniform framework.

Our classification scheme is derived from the above work, and is complemented by modeling the operations using an uniform algebraic framework.

2.2 Algebraic Frameworks

For a rigid comparison and classification, raster operations have to be described uniformly and by means of a sound mathematical framework. An additional requirement arising from our research context is that the model should embed itself naturally in a database environment, allowing not only to derive query languages, but also to describe query optimization and efficient storage management.

Most theoretical work on array operations for databases has been accomplished in the field of OLAP. However, all these mechanisms have never been applied to the geo domain, and hence do not provide an appropriate basis for our geo raster modeling. Some work exists, though, to model raster image operations on an abstract, yet fine-grain level. We concentrate on models that allow multidimensional arrays, as opposed to 2-D only.

Mennis et al [30] introduce 3-D map algebra, which extends classical map algebra. The term "map algebra" was first used in the late 1970s [31] and has since been used in loose reference to a set of conventions, capabilities, and analytical techniques that have been widely adopted for raster-based geographic information systems (GIS) [32]. Map algebra has been incorporated in many GIS and remote sensing image processing packages. A drawback is that 4-D imagery and beyond, something definitely within the scope of our research, is not supported to the best of our knowledge. Further, map algebra is not aligned with the needs of databases, such as a framework for operator optimization.

In [33] nested relational calculus is extended with multidimensional arrays, obtaining a model called NCRA. Although NCRA has been prototypically implemented using a derivative, AQL, it is not easy to see how real-life geo raster operations can be phrased adequately.

The rasdaman Array Algebra [34, 35] has been developed to obtain a comprehensive formal framework for studying query language, query optimization, and storage management in array database systems. Array Algebra is inspired by AFATL Image Algebra [36], a very general, rigorously formalized framework capable of describing a large class of imaging operations. As [35] outlines, Array Algebra is able to express both imaging and OLAP operations. Further, the rasdaman system which implements most of Array Algebra has proven feasibility in many practical applications. Particularly attractive from a modeling viewpoint is that Array Algebra relies on only three basic operators, which greatly supports the operation morphology classification task.

An algebra-based Array Manipulation Language, AML, is introduced in [37]. Two operators serve to subsample and interleave, resp., arrays based on bit sequences governing cell selection. The third operator, APPLY, corresponds to induce operations modulo the bit pattern for cell selection. Bit patterns are modeled in Array Algebra through 1-D bit arrays executing the same control function. AML is more restricted than Array Algebra in that such control arrays cannot contain arbitrary values (e.g., weights), and moreover are constrained to 1-D. Further, aggregations cannot be modeled as comprehensively as in Array Algebra.

Finally, some important operations are modeled as black boxes, which does not support very well classification work.

In summary, most array frameworks nowadays are geared towards OLAP tasks, without regarding spatio-temporal array application fields. Conversely, frameworks such as AML aiming at imaging do not consider OLAP. Sometimes array iteration or aggregation retracts to user-defined functions which make the algorithms unavailable to our classification. All operations such as aggregation and spatial join found in these approaches are expressible in Array Algebra, too, except that dimension hierarchies usually are supported by convenient mechanisms in OLAP, a feature not foreseen (yet expressible) in core Array Algebra.

As additionally the rasdaman vehicle will form the implementation platform for our subsequent research work, Array Algebra is a natural candidate for the classification work described in this report. In the next Section we give a brief introduction to Array Algebra.

3 Array Algebra

The formal framework for describing arrays and their operations, which is presented in the sequel, is called Array Algebra. Array Algebra is a domain-independent algebra for an abstract treatment of operations on multi-dimensional arrays. Following Furtado [38] we call n-dimensional arrays *multi-dimensional discrete data* (MDD).

Essentially, the algebra consists of only three operations: an array constructor, a generalized aggregation, and a multidimensional sorter. This core model does not rely on recursion and is safe in evaluation, yet it allows to express a wide range of imaging, statistical, and OLAP operations. Tentatively no recursion has been included, and any array expression can be computed in finite time if each of the base operations does so. On the other hand, Array Algebra is a complete framework in that it does not refer to "black box" user-defined functions external to the apparatus; all array-related functionality is resolved by the framework, relying solely on the operations coming with the cell type's algebraic structure. Array Algebra has been presented in [34] and, more completely, in [35].

Array Algebra forms the conceptual basis of a domain-independent array DBMS, rasdaman [39], which offers an SQL-based query language with extensive algebraic query and storage optimization. The rasdaman system is in international operational use in public administration, industry, and research.

3.1 Overview

Arrays are represented in Array Algebra as functions mapping n-dimensional points (i.e., vectors) from discrete Euclidean space to values. This is common in imaging for a long time - see, e.g., [40] - and has been transposed to database

terminology in [41, 34]. To smoothly embed Array Algebra into relational algebra we use a set-oriented basis, i.e., we allow sets over arrays and other data. Operations on such arrays have to be second order to describe functions which simultaneously apply some other function in a cell-wise manner. In practice, second order functionals are needed, on the one hand, to allow for binding variables to points for iterating coordinate sets and, on the other hand, to aggregate arrays (or part thereof) into scalar values. The latter operation corresponds very much to relational set aggregators; however, instead of providing a limited list of aggregation operations as in the relational algebra, a general constructor is introduced by Array Algebra which is parametrized with the underlying base operation.
Array Algebra is minimal in the sense that no subset of its operations exhibits the same expressive power. It is also closed in application: any expression is either of a scalar or an array type. Finally, Array Algebra does not rely on any external array handling functionality ("user-defined functions") aside of the operations coming with the algebraic structure of the cell type. We observe that in many cases operations can be formulated without explicitly referring to the array dimension, allowing to develop parametrized cross-dimensional, domain-independent query libraries which go well beyond the capabilities of object-relational ADTs [42].

3.2 N-Dimensional Interval Arithmetics

We first introduce some notation for n-dimensional integer interval arithmetics. We call the coordinate set of an array its *spatial domain*. Informally, a spatial domain is defined as a set of n-dimensional points (i.e., algebraic vectors) in Euclidean space forming a finite hypercube with boundaries parallel to the coordinate system axes.
We assume common vector notation. For a natural number $d > 0$, we write $\underline{x} = (x_1, ..., x_d) \in X \subseteq Z^d$ for some d-dimensional vector $\underline{x}$, $\underline{x} + \underline{y}$ for vector addition, etc. The point set forming the geometric extent of an array is called its *spatial domain*. A spatial domain X of dimension d spanned by $\underline{l}$ and $\underline{h}$ is defined as:

$$\begin{aligned} X &= [l_1 : h_1, ..., l_d : h_d] \\ &:= \bigtimes_{i=1}^{d} \{x_i : l_i \leq x_i \leq h_i\} \quad if \quad \forall 1 \leq i \leq d : l_i \leq h_i \\ &:= \{\} \qquad\qquad\qquad\qquad otherwise. \end{aligned}$$

Functions *low, high* $P(\mathbf{Z}^d) \rightarrow \mathbf{Z}^d$ defined as *low(X)=*$\underline{l}$ and *high(X)=*$\underline{h}$ for some spatial domain X given as before denote the bounding vectors. We will abbreviate $low_i(X) = l_i$ and $high_i(X) = h_i$ for the i^{th} component. Function *dim(X)=*d denotes the dimension of spatial domain X.
On such hypercubes, point set operations can be defined in a straightforward

way. We admit only those operations which respect closure, such as *intersect* and *union**, whereby the asterisk "*" denotes the hull operation applied to the result:

$$intersect^*(X,Y) :=$$
$$[max(low_1(X), low_1(Y)) : min(hi_1(X), hi_1(Y)), ...,$$
$$max(low_d(X), low_d(Y)) : min(hi_d(X), hi_d(Y))]$$

$$union^*(X,Y) :=$$
$$[min(low_1(X), low_1(Y)) : max(hi_1(X), hi_1(Y)), ...,$$
$$min(low_d(X), low_d(Y)) : max(hi_d(X), hi_d(Y))]$$

As can be shown easily, these operations are commutative, associative, and distributive. The shift operator allows to change a spatial domain's position according to a translation vector $\underline{t}$:

$$shift_t(X) := \underline{x} + \underline{t} : \underline{x} \in X$$

Let X be spanned by d-dimensional vectors l and h. For some integer i with $1 \leq i \leq d$ and a one-D integer interval I=[m:n] with $l_i \leq m \leq n \leq h_i$, the *trim of X to I in dimension d* is defined as

$$\tau_{i,I}(X) := \{\underline{x} \in X : m \leq x_i \leq n\} = [l_I : h_I, ..., m : n, ..., l_d : h_d]$$

Intuitively speaking, trimming slices off those parts of an array which are lower than m and higher than n in the dimension indicated; the dimension is unchanged. As opposed to this, a section cuts out a hyperplane with dimension reduced by 1. Formally, for some X as above, an integer p with $1 \leq p \leq d$, the *section of X at position p in dimension i* is given by:

$$\sigma_{i,p}(X) := \{\underline{x} \in Z^{d-1} : \underline{x} = (x_1, ..., x_{i-1}, x_{i+1}, ..., x_d), (x_1, ..., x_{i-1}, p, x_{i+1}, ..., x_d) \in X\}$$
$$= [l_1 : h_1, ..., l_{i-1} : h_{i-1}, l_{i+1} : h_{i+1}, ..., l_d : h_d]$$

Trimming is commutative and associative, whereas a section changes dimension numbering and, therefore, has neither of these properties.

3.3 The Core Algebra

Let $X \subseteq \mathbf{Z}^d$ be a spatial domain and F a value set (i.e., a homogeneous algebra). Then, an *F-valued d-dimensional array over spatial domain X* - or short: *(multidimensional) array* - is defined as

$$a : X \to F (i.e., a \in F^X),$$
$$a = \{(\underline{x}, a(x)) : \underline{x} \in X, a(\underline{x}) \in F\}$$

The array elements a($\underline{x}$) are referred to as *cells*. For notational convenience, we also allow to enumerate the components of a cell coordinate vector, e.g., $a(x_1, x_2, x_3)$. Auxiliary function *sdom(a)* denotes the spatial domain of some array a; further, we lift function *dim* to arrays. Let $a : X \rightarrow F$ be an array, then *sdom* and *dim* are defined as:

$$sdom(a) := X$$
$$dim(a) := dim(sdom(a))$$

The i^{th} dimension range of an array's spatial domain we will denote by $sdom_i(a)$.

Example: For a 1024×768 image a with lower left corner in the origin of the coordinate system, *sdom(a)*=[0:1023,0:767], *dim(a)*=2.

The first functional we introduce is the *array constructor* MARRAY. It allows to define arrays by indicating a spatial domain and an expression which is evaluated for each cell position of the spatial domain. An iteration variable bound to a spatial domain is available in the cell expression so that a cell's value can depend on its position. Let X be a spatial domain, F a value set, and v a free identifier. Let further e_v be an expression with result type F containing zero or more free occurrences of v as placeholder(s) for an expression with result type X. Then, an *array over spatial domain X with base type F* is constructed through:

$$MARRAY_{X,v} \quad (e_v) = \{(\underline{x}, a(\underline{x})) : a(\underline{x}) = e_x, \underline{x} \in X\}$$

Example: Consider scaling of a grey scale image a with *sdom(a)* = [1:m,1:n] by a factor $s \in \mathbf{R}$. We assume component-wise scalar division and rounding on vectors and write:

$$MARRAY_{[1:m*s,1:n*s],v}(a(round(v/s)))$$

For $0 < s < 1$ the image is sized down; the interpolation method then corresponds to "nearest neighbor", the simplest interpolation technique used in imaging.

The operation which in some sense is the dual to the *MARRAY* constructor is the *condenser*, *COND*. It takes the values of an array's cells and combines them through the operation provided, thereby obtaining a scalar value. Again, an iterator variable is bound to a spatial domain to address cell values in the condensing expression. Let o be a commutative and associative operation with signature $o : F, F \rightarrow F$, let further v be a free identifier, $X = \underline{x_1}, ..., \underline{x_n} | \underline{x_i} \in \mathbf{Z}^d$ a spatial domain consisting of n points, and $e_{a,v}$ an expression of result type F containing occurrences of an array a and identifier v. Then, the *condense of a by o* is defined as:

$$COND_{o,X,v}(e_{a,v}) := \underset{\mathrm{x}\in\mathrm{X}}{O} e_{a,\underline{x}} = e_{a,\underline{x_1}} o ... o e_{a,\underline{x_n}}$$

Examples: Let a be the image as defined in the above example. The sum over all pixel intensities in a is given by:

$$COND_{+,sdom(a),v}(a(v)) = \sum_{x\in[1:m,1:n]} a[x]/(m*n)$$

We abbreviate the above condense expression as $avg_cells(a)$. Obviously the aggregate operations known from the SQL database language can be introduced naturally, such as maximum and average:

$$max\_cells(a) := COND_{max,sdom(a),v}(a(v))$$

$$avg\_cells(a) := COND_{+,sdom(a),v}(a(v))/|sdom(a)| = \sum_{x\in[1:m,1:n]} a[x]/(m*n)$$

For color table computation needed, e.g., for generation of a GIF image encoding, one has to know the set of all values occurring in array a. The condenser allows to derive this set by performing the union of all cell values:

$$COND_{\cup,sdom(a),v}(\{a(v)\})$$

The third and last operator is an array sorter which proceeds along a selected dimension to reorder the corresponding hyperslices. Function $sort_s$ rearranges a given array along a specified dimension s without changing its value set or spatial domain. To this end, an order-generating function is provided which associates a "sequence position" to each (d-1)-dimensional hyperslice. Let a be a d-dimensional array, $i \in N$ with $1 \leq i \leq d$ a dimension number, and $f_{s,a} : sdom_s(a) \rightarrow N$ a total function which, for a given array a, inspects a in the sorting dimension s and delivers an ordering measure for each hyperslice. Further, let $perm(x, y)$ be a predicate indicating that vector $\underline{x}$ is a permutation of vector $\underline{y}$ (and vice versa). Then, the two sorters $sort^{asc}_{s,f}$ and $sort^{desc}_{s,f}$ for ascending and descending order, resp., are given through:

$$\begin{aligned}
sort^{asc}_{s,f}(a) := \{&(\underline{(y)}, b(\underline{(y)}) : \underline{(y)} \in sdom_s(a),\\
&\forall p, q \in sdom_s(a) : p < q \implies f_{s,b}(p) \leq f_{s,b}(q),\\
&perm(((b(x_1, ..., x_{s-1}, sdom_s(a).lo, x_{s+1}, .., x_d), ...,\\
&b(x_1, ..., x_{s-1}, sdom_s(a).hi, x_{s+1}, .., x_d)),\\
&((a(x_1, ..., x_{s-1}, sdom_s(a).lo, x_{s+1}, .., x_d), ...,\\
&((a(x_1, ..., x_{s-1}, sdom_s(a).hi, x_{s+1}, .., x_d)))
\end{aligned}$$

$$
\begin{aligned}
sort^{desc}_{s,f}(a) := \{ & (\underline{(y)}, b(\underline{(y)}) : \underline{(y)} \in sdom_s(a), \\
& \forall p, q \in sdom_s(a) : p < q \implies f_{s,b}(p) \geq f_{s,b}(q), \\
& perm(((b(x_1, ..., x_{s-1}, sdom_s(a).lo, x_{s+1}, .., x_d), ..., \\
& b(x_1, ..., x_{s-1}, sdom_s(a).hi, x_{s+1}, .., x_d)), \\
& ((a(x_1, ..., x_{s-1}, sdom_s(a).lo, x_{s+1}, .., x_d), ..., \\
& ((a(x_1, ..., x_{s-1}, sdom_s(a).hi, x_{s+1}, .., x_d)))
\end{aligned}
$$

The resulting array has the same number of dimensions, spatial domain, and base type as the input array. Note that function $f_{s,a}$ has all degrees of freedom to assess any of a's cell values for determining the measure value of a hyperslice on hand - it can be a particular cell value in the current hyperslice, the average of all hyperslice values, or even neighbored slices (e.g., for relative increases of sales values.

Example: Let a be a 1-D array with spatial domain D=[1:d] where cell values denote sales figures over time. Let further sorting function $f_{s,a}$ be given as $f_{s,a}(p) = a[p]$. Then, the following expression delivers the ranked sales:

$$sort_{0,f^{desc}}(a)$$

As an aside we note that the sort operator includes the relational group by. We omit further details, referring the interested reader to [35].

As we will demonstrate below, slice and roll-up operations arising from array access based on dimension hierarchies can be expressed, although - not very comfortably - by indicating the cell coordinates pertaining to a particular member set. Concepts for an intuitive, symbolic treatment of dimension hierarchies are currently under investigation.

3.4 Derived Operators

Several useful operations can be derived from the above ones. We present a selection of those which have turned out particularly important in practical applications.

3.4.1 Trimming and Section

The previously introduced spatial domain operations trimming and section give rise to corresponding array operations. For some array a, an 1-D interval I, and two natural numbers $1 \leq t \leq dim(a)$ and $p \in sdom_d(a)$ they are defined as:

$$
\begin{aligned}
TRIM_{t,I}(a) &:= MARRAY_{X,v}(a(v)), forX = \tau_{t,I}(sdom(a)) and d < dim(a) \\
SECT_{t,p}(a) &:= MARRAY_{X,v}(a(v)), forX = \sigma_{t,p}(sdom(a)) and d < dim(a)
\end{aligned}
$$

3.4.2 Induced Operations

A basic set of operations are those induced by the algebra of the underlying value sets. If a,$b \in F^X$ are arrays and o is a binary operation on F, then o induces a binary operation on F^X denoted by o_{ind} such that, if $c = ao_{ind}b$, then $c \in F^X$ and, for all $x \in X$, $c(x) = a(x)ob(x)$. Along this line, we also allow to induce unary operations. Notably, these operations are not axiomatic:

$$IND_f(a) = MARRAY_{X,v}(f(a(v))) \quad for\ X = sdom(a)$$
$$IND_g(a,b) = MARRAY_{X,v}(g(a(v),b(v))) \quad for\ X = sdom(a) = sdom(b)$$

Algebraic properties of F transform to corresponding structures on the set F^X of induced functions. If F is a field, then F^X is a vector space; for a ring F, F^X is a module for suitably defined spatial domains.
Examples: Let a be a grayscale image over spatial domain X. Increasing intensity by 5 can be accomplished through induction on unary "+5":

$$IND_{+5}(a) = \{(\underline{x}, b(\underline{x})) : b(\underline{x}) = a(\underline{x}) + 5, \underline{x} \in X\}$$

Consider now another grayscale image b over the same spatial domain X. Then, pixel addition can be induced to obtain image addition:

$$IND_{+}(a,b) = \{(\underline{x}, c(\underline{x})) : c(\underline{x}) = a(\underline{x}) + b(\underline{x}), \underline{x} \in X\}$$

When used in a query, binary induction obviously implies a spatial join.
Let now c be a color image where the cell type is a three-integer record of red, green, and blue intensity, resp. Such a *pixel-interleaved* image is transformed into a *channel-interleaved* representation, i.e., three separate color planes, through induction on the record access operator ".", obtaining:
$< c.red, c.green, c.blue >$
The above type of induction is also referred to as point wise induction, as points pairwise match for each application of the base function.

4 Modeling Operations

In this Section we present the modeling of fundamental *geo-raster operations* based on the algebraic framework Array Algebra. Several example queries illustrate the use of the operations under discussion. Note that we are using the terms *array*, *multidimensional data structure* (MDD) and *raster image* interchangeably.

4.1 Mathematical operators

The following groups of mathematical operators are distinguished: *arithmetic, trigonometric, boolean* and *relational*. They operate at cell level and can be applied in a single or multiple MDDs of numerical type and identical spatial domain. The

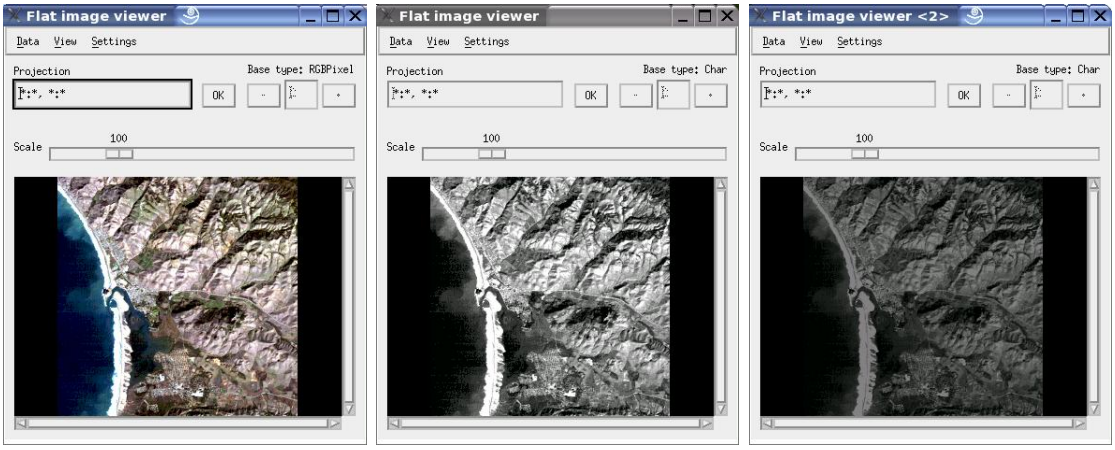
basic arithmetic operators include *addition (+)*, *subtraction (-)*, *multiplication (*)*, and *division (/)*. *Trigonometric* functions perform trigonometric calculations on the values in an input raster: *sqrt*, *log*, *ln*, *absolute*, *sine (sin)*, *cosine (cos)*, *tangent (tan)* or their inverse *(arcsin, arccos, arctan)*. Consider, for example, the following query using an arithmetic function:

Query 4.1. *Extract the green component of a RGB raster image A and decrease the contrast by a factor of 2.*

With array algebra, the query can be computed as follows:

$$MARRAY_{sdom(A),i}(A.green[i]/2)$$

Results are shown in Fig. 1.



(a) Original RGB image (b) Green component (c) Output raster

Figure 1: Usage of *arithmetic* operators for decreasing the contrast by a factor of 2 in the green channel of the original RGB image.

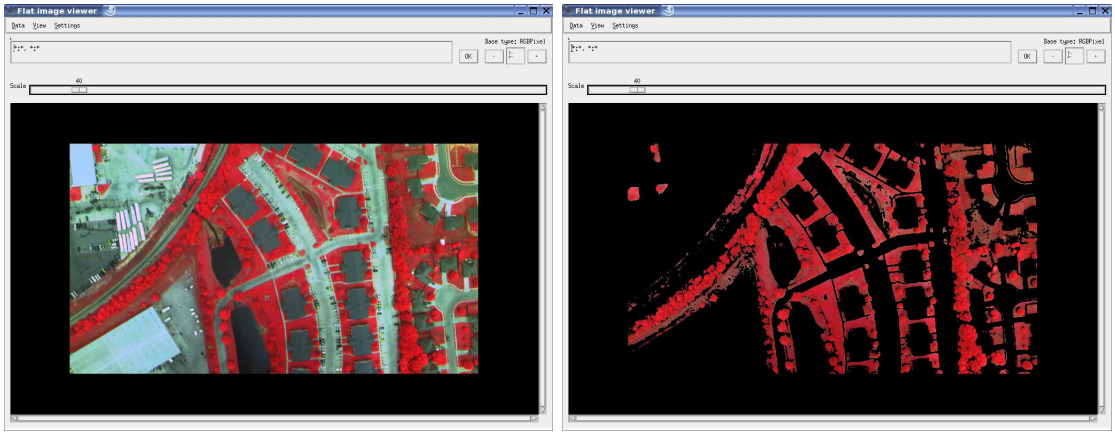
A raster image, or part of it, can be manipulated using the standard rules of Boolean algebra which are integrated in database languages such or derived from SQL [20]. Boolean algebra uses logical operators such as *and*, *or*, *not*, and *xor* to determine whether a particular condition is true or false. These operators are often combined with *relational* operators: *equal* ($=$), *not equal* ($\neq$), *less than* ($<$), *less or equal than* ($\leq$), *greater than* ($>$), and *greater or equal than* ($\geq$). Consider, for example, the following queries:

Query 4.2. *Given a NRG raster image A, highlight the cells with "sufficient near-infrared" values.*

This query can be answered by imposing a lower bound on the *infrared* intensity, and upper bounds in the *green* and *blue* intensities. The resulting boolean array is multiplied by the original image A to show the original cell where an infrared value prevails and black otherwise.

$$MARRAY_{sdom(A),i}\ (A[i] * ((A[i].nir \geq 130)\ and \\ (A[i].green \leq 110)\ and\ (A[i].blue \leq 140)))$$

Results are shown in Fig. 2.



(a) Original nrg raster (b) Output raster

Figure 2: Usage of *relational* operators in highlighting infrared areas of an NRG image.

Query 4.3. *Compare the cell values of two 8-bit gray raster images A and B. Create a new MDD where each cell value takes the value of 255 (white pixel) when the cell values of A and B are identical.*

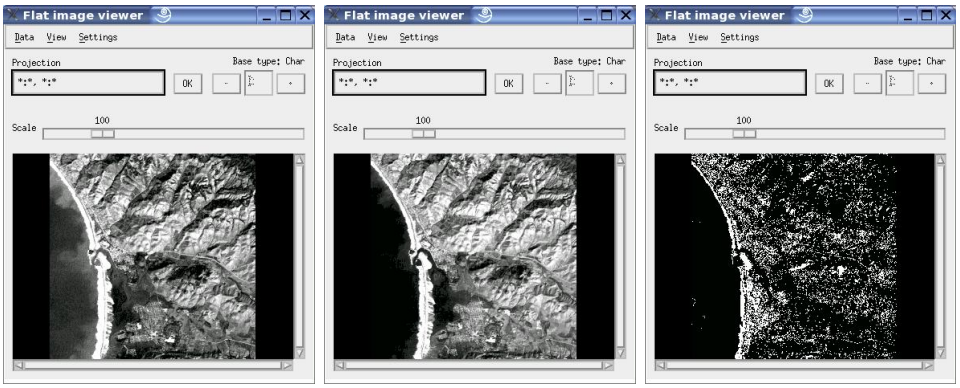
The algebraic formulation is as follows:

$$MARRAY_{sdom(A),i}((A[i] = B[i]) * 255)$$

Results are shown in Fig. 3.

4.1.1 Reclassification

Reclassification is a generalization technique used to re-assign cell values in classified MDDs. For illustration consider the query below where reclassification is based on a land suitability study. Typically, the classifications are based on international standards like the one proposed by the United Nations (UNO) and the Food and Agriculture (FAO) Organizations (Table 1).



(a) Grey 8-bit raster A (b) Grey 8-bit raster B (c) Output image.

Figure 3: Usage of *relational* and *arithmetic* operators in identifying areas where cell values of A and B are equal.

Table 1: UNO and FAO suitability classifications [2]

Classification	Description
S1	Highly suitable
S2	Moderately suitable
S3	Marginally suitable
NS	Not suitable

Query 4.4. *Given a 8-bit gray image A, map each cell value with its corresponding suitability class shown in Table 2, and decrease the contrast of the image according to the decreasing factor.*

The query can be answered as follows:

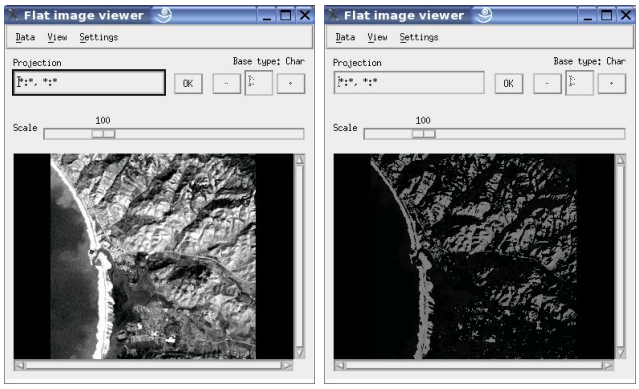
$$\begin{aligned} MARRAY_{sdom(A),g}(&(((A[g] > 180) * A[g]/2) + \\ &(((A[g] \geq 130) and (A[g] < 180)) * A[g]/3) + \\ &(((A[g] \geq 80) and (A[g] < 130)) * A[g]/4) + \\ &((A[g] < 80) * A[g]/5)) \end{aligned}$$

Results are shown in Fig. 4.

[2]Classification taken from http://www.fao.org/docrep/X5310E/X5310E00.htm

Table 2: Capability indexes for the different capability classes

Capability index	Class	Suitability class	decrease factor
>180	I	S1	2
130-180	II	S2	3
80-130	III	S3	4
< 80	IV	NS	5



(a) Original raster (b) Output raster

Figure 4: Use of *relational* and *arithmetic* operations in *reclassification*.

4.1.2 Proximity

The *proximity* operation creates a new MDD where each cell value contains the distance to a specified *reference point*. As an example consider the following query:

Query 4.5. *Estimate the proximity of each cell of the raster image shown in Fig. 4(a) to the reference cell located in [30,5].*

This query can be formulated as:

$$MARRAY_{sdom(A),(g,h)}(|g-30|+|h-5|)$$

Results are shown in Fig. 5.

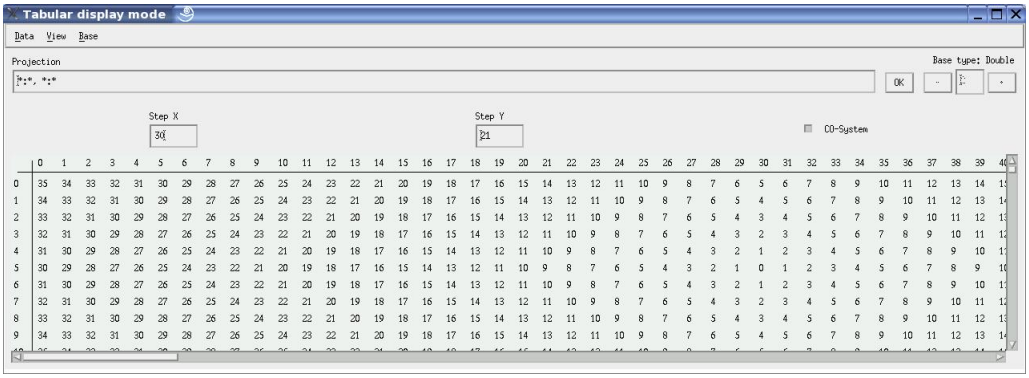


Figure 5: Usage of *relational* and *arithmetic* operators in computing a *proximity* operation.

4.1.3 Overlay

The *overlay* operation refers to the process of stacking two or more identical geo-referenced MDDs on top of each other so that each position in the area covered can be analyzed in terms of these data. The overlay operation can be solved by using arithmetic and logical operators. For example, consider the following query:

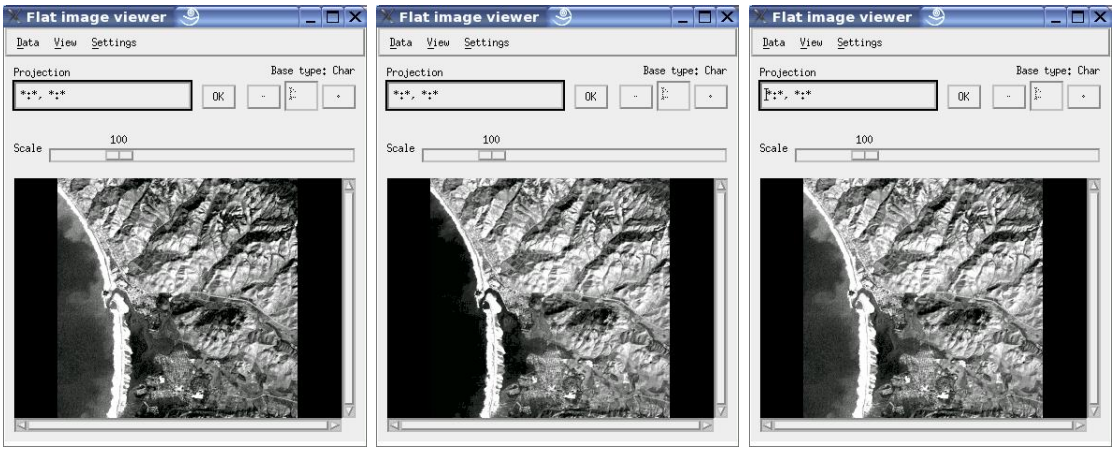
Query 4.6. *Given two 8-bit gray raster images A and B with identical spatial domain perform an overlay operation. For each cell value of the new array select the maximum cell value between A and B.*

The computation of this query can be formulated as:

$$MARRAY_{sdom(A),g}(((A[g]>B[g])*A[g])+((A[g]\leq B[g])*B[g]))$$

Results are shown in Fig. 6.

As an alternative approach, a different condition can be tested to compute the *overlay* operation:



(a) 8-bit gray raster A (b) 8-bit gray raster B (c) Output raster

Figure 6: Usage of *relational* and *arithmetic* operators in computing an *overlay* operation according to Query. 4.6.

Query 4.7. *Perform an overlay operation between the MDDs A and B. Wherever the cell value of B is non-zero, the result cell value in the new array will be this value. Otherwise, the cell value of A must be taken.*

The query can be answered as follows:

$$MARRAY_{sdom(A),g}(((B[g] > 0) * B[g]) + ((B[g] \leq 0) * A[g]))$$

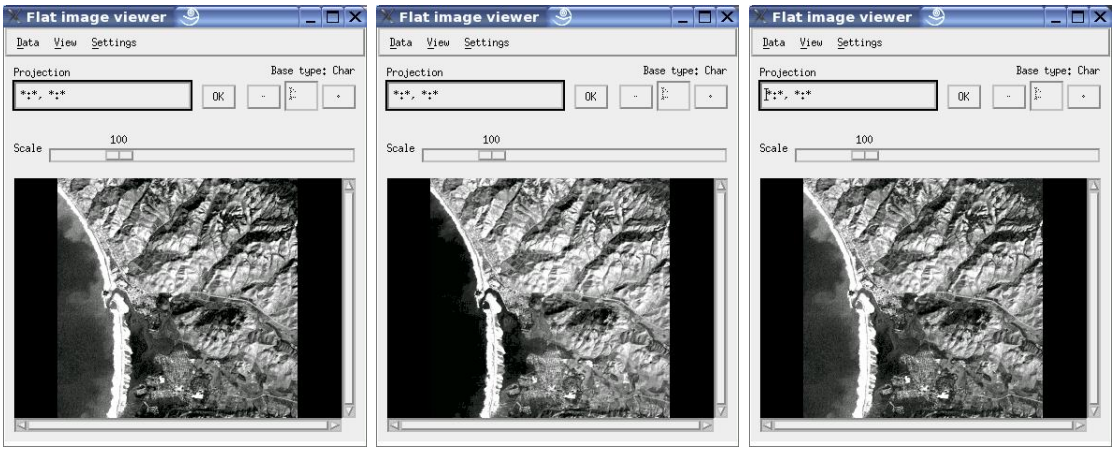
Results are shown in Fig. 7.

4.2 Aggregate operations

An *aggregate* function takes a collection of cells and returns a single value that summarizes the information contained in the set of cells. The SQL standard provides a variety of aggregate functions. SQL-92 includes *count, sum, average, min,* and *max*. SQL-1999 adds *every, some* and *any*. The SQL-OLAP addendum to the SQL-1999 standard includes 18 additional aggregates. The remaining of this Section discusses the usage of aggregate functions in the processing of geo-raster operations.

4.2.1 Add

The *add* operation can be applied in entire rasters (or part of the rasters). The operation sums up the content of the cells and returns the total as a scalar value. It can be applied in two or more rasters with identical spatial domain returning a



(a) Grey 8-bit raster A (b) Grey 8-bit raster B (c) Output raster

Figure 7: Usage of *relational* and *arithmetic* operators in computing an *overlay* operation according to Query. 4.7.

new raster of same spatial domain. In this case, the cells of the new raster contain the sum of the inputs computed on a cell-by-cell basis. As an example of the add operation in a single raster consider the following query:

Query 4.8. *Return the sum of all cell values of the MDD shown in Fig. 8(a).*

$$add\_cells(A) = COND_{+,sdom(A),i}(A[i])$$

Results are shown in Fig. 8.

4.2.2 Count

The *count* operation returns the number of cells that fulfill a boolean condition applied in a raster. For example, consider the following query:

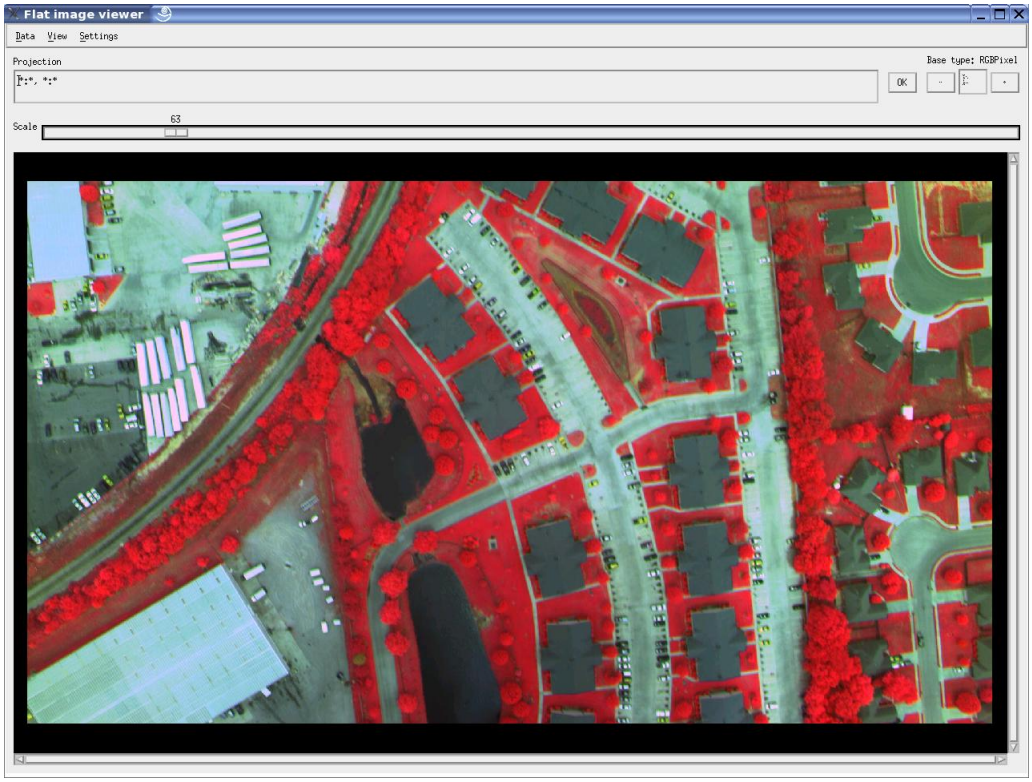
Query 4.9. *Return the number of cells of a MDD A of boolean type, containing true value in the green channel.*

$$count\_cells(A) = COND_{+,sdom(A),i}(A[i].green = 1)$$

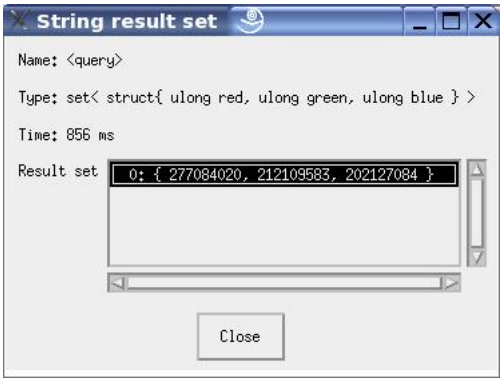
4.2.3 Average

The *average* function returns a scalar value representing the mean of all values contained in a raster. As an example consider the following query:

Query 4.10. *Return the average of the cell values in each channel of the NRG image shown in Fig. 8(a).*



(a) Original NRG raster



(b) Output raster

Figure 8: Usage of the *add* aggregate function in estimating the total sum of cell values in a MDD.

Let *sum_cells(A)* be a function calculated as shown in Section 4.2.1, and *card*(sdom(A)) a function returning the cardinality of A, i.e., the number of cells covered by A's spatial domain. Then, the *average* of A is calculated as follows:

$$avg\_cells(A) = \frac{sum\_cells(A)}{card(sdom(A))}$$

Results are shown in Fig. 9.

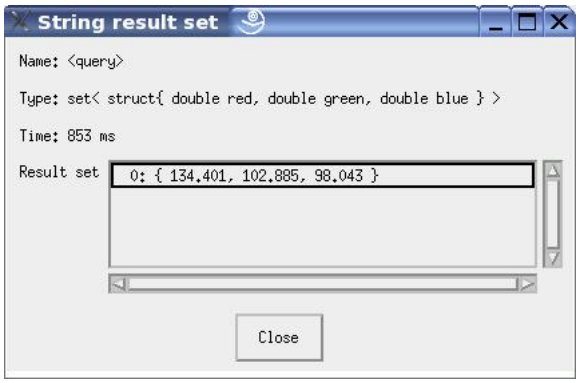


Figure 9: Usage of the *average* aggregate function

4.2.4 Maximum

The *maximum* operation returns the largest cell value contained in a raster of numerical type. As an example, consider the following query:

Query 4.11. *Return the maximum cell value of all cells contained in the NRG raster image shown in Fig. 8(a).*

$$max\_cells(A) = COND_{max,sdom(A),i}(A[i])$$

Results are shown in Fig. 10.

4.2.5 Minimum

The *minimum* operation returns the smallest cell value contained in a raster of numerical type. As an example, consider the following query:

Query 4.12. *Return the smallest element of all cell values in the NRG raster image shown in Fig. 8(a).*

$$min\_cells(A) = COND_{min,sdom(A),i}(A[i])$$

Results are shown in Fig. 11.

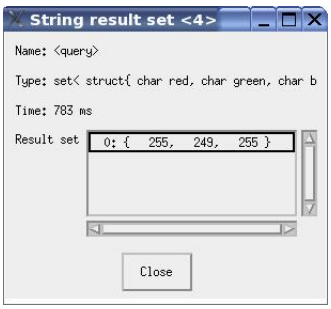


Figure 10: Usage of the *maximum* aggregate function

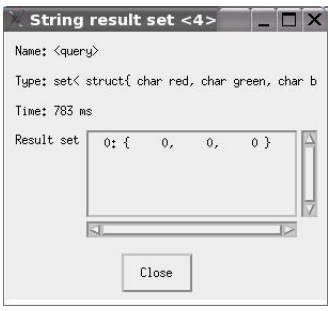


Figure 11: Usage of the *minimum* aggregate function

4.2.6 Histogram

A *histogram* provides information about the number of times a value occurs across a range of possible values. For a 8-bit MDD up to 255 different values are possible. As an example consider the following query:

Query 4.13. *Calculate the histogram for a two-dimensional MDD, 8-bit resolution pixel integer array A.*

The query can be computed as follows:

$$MARRAY_{sdom(A),g}(count\_cells(A = g[0])) \quad (1)$$

Results are shown in Fig. 12.

4.2.7 Diversity

The *diversity* operation returns the different classifications existing in a raster MDD. For example, consider the following query:

Query 4.14. *Given the classifications existing in a 8-bit gray raster image, return true (1) for those classes whose total number of cells are greater than 0.*

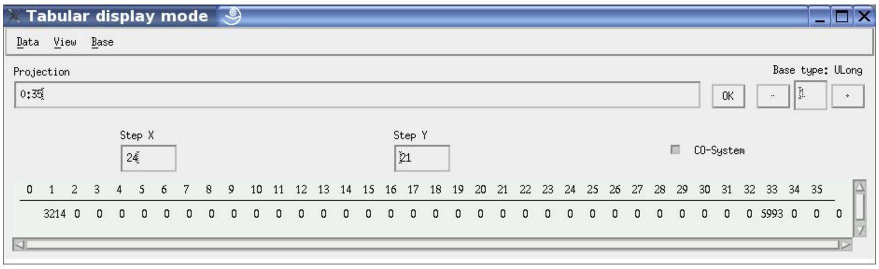


Figure 12: Usage of the *count* aggregate function for the computation of the *histogram.*

For the computation of this operation we can make use of the histogram calculated in Query. 4.2.6. Let B be a 1-D array containing the histogram values:

$$B = MARRAY_{sdom(A),g}(COND_{+,sdom(A),i}(A[i] = g))$$

then, C is the array containing *true* values for the elements of the histogram that are greater than 0:

$$C = MARRAY_{sdom(B),i}(B[i] > 0)$$

Results are shown in Fig. 13.

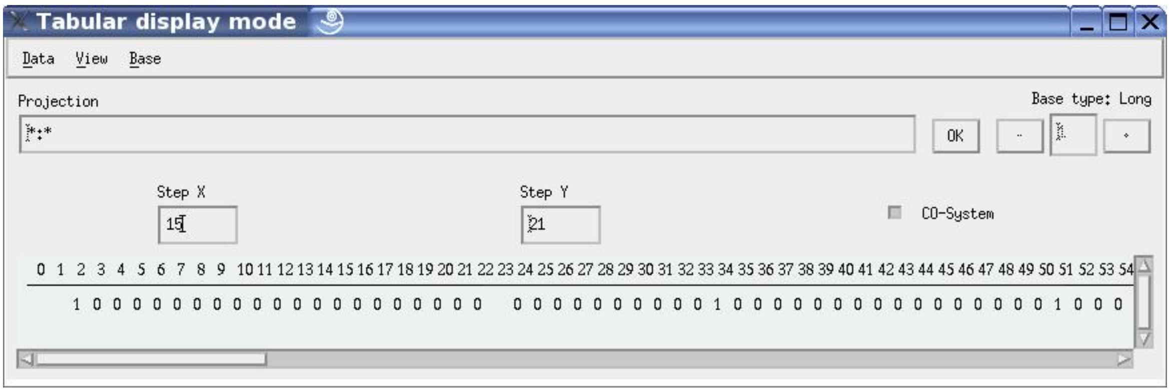


Figure 13: Usage of the *add* aggregate function for the computation of *diversity.*

4.2.8 Majority/Minority

In a classified MDD, the *majority* operation finds the class value with the larger number of elements in the MDD. Similarly, the *minority* operation finds the cell value with fewer number of elements. As an example, consider the following query:

Query 4.15. *Return the cell representing the majority of all cell values contained in a 2D 8-bit gray raster image A shown in Fig. 14(a).*

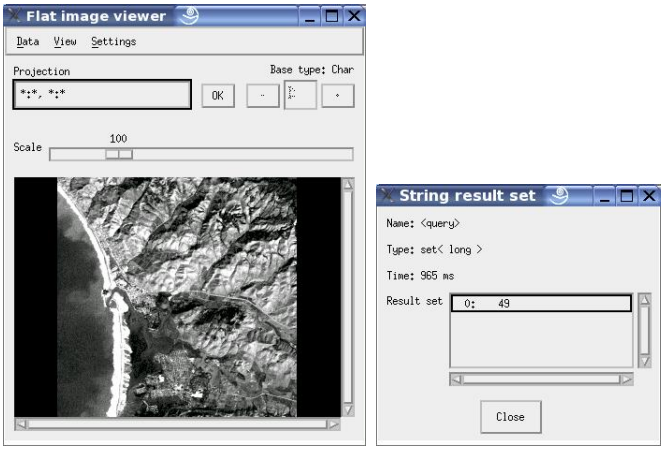
To solve this query we can use the histogram computed in Query. 4.2.6, and then select the cell value representing the majority of the different classes. Let h be a 1-D array containing the histogram values, *h1* a 1-D array of spatial domain[0:255] containing a list of values from 0 to 255. Let *h2* be an array containing the sum of h and h1:

$$h2 = MARRAY_{[0:255],g}(h + h1)$$

then, *majority* can be computed as follows:

$$COND_{+,sdom(A),i}((max\_cells(h) = (h2[i] - h1[i])) * h1[i])$$

Results are shown in Fig. 14.



(a) Classified raster (b) Majority class

Figure 14: Usage of *maximum* aggregate function in computing a *majority* operation.

4.3 Statistical operations

The basic *statistical* functions include *standard deviation*, *root square*, *power*, *mode*, *median*, *variance*, and *top-k*. These functions can be applied to a raster, or set of rasters that are retrieved by a logical search. Consider the following examples:

4.3.1 Variance

Let n be the cardinality of A, n = *card(sdom(A))*; and *avg* a variable containing the average of all the cell values of A, *avg=avg_cells(A)*; then the variance v of A can be solved as follows:

$$v(A) = \frac{1}{n} * COND_{+,sdom(A),i}((A[i] - avg) * (A[i] - avg))$$

Results are shown in Fig. 15.

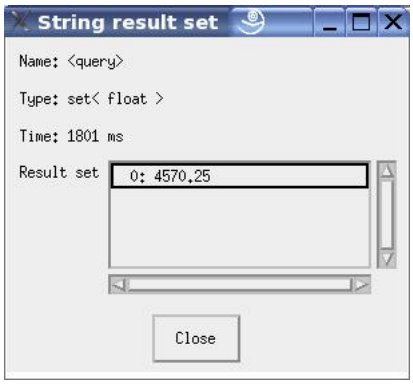


Figure 15: Variance operation.

4.3.2 Standard Deviation

Query 4.16. *Estimate the standard deviation of the cell values of the NRG raster image shown in Fig. 8(a).*

Let n be the cardinality of A, n = *card(sdom(A))*; and *avg* the average of the cell values of A, *avg=avg_cells(A)*; then the standard deviation s of A can be solved as follows:

$$s(A) = \sqrt{\frac{1}{n} * COND_{+,sdom(A),i}((A[i] - avg) * (A[i] - avg))}$$

Results are shown in Fig. 16.

4.3.3 Median

The *median* can be calculated by sorting the MDD A in ascending order and choosing the middle value. In case the number of cells is even, the median is the average of the two middle values. In solving this operation, we may use the *sort* operator to perform the ascending sorter of A. However, for A of dimensionality higher than 1 it is necessary to flat the array into a one-dimensional array. For example, the

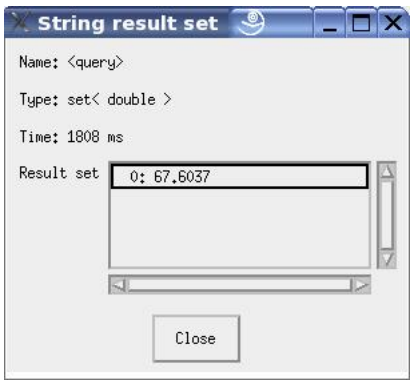


Figure 16: Standard deviation operation.

conversion from a two-dimensional MDD A[0:m,0:n] into a one-dimensional MDD B[0:m*n] can be calculated as follows:

Let d be the cardinality of A, d=*card(sdom*(A); let r be the number of rows; and let c be the number of columns. Then, the *flattening* of A can be calculated as:

$$\begin{aligned} B = & MARRAY_{[0:255],g} \ (\\ & COND_{+,[0:m,0:n],i} \ (\\ & ((g > (m * (i-1))) \ and \ (g \leq i)) * A[1:(g-(m*(i-1))), 1:i])) \end{aligned}$$

Let S be the MDD array containing the sorted values of B (the flattening of A), $S = SORT_{0,f}^{asc}$ *(B)*, and let n be the cardinality of S, n = *card(sdom(S))*. Then, the *median* of A can be solved as follows:

$$If \ N \ is \ odd \ then \ median \ = \ S[\frac{n}{2}], \ else \ median \ = \ \frac{S[\frac{n-1}{2}] + S[\frac{n+1}{2}]}{2}$$

As another example consider the following query:

Query 4.17. *Obtain the median of the 1-D array A whose cell values are shown in Fig. 17(a).*

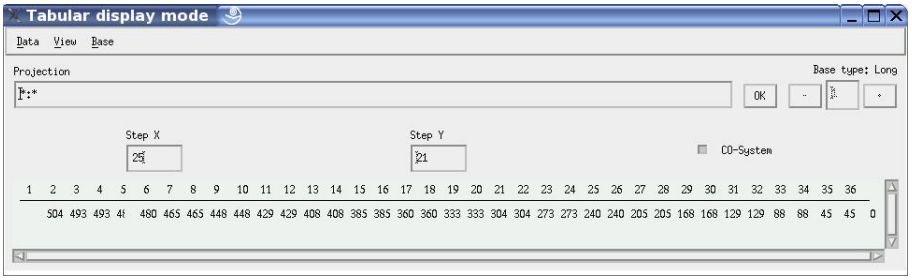
Since we have an even number of elements the computation of the query is as follows:

$$A[card(A)/2]$$

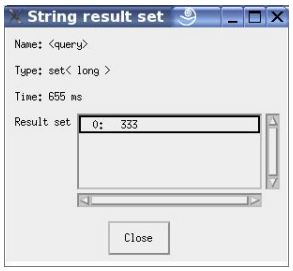
Results are shown in Fig. 17(b).

4.3.4 Top-*k*

The Top-k function returns the k cells with highest value of a MDD. For example, consider the following query:



(a) 1-D array



(b) Median

Figure 17: Median operation.

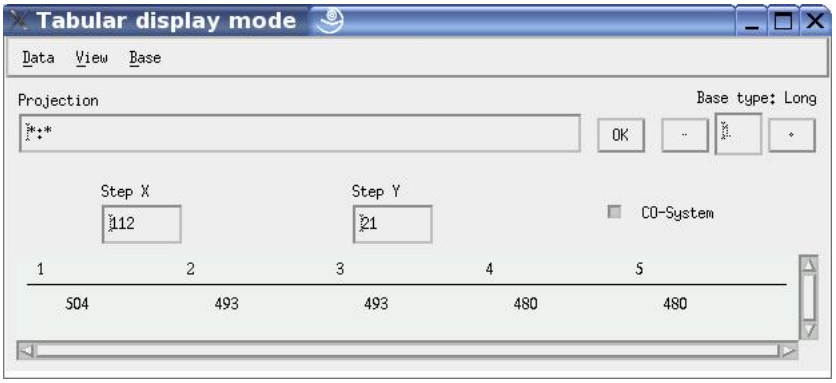
Query 4.18. *Find the 5 highest values contained in the MDD A.*

To solve this query we can first sort A in ascending order and then select the top 5 values. Let d=0 to indicate a sorting in the 0 dimension, and let f be the sorting function $f_{d,A}$(p)=A[P], then S is a sorted array of the MDD A (see Fig. 17(a)):

$$S = SORT_{0,f}^{asc}(A)$$

thus, the top 5 cell values are obtained by:

$$S[0:4]$$



(a) Top 5 values

Figure 18: Top-k operation.

4.3.5 Edge Detection

Edge detection produces a new MDD containing only boundary cells of a given MDD. The detection of intensity discontinuities in a MDD is very useful, e.g. the boundary representation is easy to integrate into a large variety of detection algorithms. The following parameterized function can be used to express filtering operations in Array Algebra:

$$f(A, M) = MARRAY_{sdom(A),x}(COND_{+,sdom(M),i}(A[x+i] * M(y)))$$

where *sdom*(M) is the size of the corresponding filter window, e.g., 3x3. As an example consider the following query:

Query 4.19. *Apply edge detection to the MDD A shown in Fig. 20(a) using a 3x3 Sobel filter.*

To compute this query, a Sobel filter and its inverse is applied in the original MDD A (see Fig. 19):

$$\frac{|f(A, M1)| + |f(A, M2)|}{9} =$$

which in Array Algebra can be computed as follows:

$$MARRAY_{sdom(A),x}\ (COND_{+,sdom(M1),i}\ (\\ (abs(A[x+i] * M1(i))) + (abs((A[x+i] * M2(i))))/9))$$

Results are shown in Fig. 20.

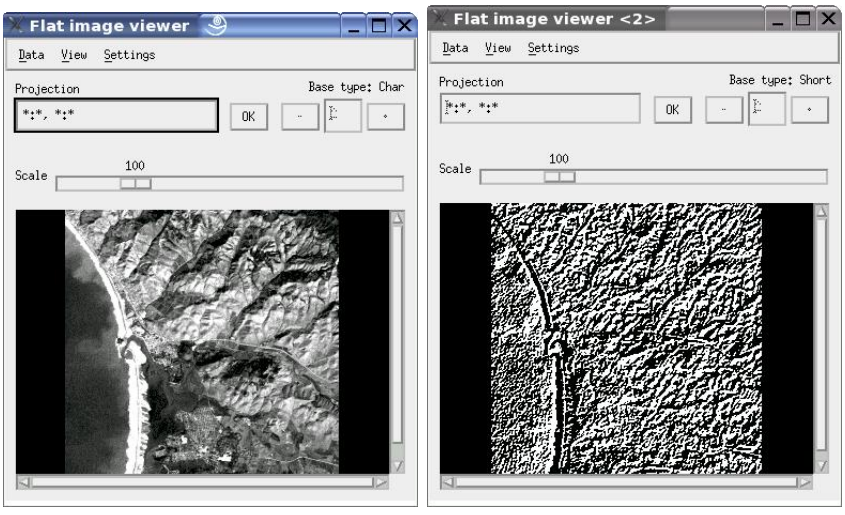
1	3	1
0	0	0
-1	-3	-1

(a) M1

1	0	-1
3	0	-3
1	0	-1

(b) M2

Figure 19: Sobel Masks



(a) Original raster image (b) Output raster image

Figure 20: Edge detection operation

4.4 Affine Transformations

Geometric transformations permit the elimination of geometric distortion that occurs when an image is captured. An example is the attempt to match remotely sensed images of the same area taken after one year, when the more recent image was probably not taken from precisely the same position. Another example is the Landsat Level 1B data which are already transformed to a plane, but that may not be rectified to the user's desired map projection [22]. Applying an affine transformation to a uniformly distorted raster image can correct for a range of perspective distortions by transforming the measurements from the ideal coordinates to those actually used. An affine transformation is an important class of linear 2-D geometric transformations which maps variables (e.g. cell intensity values located at position (*x1,y1*) in an input raster image) into new variables (e.g. (*x2,y2*) in an output raster image) by applying a linear combination of *translation, rotation, scaling* and *shearing* (i.e. non-uniform scaling in some directions) operations [29]. The computation of these operations may require of interpolation techniques. In [20], *interpolation* is defined as the process of predicting a value of an attribute $\hat{z}$ at an un-sampled site X_0. The following are the most popular approaches:

- *Nearest neighbor* selects the value of the nearest point in order to predict the cell value for a non-given point in some space. That is, the cell that is closest to the re-transformed coordinate is the nearest neighbor. This approach is suitable when using thematic rasters.

- *Bilinear interpolation* estimates the average of the four closest cells to the specified input cell and assigns that value to the output cell. Because the cell values are altered (averaged) by this method, any classification process should be performed before interpolation. Resulting output images are smoother that those obtained with *nearest neighbor*.

- *Cubic interpolation* determines the new value from the weighted average of the 16 closest cells to the specified input cell, and assigns that value to the output cell. This method has the tendency to sharpen the edges of the data more than bilinear interpolation since more cells are involved in the calculation of the output value. Bilinear or Cubic interpolation should not be used on categorical data since the original categories will not be maintained in the output raster dataset.

In the remaining of this Section we discuss special cases of affine transformations.

4.4.1 Translation

Translation performs a geometric transformation which maps the position of each cell in an input raster image into a new position in an output raster image. Under

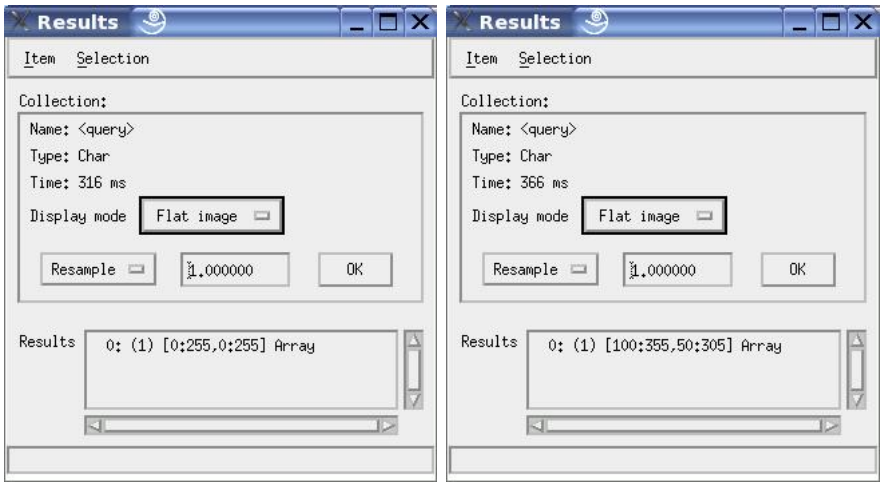
translation, a cell located at (*x1,y1*) in the original is shifted to a new position (*x2,y2*) in the corresponding output raster image by displacing it through a user-specified *translation vector* (*h,k*). The cell values remain unchanged and the spatial domain of the output raster image is the same as the original input raster. Consider for example, the following query:

Query 4.20. *Shift the spatial domain of a MDD with spatial domain A[0:255,0:255] by the point [100:50].*

The query can be solved by invoking the *shift* function of Array Algebra:

$$shift(A[0:255, 0:255], [100:50]])$$

Results are shown in Fig. 21.



(a) Original domain (b) Translated domain

Figure 21: Translation operation.

4.4.2 Rotation

Rotation performs a geometric transformation which maps the position (*x1,y1*) of a cell in an input raster image onto a position (*x2,y2*) in an output raster image by rotating it, clockwise or counterclockwise, through a user-specified angle (θ) about an origin O. Thus, the rotation operation performs a transformation of the form:

$$x2 = cos(\theta) * (x1 - x0) - sin(\theta) * (y1 - y0) + x0$$
$$y2 = sin(\theta) * (x1 - x0) + cos(\theta) * (y1 - y0) + y0$$

where (*x0,y0*) are the coordinates of the center of rotation in the input raster image, and θ is the angle of rotation. Existing algorithms for the computation of *rotation*, unlike those employed by *translation*, can produce coordinates (*x2,y2*) which are not integers. A common solution to this problem is the application of interpolation techniques like *nearest neighbor, bilinear,* or *cubic interpolation.* For large raster datasets this is a very intensive computing problem because every output cell must be computed separately using data from its neighbors. Consequently, the *rotation* operation is not yet properly supported by Array Algebra and it is currently object of research within our group.

4.4.3 Scaling

Scaling allows stretching or compressing the coordinates of a MDD (or part of a MDD) according to a *scaling factor.* This operation can be used to change the visual appearance of an image, to alter the quantity of information stored in a scene representation, or as a low-level preprocessor in multi-stage image processing chain which operates on features of a particular scale. For the estimation of the cell values in the output raster image two common approaches exist:

- one pixel value within a local neighborhood is chosen (perhaps randomly) to be representative of its surroundings. This method is computational simple but it may lead to poor results when the sampling neighborhood is too large and diverse.
- the second method *interpolates* cell values within a neighborhood by taking the average of the local intensity values.

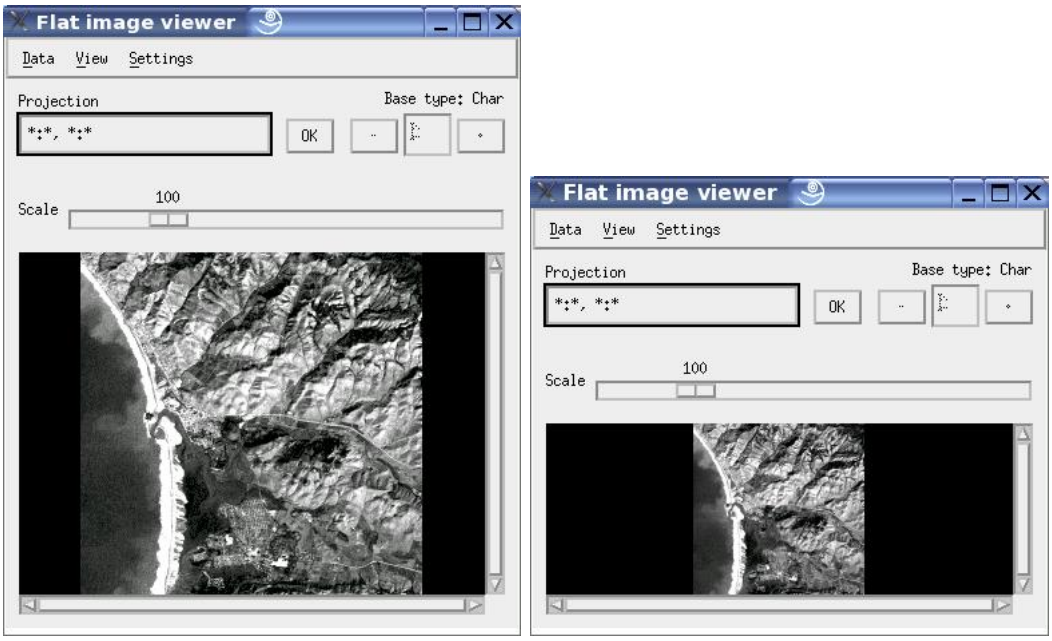
As in the *rotation* operation, the application of *scaling* using interpolation techniques in large raster datasets is a very intensive computing problem because every output cell must be computed separately using data from its neighbors. Consider the following query performing the scaling operation using *bilinear interpolation.* That is, the cell value for (*x0,y0*) in the output raster is calculated by averaging the values of its nearest cells: two along the horizontal plane (*x0,x1*) and two in the vertical plane (*y0,y1*). Note that the query is applied in a raster of spatial domain [0:255, 0:255] but as it was mentioned earlier in this report, raster datasets tend to be extremely large.

Query 4.21. *Scale the 2D raster shown in Fig. 22(a), along the x and y dimensions by a factor of 2.*

The query can be solved as follows:

$$B = MARRAY_{[0:\frac{m}{2},0:\frac{n}{2}],(x,y)}(COND_{+,[0:1,0:1],(i,j)}(A[i + x * 2, j + y * 2]/4))$$

Results are shown in Fig. 22.



(a) Original raster (b) Scaled raster

Figure 22: Scaling operation

4.5 Slicing

The *slicing* operation allows to extract lower-dimensional sections from a MDD. Array Algebra accomplishes the slicing operation by indicating the slicing position in the desired dimension. Thus, this operation reduces the dimensionality of the MDD by one. For example, consider the following query:

Query 4.22. *Slice the MDD A along the second dimension at position 50.*

The query is solved by specifying the slicing position as follows:

$$MARRAY_{sdom(A),(x,y,z)}(A[x, 50, z])$$

4.6 Terrain Analysis

Raster image data is particularly useful for tasks related to *terrain analysis*. Some of the most popular operations include *slope/aspect, drainage networks,* and *catchments (or watersheds)*. The processing of these operations may involve interpolation techniques, which lead to expensive computational costs. For simplicity, we model these operations with approaches not using interpolation methods.

4.6.1 Slope/Aspect

Slope is defined by a plane tangent to a topographic surface, as modelled by the DEM at a point [20]. Slope is classified as a *vector*, thus having two components: a quantity (*gradient*) and a direction (*aspect*). The *slope* (gradient) is defined as the maximum rate of change in altitude, and *aspect* as the compass direction of this maximum rate of change. Several approaches exist for the computation of slope/aspect, we follow the method proposed by [21]:

- slope in the X direction (difference in the height values either side of P is give by:
$$Tan\Theta_x = \frac{z(r, c+1) - z(r, c-1)}{2g}$$
- slope in the Y direction
$$Tan\Theta_y = \frac{z(r+1, c) - z(r-1, c)}{2g}$$
- Gradient at P
$$\sqrt{(tan^2\Theta_x + tan^2\Theta_y)}$$
- Direction or aspect of the gradient
$$tan\alpha = \frac{tan\Theta_x}{tan\Theta_y}$$

Note that after the calculation of the slopes for each cell in a raster image, the results may need to be classified in order to display them clearly on a map [20].

Query 4.23. *Calculate the slope along the X direction of a 8-bit grey MDD A:*

$$MARRAY_{sdom(A),(r,c)} \frac{(arctan(A(r, c+1) - A(r, c-1)))}{2g}$$

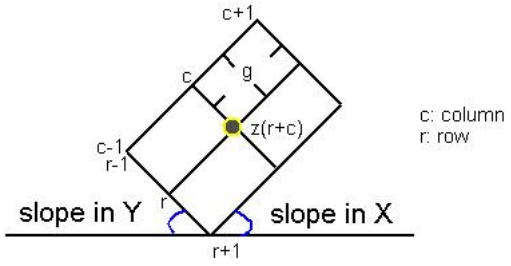


Figure 23: Slopes along the X and Y directions.

4.6.2 Local drain directions (ldd)

The *ldd* network is useful for computing several properties of a DEM because it explicitly contains information about the connectivity of different cells. Two steps are required to derive a drainage network: the estimation of flow of material over the surface and removal of pits. For instance (see Fig. 24), cell A1 has 3 neighboring cells (A2, B1 and B2) and the lowest of them is B1, thus the flow direction is south (downward). For cell C3, the lowest of its 8 neighboring cells is D2, so the flow direction is southwest (to lower left). This method is one of the most popular algorithms to estimate flow directions and it is commonly known as D8 algorithm [20].

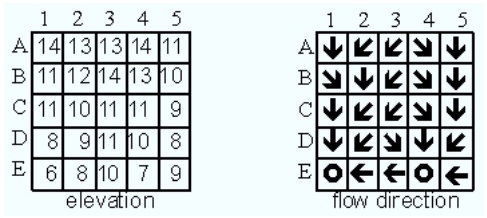


Figure 24: Flow Directions

Query 4.24. *Estimate the flow of material over a MDD A where each cell contains the slope along the X direction.*

Let A be a MDD with the slopes along the X direction of A, then the ldd is calculated as:

$$MARRAY_{sdom(A),(i,j)}(COND_{min,[-1:1,-1:1],(v,w)}\ (A[i+v, j+w]))$$

Irrespective of the algorithm used to compute the flow directions, the resulting ldd network is extremely useful for computing other properties of a DEM, e.g., *stream channels*, *ridges*, and *catchments*.

5 A Raster Operations Classification Scheme

By examining the fundamental structure of the operations presented in Section 4 and by breaking down all the steps required for their computation to a few basic Array Algebra operators, we distinguish the following basic classes of operations with geo-raster data:

- COND and MARRAY combined operations. This group contains operations whose computation requires of both MARRAY and COND operators:
  add, count, average, maximum, minimum, majority, minority, histogram, diversity, variance, standard deviation, scaling, edge detection, and *local drain directions.*

- MARRAY exclusive operations. This group contains operations whose computation requires only of the MARRAY operator:
  arithmetic, trigonometric, boolean, logical, overlay, reclassification, proximity, translation, slicing, and *slope/aspect.*

- SORT operations. This group contains operations whose computation requires of the SORT operator: *top-k, median.*

- AFFINE transformations. This group contains special cases of affine transformations partially or not supported yet by Array Algebra:
  interpolation, rotation, scaling, re-projection

From these groups of operations we can identify a set of operations that require of data summarization and thus, the potential candidates to be treated with OLAP pre-aggregation techniques: *add, count, average, maximum, minimum, majority, minority, histogram, diversity, variance, standard deviation, scaling, edge detection,* and *local drain directions.*

Table 3 summarizes the usage of Array Algebra operators for each operation discussed in Section 4.

6 Summary and Future Work

In this report we have presented a set of fundamental operations with raster image data. The selection of the operations was mainly derived from an exhaustive review of existing surveys in GIS operations. In order to better understand the structure of the most common queries in this type of databases, we have modeled the operations using an algebraic framework in our case, Array Algebra. By comparing the operators required for the computation of the modeled operations we identified a set of operations that require of data summarization (aggregation) and therefore, potential candidates to be treated with pre-aggregation algorithms.

[3] *Array Algebra* partially supports the scaling operation.

Table 3: An operators-based classification of geo-raster operations.

Operation	MARRAY	COND	SORT	AFFINE
1. Count		x		
2. Add		x		
3. Average		x		
4. Maximum		x		
5. Minimum		x		
6. Majority	x	x		
7. Minority	x	x		
8. Std. Deviation		x		
9. Median		x	x	
10. Variance		x		
11. Top-k			x	
12. Histogram	x	x		
13. Diversity	x	x		
14. Proximity	x			
15. Arithmetic	x			
16. Trigonometric	x			
17. Boolean	x			
18. Logical	x			
19. Overlay	x			
20. Re-Classification	x			
21. Re-projection				x
22. Interpolation				x
23. Translation	x			
24. Rotation				x
25. Scaling[3]	x	x		x
26. Slicing	x			
27. Edge Detection	x	x		
28. Slope/Aspect	x			
29. Local drain directions (ldd)	x	x		

We are currently working in the formalization of the required elements in the raster data model for the implementation of OLAP pre-aggregation algorithms. We believe that applying such techniques will speed up significatively the processing of aggregate queries in raster image databases.

References

[1] Hanan Samet, Foundations of Multidimensional and Metric Data Structures, pp. 191-202, Morgan Kaufmann Publishers, 2006.

[2] Gutpa, A., Harinarayan, V., Quass, D., Aggregate-Query Processing in Data Warehousing Environments, Proc. 21st VLDB Conference Zurich, Swizerland, 1995.

[3] Zhuge, Y., Garcia-Molina, H., Hammer, J., Widom, J., View Maintenance in a Warehousing Environment, Proc. of the ACM SIGMOD International Conference on Management Data, 1995.

[4] Baralis, E., Paraboschi, S., Teniente, E., Materialized View Selection in a Multidimensional Database, Proc. of the 23rd BLVD. conference in Athens, Greece, 1997.

[5] Shukla, A., Naughton, K., Deshpade, P., Materialized View Selection for Multidimensional Datasets, Proc. of 24th VLDB Conf., pp.488-499, 1998.

[6] Ramachandran, K., Shah, B., Raghavan, V., Dynamic Pre-Fetching of Views Based On User-Access Patterns in an OLAP system, 2005.

[7] Harinarayan, V., Rajaraman, Ullman, J., Implementing Data Cubes Efficiently, Proc. SIGMOD, pp. 205-216 1996.

[8] Gutpa, A., Harinarayan, V., Rajaraman, A., Index Selection for OLAP, Proc. of ICDE, pp. 208-219 1997.

[9] Ullman, J., Efficient Implementation of Data Cubes via Materialized Views, Proc. KDD, pp. 386-388 1996.

[10] Agrawal, S., Chaudhuri, S., Narasaya, V., Automated Selection of Materialized Views and Indexes in SQL Databases Proc. of VLDB, pp. 496-505 2000.

[11] Shrivastava, D., Dar, S., Jagadish, H., Levy, A., Answering Queries with Aggregation Using Views Proc. of VLDB, pp. 318-329 1996.

[12] Raja S., Chen Y., A Hybrid Spatio-Temporal Data Model and Structure (HST-DMS) for Efficient Storage and Retrieval of Land Use Information, Transactions in GIS 8 (3), pp. 351-366, 2004.

[13] Peuquet Donna J., Making space for time: Issues in space-time data representation, Geoinformatica (Geoinformatica) ISSN 1384-6175, vol. 5, pp. 11-32, 2001.

[14] Baumann P., A Database Array Algebra for Spatio-Temporal Data and Beyond, NGITS'99 LNCS 1649 pp.76-93, 1999.

[15] Howe B., Algebraic Manipulation of Scientific Datasets, Proc. of the 30th VLDB conference, 2004.

[16] van Ballegooij A. R., Cornacchia R., de Vries A. P., Kersten M. L., Distribution Rules for Array Database Queries, International Workshop on Database and Expert Systems Application, pp. 55-64, 2005.

[17] Reiner B., Hanh K., Tertiary Storage Support for Multidimensional Data, IEEE Distributed systems, 1541-4922, vol 5. No.5, 2004.

[18] Rivest S., Bedard Y., Proulx M-J., Nadeau M., Hubert F., Pastor J., SOLAP technology: Merging business intelligence with geo-spatial technology for interactive spatio-temporal exploration of data, ISPRS Journal of Photogrammetry and Remote Sensing, 2005.

[19] Han J., Stefanovic N., Koperski K., Selective Materialization: An Efficient Method for Spatial Data Cube Construction, Proc. Conference on knowledge discovery and data mining, pp.144-158, 1998.

[20] Burrough P.A., McDonell R.A., Principles of Geographical Information Systems, Ch.7, pp. 162-179, Ch. 8, pp. 183-185 Oxford, 2004.

[21] O'Sullivan D., Unwin D., Geographic Information Analysis, John Wiley, 2003.

[22] ERDAS IMAGINE, ERDAS Field Guide, Fourth Edition, 1997,

[23] Albrecht J. H., Universal GIS Operations for Environmental Modeling, Proc. 3rd International Conference on Integrating GIS and Environmental Modeling, Santa Fe, NM. Santa Barbara, 1996.

[24] Kvamme Kenneth A., Fundamental Raster Operations, n.d.

[25] Aronoff, S., Geographic Information Systems: A Management Perspective, WDL Publications, p. 294. 1991.

[26] Nguyen D. H., Using Javascript for some Interactive Operations in Virtual Geographic Model with GEOVRML, Proc. Int. Symposium on Geoinformatics for Spatial Infrastructure Development in Earth and Allied Sciences, 2006.

[27] ArcGIS 9, Geo Processing Commands, quick reference Guide, 2004.

[28] Open GIS Consortium, Web Coverage Processing Service (WCPS), best practices document # 06-035r1, pp. 21-47, 2006.

[29] Lusch D.P., A Classification of GIS Functions, Technical Report, Center for Remote Sensing, Michigan State University, 1999.

[30] Mennis J., Viger R., and Tomlin C.D, Cubic Map Algebra Functions for Spatio-Temporal Analysis, Cartography and Geographic Information Science, Vol. 32, No. 1, 2005, pp. 17-32

[31] Tomlin C. D., J. K. Berry, A mathematical structure for cartographic modeling in environmental analysis. In 39th annual symposium proceedings. American Congress on Surveying Mapping, pp. 269-284, 1979.

[32] Tomlin C.D., Geographic Information Systems and Cartographic Modelling, Prentice-Hall, New Jersey, 1990.

[33] Libkin L., Machlin R., Wong L., A query language for multidimensional arrays: design, implementation and optimization techniques. In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'96), pages 228-239.

[34] Baumann P., On the Management of Multidimensional Discrete Data, VLDB Journal 4(3), pp. 401 - 444. Special Issue on Spatial Database Systems, 1994.

[35] Baumann, P., The RasDaMan Array Algebra, RasDaMan Project Technical Report for012, FORWISS, 1998.

[36] Ritter G., Wilson J., Davidson J., Image algebra: An Overview, Computer Vision, Graphics, and Image Processing, 49(1):297-331; 1990.

[37] Marathe A., Salem, K., A language for manipulating arrays. In Proc. International Conference on Very Large Data Bases (VLDB'97), pages 46-55, August 1997.

[38] Furtado P., Storage Management of Multidimensional Arrays in Database Management Systems. Dissertation, Technische Universitdt M—nchen, 1999.

[39] Baumann P., Large-Scale Raster Services: A Case for Databases. Invited keynote, 3rd Intl Workshop on Conceptual Modeling for Geographic Information Systems (CoMoGIS), In: John Roddick et al (eds): Advances in Conceptual Modeling - Theory and Practice, 2006, pp. 75 - 84, 2006.

[40] Ritter G., Wilson J., Davidson J., Image Algebra: An Overview, Computer Vision, Graphics, and Image Processing, 49(1), pp. 297-336, 1994.

[41] Baumann, P., Language Support for Raster Image Manipulation in Databases, Proc. Int. Workshop on Graphics Modeling and Visualization in Science & Technology, Darmstadt, Germany, 1992.

[42] Stonebraker M., Moore, D., Object-Relational DBMSs. The Next Great Wave. Morgan Kaufmann Publishers, 1996.