Querying the Content of Images in Material Science: An Integration of SQL and Map Algebra

Yan Huang*, Brian Harrington
Computer Science & Engineering
University of North Texas
$[huangyan, brh]@cs.unt.edu$

Nandika Dsouza
Material Science
University of North Texas
$ndsouza@unt.edu$

Robert Brazile
Computer Science & Engineering
University of North Texas
$brazile@cs.unt.edu$

Abstract

Material scientists regularly acquire and analyze infrared images of deforming objects in their material tensile deformation, crack propagation, and fracture toughness tests. Although there are many image processing packages, none of them are available in a database integrated fashion. Material scientists typically select a set of image files satisfying given constraints by browsing a file directory/catalog, and then perform simple but labor-intensive content-querying on the images. These simple queries take days to answer. A more serious problem is that material scientists are not equipped with the flexibility to query images across different time snapshots or materials to validate their research hypotheses.

In this paper, we report about our work on helping material scientists accelerate their work. Specifically, we propose a database approach to solve the problem of storing images and querying the content of images. In particular, we (1) proposed to use map algebra operations to compose image content querying needed by material scientists; (2) developed an SQL integrated image data cartridge which implements a core set of map algebra operations needed by material scientists; (3) analyzed the query processing and evaluation challenges; and (4) empirically evaluated the performance of three approaches, namely a multi-dimensional array based approach, a relational table based approach, and a binary large object (BLOB) based approach on bulk loading and typical material science queries using both real and synthetic data.

1 Introduction

Material scientists regularly acquire and analyze infrared images of deforming objects [22, 23, 3, 4] in their material tensile, crack propagation, and fracture toughness tests. The technique of thermal wave imaging utilizes the theories of radiation heat transfer which occur in the wavelengths of 3-13 m band. The overlap of a substantial wavelength with that of infrared (IR) wavelength leads to the utility of IR thermometry to interpret the temperatures generated in the emitting material. Figure 1 shows the basic setup material scientists use to study the materials.

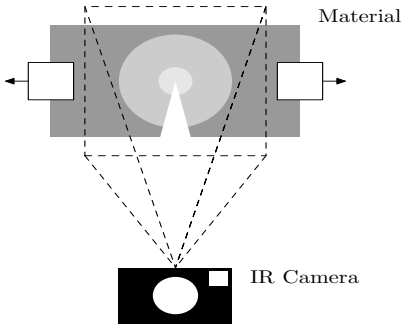
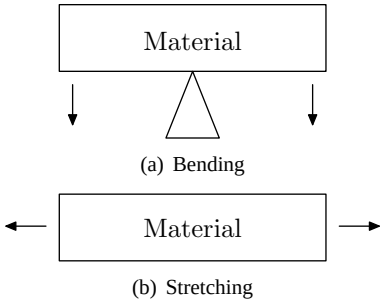


Figure 1. Setup used for collecting images.

Test machines such as Instron Materials are used to apply bending or stretching to material samples as shown in Figure 2, which allows material scientists to study how the material reacts under stress and strain. During the test the IR emission from the sample is monitored as a function of time using infrared cameras such as an Inframetrics IR740 Camera (8 m-12 m spectral band) at a 30 Hz frame rate. When a crack forms, energy is dissipated as heat [22, 23]. In general, the area around the crack-tip should have the highest temperature. Figure 3 (gray scaled to allow proper print) shows two images from the same test at different times. Notice that the area between the two crack-tips is brighter in the latter image indicating it has a higher temperature.

Each sample image set has associated meta-data which may correspond to the type of material, the concentration of filler, timestamps for the images, and the experimental setup. Material scientists ask various spatio-temporal questions [22, 23] about the images they capture, for example (1) Where is(are) the crack-tip(s) for a sample? (2) What is the average temperature 5 mm from the crack-tip for a given sample? (3) What is the temperature change for a point

*The work is partially supported by the Oak Ridge Associated University Ralph E. Powe Junior Faculty Enhancement Award.

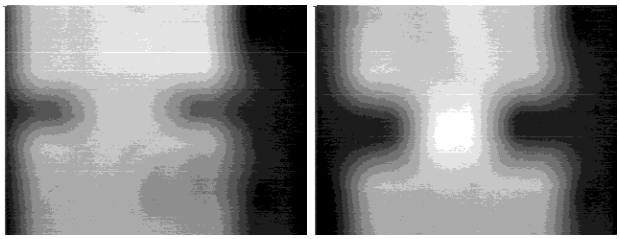


(a) Bending

(b) Stretching

Figure 2. Deformations applied to the material.

(x, y) while a sample is pulled? (4) What is the change in distance between two marked points after a given material is pulled? (5) What is the area of deformation for a sample? (6) What is the difference in the area of deformation for different samples at time=10 seconds? (7) How do the contours of the image grow over time? (8) When does the area of deformation become greater than A? (9) Which materials have similar areas of deformation to a given sample? These questions involve both meta-data (e.g. material and time) and content of the sample images (e.g. crack tip and formation area).



(a) Time=0 sec (b) Time=13 sec

Figure 3. Snapshots of a material as it is being stretched.

Like scientists in many other domains, material scientists are facing both data processing and data management challenges. Currently, all questions are typically answered by isolated software packages, e.g. Image J and Matlab, on isolated image files. It takes material scientists days to get answers to simple queries on these data using either inflexible predefined functionalities of software packages or writing customized codes. A more serious problem is that material scientists are not equipped with the flexibility to issue ad hoc queries across different time snapshots or materials to validate their research hypotheses. Their research has been greatly hindered by the limited expressive power of image querying languages/tools.

In this paper, we report our work on helping material scientists accelerate their answers to their research questions. Specifically, we propose a database approach to solve the problem of storing and querying images that has plagued material scientists for years. In particular, we first proposed to use map algebra operations to compose image content querying needed by material scientists. Then we developed a SQL integrated image data cartridge which implements a core set of map algebra operations needed by material scientists. Third, we analyzed the query processing and evaluation challenges. We proposed possible solutions although the actual implementation and evaluation of these solutions are out of the scope of this paper. Finally, we empirically evaluated the performance of three basic approaches, namely a multi-dimensional array based approach, a relational table based approach, and a binary large object based (BLOB) approach on bulk loading and typical material science queries using both real data and synthetic data.

The paper is organized as follows. Section 2 starts off by surveying related work. The proposed SQL integrated map algebra approach is described in section 3. Section 4 discusses the various implementation options of the proposed approach and their query processing challenges. We present the experiment results on the performance of data load and typical queries on both real and synthetic datasets in section 5.The paper concludes by discussing some of the future work in section 6.

2 Related Work

Researchers in scientific database management have been addressing the problem of organizing and querying images for many years [2, 19, 7]. Images typically have both meta-data and content. Approaches for managing images have been centered around meta-data based, content based, or a hybrid approach. A meta-data based approach could easily adopt traditional database operations. A content based approach, noticeably Content Based Image Retrieval (CBIR) [21], retrieves images similar to a given image using features derived from image content, e.g. color, textures, and shapes. A hybrid approach allows queries involving both meta-data and image contents. However the "content" in research literature is generally referring to features describing the images in a very general sense, e.g. color histogram and textures. In a broad sense, our work is related to the hybrid approach. The uniqueness of our approach is the ability to allow material scientists to ask ad hoc queries to the content of their images in a SQL integrated way.

Material scientists are interested in quantifying qualitative assessment of failure in materials. Temperature changes on the surface represented by thermal images are related to the change in stresses via thermoelastic equations. Quantification of this has been limited since tools that allow queries across data that are continuous across the surface are not generally available to material scientists. As a result, one of the authors had to resort to map algebra [18, 5, 13],

supported by many GIS software packages, to query the images in a semi automatic but still laborious fashion.

Spatial information is generally represented by two models: *object* and *field* [16]. The object model represents conceptual entities as objects in an intuitive and direct way. The *vector* data structure implements the *object* model using polygons, lines, and points to represent shapes of objects, e.g. lakes, buildings, and rivers. The *field* model is often used to represent continuous phenomena over a space, e.g. temperature. The raster data structure obtained by imposing a uniform grid on the underlying space implements the *field* model on a computer. Map algebra [18, 5, 13] is an informal and the *de facto* geographic and cartographic modelling language for manipulating raster data. As a high-level computational language to describe geographic data processing, map algebra creates new map layers using existing map layers and operations in a sequence. It is a powerful raster manipulation language by allowing calculation of values for locations based on the location itself, its neighborhood, a related zone, or the entire raster. Although supported by many GIS software products, e.g. ArcInfo [6], AutoCAD, and PCRaster [11], map algebra is not supported in an SQL integrated fashion. Users first need to select the raster files satisfying given constraints, e.g. all images of material X with $y\%$ filler concentration, by either browsing a file directory/catolog or writing SQL statements. The names of raster files are then embedded into map algebra operations manually.

A logical way to support image content querying in a database system is to extend current database data types and query languages. Many object-oriented or object-relational database management systems allow abstract data types (ADT) together with stand alone/member functions defined by end users. Such research prototypes/commercial/open source software products include Postgres [8], Monet [1], Jasmine [9], Starburst [15], Oracle, DB2, and MySQL. Many of them support images as binary large objects (BLOB). A cartridge supporting map algebra in an SQL integrated fashion is not known.

In a similar vein as our proposed approach in this paper, Nes proposed to add core image processing functionality to the database management system, making it a better tool for image analysis research in his thesis [10]. Image algebra [14] is a formal set of image-processing operations many of which may not have practical usage in material science image content querying, which is a very light form of image analysis. Geo-algebra [17] attempts to borrow techniques from image algebra to formalize and extend the functionalities of map algebra.

RasDaMan [20] is an extension of an ODBMS to support multidimensional arrays. In addition, it also supports an extended version of SQL called RasQL that allows the user to select and manipulate multidimensional array data. However, for the queries posed by material scientists the expressive power of RasQL is limited by the difficulty of performing focal and zonal operations.

3 SQL Integrated Map Algebra Approach

Fracture mechanics relies on balancing the energy before and after failure in terms of incremental crack growth. Since temperature has a dominating effect on polymeric materials, analysis of the temperature fields around the crack tip over the time of propagation of the crack to failure yields information on the state of the material. Being able to issue ad hoc queries in the form of :"What is the temperature change for a point (x, y) while a given material X with filler concentration $s\%$ is pulled?" or "What is the average temperature l mm from the crack-tip for material X with filler concentration $s\%$ at time t?" helps material scientists to study the state of a material at a given time. The latter question could also be easily extended to queries like "What is the rate of temperature change l mm from the crack-tip for material X with filler concentration $s\%$ at time t? This section describes how map algebra operations can be used in conjunction with SQL to answer these queries.

We classify the material science queries into two categories, namely image content queries and multi-criteria queries. The first category has simple meta-data conditions to specify which images you want content information for, while the second category has mixed meta-data and image content conditions.

3.1 Map Algebra

Map algebra [18] is a cell by cell combination of raster layers using some mathematical operation. A raster layer is a grid imposed over an image with a value for each cell. Map algebra can be extended to support most mathematical operations found in a spreadsheet tool. The result of a map algebra operation is another raster layer. Operators can be grouped into four basic categories (1) local operations where the value of a cell in the output is a function of the corresponding cells of the input; (2) focal operations that determine the output value of a cell based on a small specified neighborhood; (3) zonal operations that determine the output based on all of the values in a zone; (4) global operations determine the output based on all of the values in the raster.

1	2	4		2	3	5		3	5	9
3	6	7	+	4	7	8	=	7	13	15
5	9	8		6	1	9		11	10	17

Figure 4. Local addition operator.

Figure 4 shows a local addition operator on two rasters. Local operations are useful for general manipulation such as for looking at the difference in temperatures between two images of a material over time.

Figure 5 shows the general focal max with a 3×3 neighborhood. This operation chooses the new cell value for the center pixel of the output by looking at all of the cells in

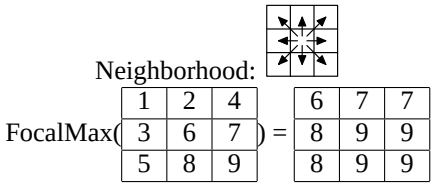


Figure 5. Focal maximum operator.

the specified neighborhood. Focal operations are useful in many material science image queries, e.g we can find the boundaries of the temperature zones by using the focal majority operator (will be described in detail later in section 3.2).

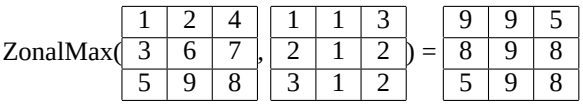


Figure 6. Zonal maximum operator. The first raster is the input and the second is the zonal map.

For some tasks zonal operations are also useful. For example, as stated earlier, the crack-tip in general should have the highest temperature. In order to find the location of the crack-tip, we need to know the location with the highest temperature or in terms of the raster the pixel with the maximum value in some zone. Figure 6 shows how the zonal maximum operator works. A zone is the set of cells that have the same value in the zonal map raster. The new cell value is determined by looking at all cells in the same zone. Global operations are a special case of zonal operations where the entire raster is one zone.

Local Operations	
Arithmetic	add, subtract, abs, sqrt, etc...
Boolean	equal, greater than, less than
Aggregate	min, max, minority, majority, median, sum
Statistic	standard deviation, mean
Focal Operations	
Aggregate	min, max, minority, majority, median, sum
Statistic	standard deviation, mean
Zonal and Global Operations	
Geometric	fill, area, perimeter, thickness, distance
Aggregate	min, max, minority, majority, median, sum
Statistic	standard deviation, mean
Miscellaneous Operations	
Miscellaneous	get value, trim

Table 1. Map algebra operations.

There are also two useful operations that do not fit into the previously defined categories, both involve reducing the dimensions of an input raster to get a new raster. The first provides for getting the value of a specified cell or zone and returns a scalar value[1]. The second trims a raster and returns some sub-raster.

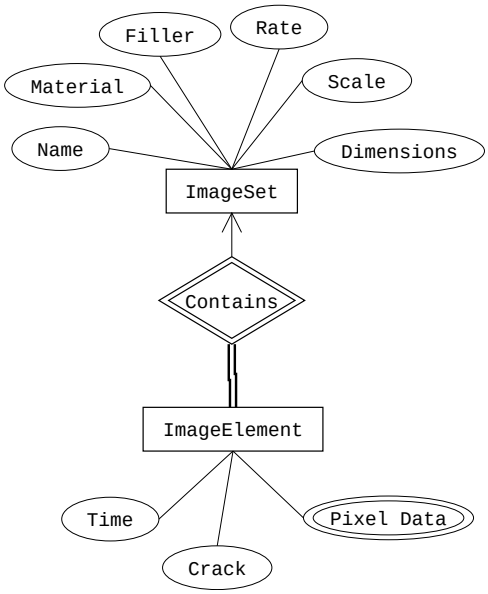


Figure 7. ER diagram of schema.

Many GIS products support some form of map algebra, however currently there is no map algebra standard. For this paper table 1 summarizes the representative operations we use for each category.

For all our queries we will assume a simple schema shown in the entity-relationship diagram in figure 7. The *ImageSet* entity set would contain the general description of a set of images. This would include the meta-data that is the same for every image in the set such as the set name, material, filler, the scale for the images, and the dimensions of the images, and strain rate applied to the material. For example, the material might be a polypropylene + ethylene propylene diene blend, with a filler that is a 1000 nanometer wide by 1 nanometer thick ceramic platelet added to the material, and a 4 mm/min strain rate. For simplicity, we will use filler concentration to represent questions related to various treatment applied to the material. The *ImageElement* entity set has the actual image data along with the time stamp for the image and the location of the crack tip.

3.2 Expressing Image Content Queries

The first class of queries involves querying the content of the images. The constrains of these queries are simple meta-data constraints, e.g. material type and filler concentration. The main challenge is figuring out how to express the query in terms of map algebra operations.

It is assumed for this paper that the materials have only one crack and thus one crack-tip. The crack propagation

[1] This can be thought of as a 1×1 raster so the operation is still closed.

ID	SQL Statement	Description
Q1	**SELECT** ct.crack_tip **FROM** ImageElement **as** ie, ImageSet **as** is **WHERE** Material=X **and** Time=t **and** Filler=s **and** is.SetID=ie.SetID;	Where is(are) the crack-tip(s) for material X with filler concentration $s\%$ at time t?
Q2	**SELECT** FindShape(ie.Data) **FROM** ImageElement **as** ie, ImageSet **as** is **WHERE** Material=X **and** Filler=s **and** is.SetID=ie.SetID;	What is the shape of the crack for material X with filler concentration $s\%$ at time t?
Q3	**SELECT** GetValue(ie.Data, x, y) **FROM** ImageElement **as** ie, ImageSet **as** is **WHERE** Material=X **and** Filler=s **and** is.SetID=ie.SetID;	What is the temperature change for a point (x, y) while a given material X and filler concentration $s\%$ is pulled?
Q4	ZM=MakeRaster("circle", CX, CY, ToPixels(5)) **SELECT** GetValue(ZAvg(ie.Data, ZM), ZM, 1) **FROM** ImageElement **as** ie, ImageSet **as** is **WHERE** Material=X **and** Filler=s **and** is.SetID=ie.SetID **and** time = t;	What is the average temperature 5 mm from the crack-tip for material X with filler concentration $s\%$ at time t?
Q5	**SELECT** Time, CreateMap(ie.Data) **FROM** ImageElement **as** ie, ImageSet **as** is **WHERE** Material=X **and** Filler=s **and** is.SetID=ie.SetID;	How do the contours of the image grow over time?

Table 2. Typical Material Science Image Content Queries(Q1-Q5) and Their SQL Statements

rate is important for determining how resilient a material is to stress. The propagation rate can easily be determined given the location of the crack-tip for each image. So the query can be formulated as "(Q1) Where is the crack-tip for material X with filler concentration $s\%$ at time t?". As the crack-tip is part of the meta-data this query can be answered using standard SQL. The resulting SQL statement is shown in Table 2.

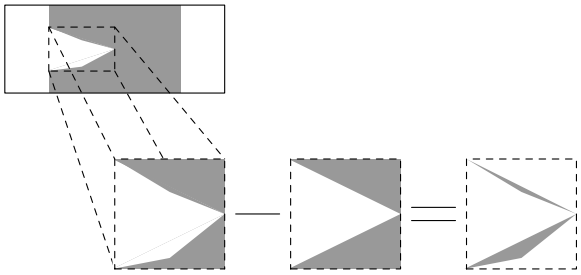


Figure 8. Finding the shape of a crack.

The shape of the crack that is formed in the material is important because it can indicate how fast the crack is likely to propagate through the material. A common question would be "(Q2) What is the shape of the crack for material X with filler concentration $s\%$ at time t?".

To determine the shape of the crack the image first needs to be cleaned up. For determining the shape of the crack only two zones are needed: one that represents the background with a value of zero in every cell, and one that represents the material with a value of one in every cell. This can be done using a simple local thresholding operation. Then the minimum bounding box around the crack is selected as shown in figure 8. Since the location of the crack-tip is part of the meta-data, a reference raster can be created for a particular shape such as a triangle. The reference raster is subtracted from the actual crack. The fewer non-zero values there are in the result the closer the actual crack is to the reference shape. The *FindShape* macro performs the steps outlined earlier for each of the basic crack shapes to see which shape matches with the highest probability. The returned result is a textual description of the basic shape, e.g. triangle or parabolic. The resulting SQL statement is shown in table 2.

As mentioned earlier temperature has a dominating effect on polymeric materials so it is important to be able to query the temperature in a number of ways including: a specific value "(Q3) What is the temperature for a point (x, y) while a material X with filler concentration $s\%$ is pulled?", the average for a set of values "(Q4) What is the average temperature p mm from the crack-tip for a material X with filler concentration $s\%$ at time t?", and for the growth of temperature zones "(Q5) How do the contours of the image for a material X with filler concentration $s\%$ grow over time?". Query Q5 is particularly useful as fracture toughness equations call for understanding the volume of the deformed material.

Query Q3 can be answered by simply using the get value operation to get a scalar value. As the image uses pixels a simple macro can be defined to convert millimeters to pixels as the dimensions of a pixel in millimeters is part of the meta-data for each image set. Query Q4 can be answered by creating a zone map raster that has a 1 in every cell that is p mm from the crack-tip and getting the value of that zone. Q3 and Q4 are shown in table 2

Query Q5 is more complicated. Since an image can have hundreds of distinct temperature values, the first step is to assign ranges of temperatures to a given value so that the result will have the desired number of zones. This is similar to adjusting the distance between contours on a topographic map to get the desired amount of detail. Let R be the original raster and M be the contour map.

$$T = GMax(R) - GMin(R) \quad (1)$$

$$D = LDivide(T, MakeRaster(n)) \quad (2)$$

$$M = FMajority(LRound(R, D, B)) \quad (3)$$

Equation 1 uses the global operations min and max to find the overall temperature range for the image. The output raster, T, is a raster where every cell has the temperature range as the value. In equation 2 the temperature range is divided by the desired number of zones, n. The function $MakeRaster(n)$ creates a raster where every cell has the value n. Equation 3 produces the final contour map raster, M, by rounding the values and using the focal majority operation with a 3×3 neighborhood to clean up the zones. Equation 4 shows how the i, j^{th} cell of the raster returned from the local round operation is determined from the corresponding cells of the three input rasters. A good value for B is $GMin(R)$.

$$R'_{i,j} = \left\lfloor \frac{R_{i,j} - B_{i,j}}{D_{i,j}} \right\rfloor D_{i,j} + \frac{D_{i,j}}{2} + B_{i,j} \quad (4)$$

As you will see later creating a contour map is useful for other operations. As such we will define the *CreateMap* macro which performs the steps outlined earlier to create a contour map from an image. Q3, Q4, and Q5 are shown in table 2.

Stress-strain curve could be used to divide materials into different classes that depend on their relative behavior under stress. Stress is a force applied over an area. When materials deform they are said to strain. A strain is a change in size, shape, or volume of a material. When a material is subjected to increasing stress it passes through three successive stages of deformation: reversible elastic deformation, irreversible ductile deformation, and fracture. Depending on the regions of elastic and ductile behaviors, materials can be said to be brittle or ductile. The extent of certain parts of the material have stretched could be used to measure the strain of the material. So material scientists want to know "(Q6) What is the change in distance between two marked points as the material is pulled?". This query can be answered by using the global distance function to find the distance from the cell marked $M1$ to every other cell. Then, using the original image as a zone map, get the value of the cell marked $M2$.

Many of the equations for determining the energy that is being dissipated as heat require the area of the deformed region. Given a contour map of the image this can be thought of as the area of the zone that has the highest temperature. In particular queries such as "(Q7) What is the area of deformation for material X with filler concentration $s\%$?" and "(Q8) What is the difference in the area of deformation for different samples at time t?" are very useful. As finding the deformation area is fairly common a *DfrmArea* macro can be defined that finds the area of the contour with the highest temperature as follows:

```
DfrmArea(R)
  MAP:=CreateMap(R, n)
  ZONE:=GetValue(GMax(MAP), 0, 0)
  return GetValue(ZArea(MAP), MAP, ZONE)
```

Essentially it creates a contour map and then uses the global max operation to find the correct zone. As global max creates a raster where every cell has the same value, we can get any value to pass to the operation for getting the value of a zone. The zonal area function calculates the area of each zone and then the value of the zone with the highest temperature is returned. Q6, Q7, and Q8 are shown in table 3.

3.3 Expressing Multi-Criteria Queries

In this section, we discuss queries whose constraints use information about image content as well as other meta-data to determine the result set.

Materials are selected by matching their properties to the service conditions. The growth rate of the area of deformation of a material is an important property of a material. The deformation area is defined as the area of the highest temperature zones. Useful questions include "(Q9) When does the area of deformation become greater than A?" and "(Q10) Which materials have similar areas of deformation to material X with filler concentration $s\%$ at time t?". Query Q9 requires a query with a image content condition "deformation area is greater than A" to be specified and can be expressed as:

```
SELECT Min(Time)
FROM ImageElement as ie, ImageSet as is
WHERE Material=X and Filler=s and
   is.SetID=ie.SetID
   and DfrmArea(ie.Data)>A;
```

Query Q10 is a complex query involving calculating intermediate results storing the deformation area of a given material with a given filler concentration at a given time and comparing every other images in the database with it. This query can be expressed as:

```
WITH Refer(A) as
  SELECT DfrmArea(ie.Data) as A
  FROM ImageElement as ie,
     ImageSet as is
  WHERE Material=X and Time=t and
     Filler=s and is.SetID=ie.SetID
```

ID	SQL Statement	Description
Q6	**SELECT** GetValue(GDist(ie.Data, M1), ie.Data, M2) **FROM** ImageElement **as** ie, ImageSet **as** is **WHERE** Material=X **and** Filler=s **and** is.SetID=ie.SetID;	What is the change in distance between two marked points as the material is pulled?
Q7	**SELECT** DfrmArea(ie.Data) **FROM** ImageElement **as** ie, ImageSet **as** is **WHERE** Material=X **and** Filler=s **and** is.SetID=ie.SetID;	What is the area of deformation for material X with filler concentration $s\%$?
Q8	**SELECT** DfrmArea(ie1.Data)−DfrmArea(ie2.Data) **FROM** ImageElement **as** ie1, ImageElement **as** ie2 **WHERE** ie1.SetID=1 **and** ie2.SetID=2 **and** Time=10;	What is the difference in the area of deformation for different samples at time=10 seconds?

Table 3. Typical Material Science Image Content Queries (Q6-Q8) and Their SQL Statements

```
SELECT Name, Material, Filler
FROM ImageElement as ie, ImageSet as is
WHERE is.SetID=ie.SetID
  and DfrmArea(ie.Data) − Refer.A <0.5;
```

These two queries represent a broad range of complex queries. Q10 may be easily extended to questions that require a spatial join of the raster images, e.g. "Find the pair of materials whose area of deformation correlated well".

4 Implementation and Query Processing

There are three basic methods which can be used to store images in an OO/OR database management systems. One way is to store them in a table that has an image id, (x,y)) coordinates, and the pixel value. This method is not very practical as it wastes space, since an image id and the coordinates must be explicitly stored for each pixel, and it leads to a huge table which usually has to be joined with another table to connect an image with its meta-data. One reason that this approach may still be useful is that it represents images and meta-data in a consistent way and standard database language such as SQL could be adopted in querying content of the image tables. One may want to implement the image tables in an efficient way while still providing end users with a table kind of view. A second method is to store images using the binary large object (BLOB) data type. The primary disadvantage with this is the lack of random access to cells. Finally, some OO/OR database management systems provide multidimensional array types which provide efficient storage with random access (provided the implementation is good).

For the first class of queries (Q1-Q8), the constraints are meta-data based. Traditional query processing and optimization techniques could be adopted well. When the images are implemented as tables, the image content queries require time consuming join operations and indexing on coordinates of cells may be useful. For the BLOB and multi-array based approaches the support for random cell access is critical in improving query performance.

The second class of queries have complex constraints involving both meta-data and derived values of image content. The major performance enhancement techniques include materializing frequently calculated values, e.g. deformation area, and building function based indexes for image content queries with variables, e.g. l mm from crack-tip.

5 Experiment Design and Results

We chose to test queries Q1, Q3, Q5, and Q9 to see how their performance varied with the number of images in a single image set. These four queries were chosen to get a representative sample of the performance of queries with no map algebra functions (Q1) compared to those using the get value operation (Q3), the create map macro (Q5), and the deformation area macro (Q9). Finally query Q10 was tested to see how it performed depending on the number of image sets where each set had 100 images.

The performance was tested on both synthetic and real data. The synthetic data consisted of 100 image sets that each had 100 images. Each image was 200 by 200 pixels. The deformation area of the images for each set was chosen randomly to be between 100 and 1600 pixels. The real data consisted of 26 images of a material as it was being deformed. The material was a polyethylene teraphthalate (PET) nanocomposite with 3% MLS concentration supplied by KOSA [12]. Like the synthetic data, each image was 200 by 200 pixels.

The test environment is PostgreSQL 8.0 running on Windows XP. The machine is a 2.8 GHz Pentium IV with 512 MB of ram. The map algebra operations and user macros were implemented in Java and connected to PostgreSQL using PLJava.

5.1 Results

Importing the data consists of reading the bitmap files for the real data or generating the appropriate synthetic data and storing this data in the database system. We used the Java ImageIO library with the Java Advanced Imaging plug-ins for working with the bitmap images.

For inserting the data the blob representation was the fastest. It took 512.116 seconds (8.54 minutes) to insert the

100 image sets of synthetic data. The array approach took 5994.515 seconds (99.91 minutes). The table approach was much slower than the other two taking 2594.297 seconds (41.57 minutes) to insert 1 data set.

For the synthetic data queries Q1, Q3, Q5, and Q9 were tested to see how the performance varied between the three storage techniques.

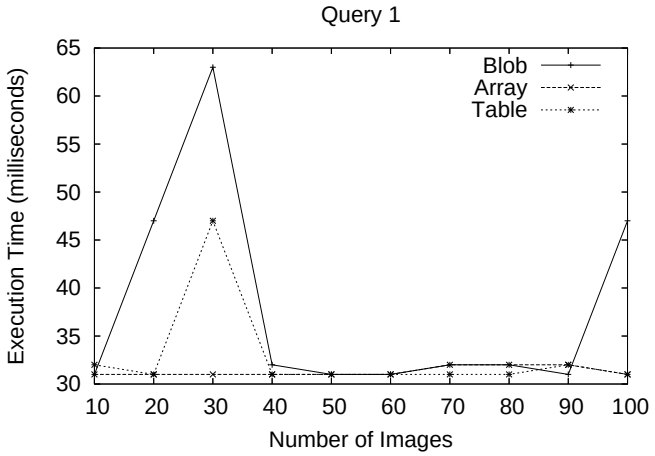


Figure 9. Results for tests on synthetic data with Q1.

Query Q1, shown in figure 9, does not use any map algebra functions and does not access the image data.

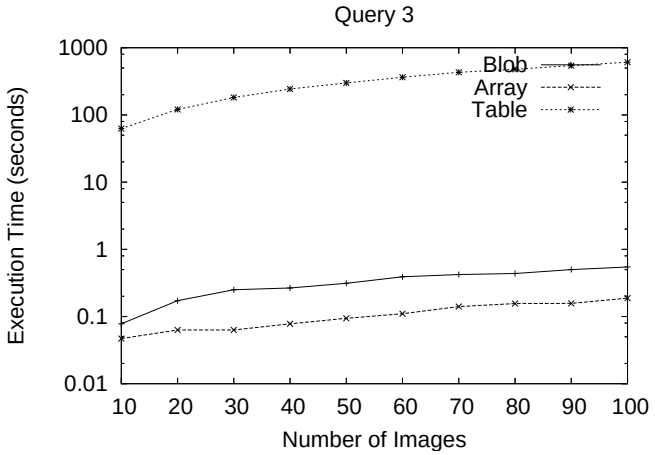


Figure 10. Results for tests on synthetic data with Q3.

It is shown as a reference for the general response time for simple queries under the test environment. Query Q1 could be answered in under 70 milliseconds regardless of the image representation. Note that the times for all of these queries are small so the spikes in the figure for the blob and table based representations were probably the result of other factors on the machine or in the database.

Query Q3, shown in figure 10, tests the time to access a single value from a particular image. The array approach did the best followed by the blob. The blob is approximately 2-3 times slower than the array based method. The table based approach was more than a thousand times slower than both. All methods had a linear growth rate.

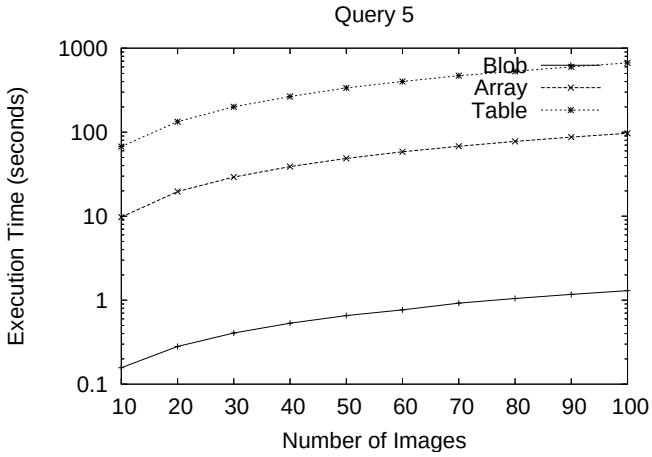


Figure 11. Results for tests on synthetic data with Q5.

Query Q5, shown in figure 11, tests the create map function. The blob approach was about 60 times faster than the array based approach. This was probably due to string processing that has to be done to pass the array from the system to the back-end functions. The table approach was several hundred times slower than the blob approach.

Query Q9, shown in figure 12, tests the deformation area function. The blob approach was about 16 times faster than the array based approach. The array approach was about 4 times faster for this query compared to query Q8 with the same number of images. This speed up is probably due to the deformation area function returning a simple floating point number and the create map returns a raster which for the array approach must be packaged into a string. The table approach was still much slower than the other two.

Query Q10 was also tested for synthetic data using the blob and array representations to see how they performed on a more complicated query comparing image sets. Figure 13 shows the results. The blob approach was considerably faster. With 100 image sets the blob approach took 364.938 seconds (6.08 minutes) where the array approach took 5108.891 seconds (85.14 minutes).

Real data was also tested for queries Q1, Q3, Q5, and Q9 using all of the images. It had roughly the same per-

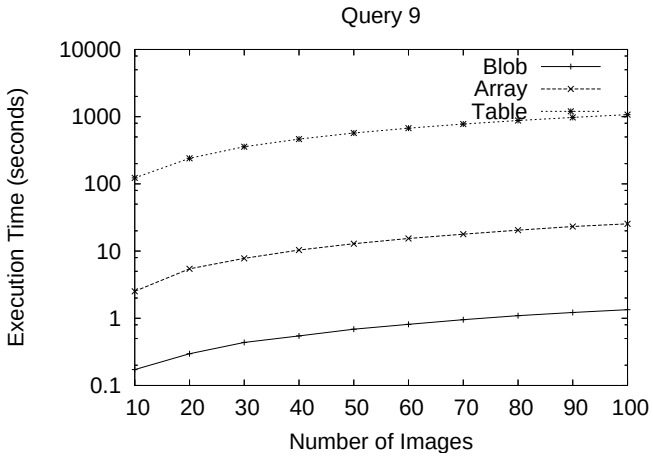


Figure 12. Results for tests on synthetic data with Q9.

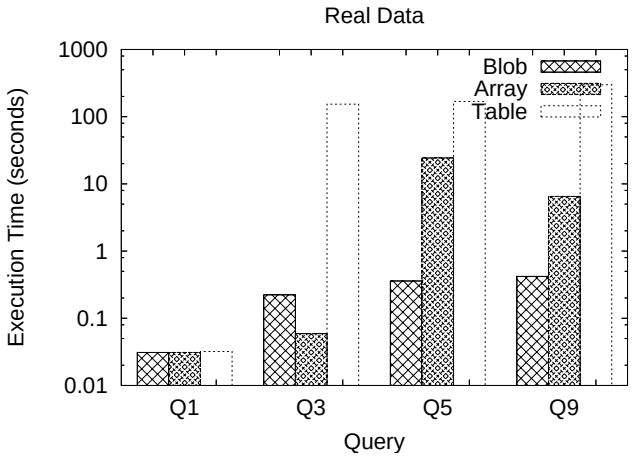


Figure 14. Results for tests on real data with different queries.

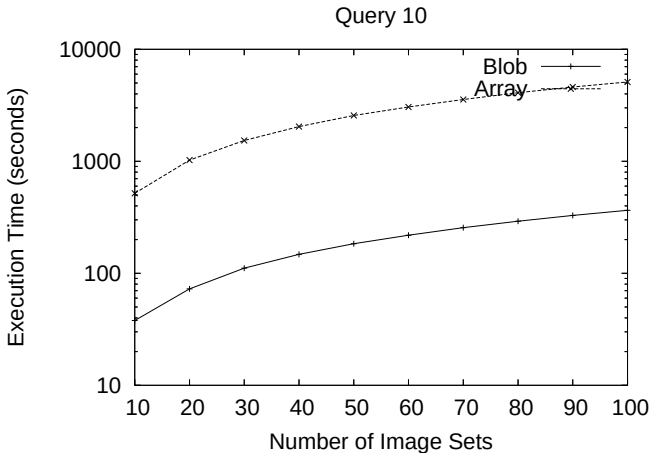


Figure 13. Results for Q10 on synthetic data.

formance as the synthetic data, which was expected since the image sizes and processing requirements were the same. Figure 14 shows the results. It can clearly be seen that the blob and array based representations have much better performance.

6 Conclusion

By integrating map algebra operations with SQL material scientists can now efficiently answer some ad-hoc queries based on meta-data and image content. These queries can be image content queries which return information about image content using conditions based solely on meta-data, or multi-criteria queries which allow for operations based on image content to be part of the conditions.

Of the three ways outlined for storing images in ORDBMS emperical results indicate that using the BLOB or multidimensional array currently offer the best performance with arrays being better for getting a specific value and BLOBs being better when the entire image must be loaded and manipulated as with the create map and deformation area macros. Using a separate relation does not perform well but has the advantage that the images can be manipulated using standard SQL.

Future work should focus on improving the shortcomings of the various ways of representing images, and increasing the integration with SQL to make image content queries less complex. Furthermore, work needs to be done to evaluate the benefit of storing or pre-computing values for commonly used functions or macros such as create map and deformation area. It is also possible that indexes could be built based on the results of these functions to speed up processing at the expense of storage space.

References

[1] P. A. Boncz and M. L. Kersten. Monet: An Impressionist Sketch of an Advanced Database System. In *Proceedings Basque International Workshop on Information Technology*, San Sebastian, Spain, July 1995.

[2] R. Chbeir, S. Atnafu, and L. Brunie. Image data model for an efficient multi-criteria query: A case in medical databases. In *Proceedings of the 14th International Conference on Scientific and Statistical Database Management*, pages 165–174, 2002.

[3] N. A. D'Souza, W. Brostow, L. D. Favro, A. C. Ramamurthy, R. L. Thomas, and Y. Wang. Mechanics of failure of plastics by thermal imaging: Examination of a thermoplas-

tic polypropylene + epdm blend. *Polymer Engineering and Science*, 36:194–202, 1996.

[4] N. A. D'Souza and A. Hernandez-Luna. Characterization of structural changes in polypropylene nanocomposites by infrared thermal wave imaging. In *Annual Technical Conference Society of Plastics Engineers*, page 1342, 2003.

[5] M. J. Egenhofer and T. Bruns. Visual map algebra: A direct-manipulation user interface for gis. In *VDB*, pages 235–253, 1995.

[6] ESRI. Arcinfo. http://www.esri.com/software/arcgis/arcinfo/.

[7] W. I. Grosky. Managing multimedia information in database systems. *Commun. ACM*, 40(12):72–80, 1997.

[8] T. P. G. D. Group. Postgresql documentation. http://www.postgresql.org/docs/7.4/static/index.html.

[9] H. Ishikawa, Y. Yamane, Y. Izumida, and N. Kawato. An object-oriented database system jasmine: Implementation, application, and extension. *IEEE Trans. Knowl. Data Eng.*, 8(2):285–304, 1996.

[10] N. Nes. Image database management system design considerations, algorithms and architecture. Ph.D. dissertation, University of Amsterdam, 2000.

[11] PCRaster. Pcraster environmental software. http://www.pcraster.nl/.

[12] S. Pendse, N. D'Souza, and J. A. Ratto. Deformation of pet nanocomposites.

[13] D. Pullar. Mapscript: A map algebra programming language incorporating neighborhood analysis. *GeoInformatica*, 5(2):145–163, 2001.

[14] G. X. Ritter, J. N. Wilson, and J. L. Davidson. Image algebra: An overview. *Computer Vision, Graphics, and Image Processing*, 49(3):297–331, 1990.

[15] P. M. Schwarz, W. Chang, J. C. Freytag, G. M. Lohman, J. McPherson, C. Mohan, and H. Pirahesh. Extensibility in the starburst database system. In *OODBS*, pages 85–92, 1986.

[16] S. Shekhar and S. Chawla. *Spatial Databases: A Tour*. Prentice Hall, ISBN: 0130174807, 2003.

[17] M. Takayama and H. Couclelis. Map dynamics integrating cellular automata and gis through geo-algebra. *International Journal of Geographical Information Science*, 11(1):73–91, 1997.

[18] D. Tomlin. *Geographic Information Systems and Cartographic Modeling* . Prentice Hall College Div, 1990.

[19] M. Ubell and M. Olson. Embedding image query operations in an object-relational database management system. In *Proc. Storage and Retrieval for Image and Video Databases III*, pages 197–203, 1995.

[20] N. Widmann and P. Baumann. Performance evaluation of multidimensional array storage techniques in databases. In *Proceedings of the 1999 International Database Engineering and Applications Symposium*, 1999.

[21] A. Yoshitaka and T. Ichikawa. A survey on content-based retrieval for multimedia databases. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):81–93, 1999.

[22] A. Zehnder. Temperature rise due to dynamic crack growth. http://people.ccmr.cornell.edu/˜atz/crack_temp.html.

[23] A. Zehnder, K. Bawa-Bhalla, R. Thomas, and L.D.Favro. Thermal analysis of crack tearing. http://people.ccmr.cornell.edu/ atz/crack_temp.html.