HPDRC
00-VA

PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON IMAGING SCIENCE, SYSTEMS, AND TECHNOLOGY

CISST'2000

Volume II

Editor:
Hamid R. Arabnia

Associate Editors:
F-X. Coudoux, Y. Mun, G. J. Power, M. Sarfraz, Q. Zhu

Las Vegas, Nevada, USA
June 26 - 29, 2000
©CSREA Press

Variation Assessment of Large Entities Using Spatial Data*

Nagarajan Prabakar, Qing Jin and Naphtali Rishe
High Performance Database Research Center (HPDRC)
School of Computer Science, Florida International University
University Park, Miami, FL 33199

Abstract

This paper focuses on the design and implementation of a spatial image processing system, which is used to extract a desired 2-D object region from a given image, calculate its area and compare it with other object regions. A linear time labeling algorithm has been designed to extract an object region based on user-specified properties. Then, we implemented a special function that allows polygon boundary selection from the extracted object region. To overcome the limitations of the recursive pixel analysis (floodfill) algorithm, we also designed an edge detection algorithm for image segmentation. Finally, to compare two binary objects that are extracted from spatial images, we introduced a technique to calculate the maximum overlap position of the two objects so that the areas of common and different regions could be computed.

Keywords: spatial data, edge detection, image segmentation, image comparison

1. Introduction

Spatial data, which are usually collected by satellites and airplanes, have recently gained substantial importance. From spatial images, we can analyze the available water resources (e.g., area of lakes, width of canals, etc.) for any specific region, assess the impact of a hurricane, earthquake, or tornado on a specific region, and compute the erosion of islands or coastal areas. With high-resolution spatial data from airborne lasers, insurance companies can assess property damages caused by extreme events quickly and accurately.

The proposed application is intended to process spatial data retrieved from the TerraFly system [9]. TerraFly is a raster based spatial data visualization software system developed at the High Performance Database Research Center (HPDRC). The TerraFly system retrieves data from a high-performance Semantic multimedia spatial database [8] and allows users to "fly" over data in real-time. The goal of this project is to expand TerraFly's capabilities by providing a separate module, which can be used to compare the objects in spatial data. Thus, our research focuses on image enhancement and comparison of image objects.

In short, our application first compares image objects based on their shapes and then, if necessary, compares their other properties. By comparing the image shapes of two objects, we can find a global optimal position to maximize the overlap between the two images. Subsequently, we calculate the common area and difference between the image objects. The structure of the entire system is shown in Figure 1. The next section outlines the extraction of image objects from TerraFly. Section 3 describes the polygon boundary selection for an image object. The comparison of image objects and the results are presented in Section 4. The final section presents our conclusions.

* This research was supported in part by NASA (under grants NAGW-4080, NAG5-5095, NAS5-97222, and NAG5-6830), NSF (CDA-9711582, IRI-9409661, HRD-9707076, and ANI-9876409), ARO (DAAH04-96-1-0049 and DAAH04-96-1-0278), AFRL (F30602-98-C-0037), BMDO (F49620-98-1-0130 and DAAH04-0024) DoI (CA-5280-4-9044), and State of Florida.

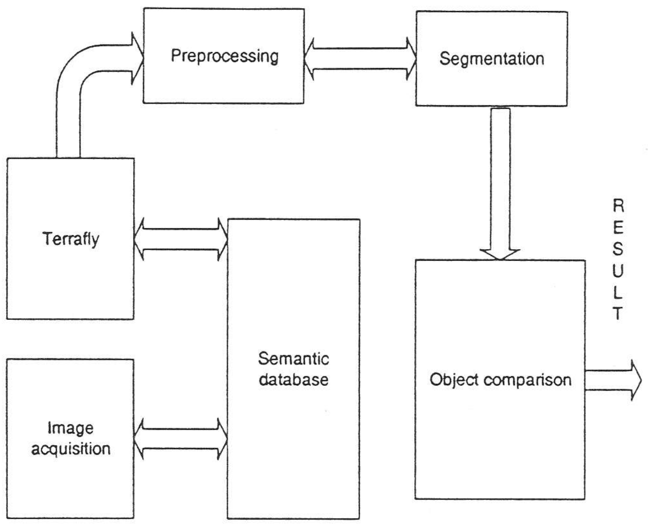

Figure 1. Structure of the variation assessment

2. User Region Selection

We have developed an interactive GUI (Graphical User Interface) tool that allows the user to select a region of interest (ROI) as a polygon over the image. Normally, this ROI includes the intended image object and pixels adjacent to it. In most cases, we are just interested in a specific area of a large spatial image. Selecting the region of interest and then processing this part of the image can minimize computational time and improve the overall response time. Although selection of a rectangular sub-region of an image is simple, it is not suitable for many applications which deal with irregular regions like curved pathways, canals, bay areas, etc.

Once the user provides an outline of a desired ROI on the spatial data image by specifying a closed polygon, the polygon can be resized by stretching or contracting any vertex of the polygon interactively. For example, if we are interested in a lake that is connected to several rivers, a ROI specification with a polygon will correctly exclude the rivers from the lake. From the polygon region, a binary mask is created for the ROI. Subsequently, a standard scan conversion algorithm [3] is applied to extract the ROI from the image according to the polygonal area on the binary mask.

3. Image Segmentation

To improve accuracy, pixels in the adjacent area have to be removed by image segmentation. There are several approaches to image segmentation [5, 7]. Most of them focus on finding specific features in an image such as points, lines, edges, etc., and then partitioning the image using these features [1, 2]. In this section, we describe another image segmentation approach called *image growing*. Image growing is a procedure that groups pixels or sub-regions into a larger region.

3.1 Image Growing Approach

Initially, the user can select a seed pixel within the user-specified polygon that satisfies the image region pixel characteristics. From the seed pixel, all adjacent connected pixels within the polygon that are similar to the seed pixel can be identified using a standard floodfill recursive algorithm [4]. As this algorithm is computationally expensive due to its recursive nature, we implemented a linear-time labeling algorithm that generates the same image region.

The basic idea of the labeling algorithm is to scan an image pixel by pixel, from left to right and from top to bottom, and assign a label to each pixel. The label assignment for each pixel employs the following rules:

1. If a pixel doesn't have the required property (i.e., that of the seed pixel), assign "–1" to its label .
2. If a pixel has the required property, check all the left and top neighbor pixels (these pixels already have labels). If **all** these labels are –1, assign a new label (current highest label number + 1) to this pixel. If any of the neighbor labels is positive (i.e., not –1), then assign the label of that neighbor to this pixel. If the neighbor labels are positive and have different label values, make a note of the label equivalencies (the neighbor with the high

label value is assigned with the low label value) for all these positive labels.

We use a label array to save the equivalent labels information. Initially, the label value for each element of the array is same as its array index. The labels are merged with successive nested merge operations in linear time [6] as described below. This merging is done in one pass (starting from the lowest index) with the following criterion:

If Label[i] ≠ i, we assign the value of Label[Label[i]] to Label[i]. For instance, if we have pixel values of a label array as shown in Table 1, the merged label array will contain the values as in Table 2:

After resolving label equivalences for all pixels of the image, find the label of the seed pixel and all array elements with this label. All pixels with this label value will form the desired region.

A binary mask for the region and for each pixel is built, and its binary value is set to 1 if it is in the region or 0 otherwise. With this binary mask, we can calculate the area of this region by counting all the pixels that are not zero. If we know the scale of the image, we can estimate the realistic area of this region. The labeling algorithm is applied to the image shown in Figure 2 and the result is presented in Figure 3.

Table 1. Pixel values of a label array before merging

Index	...	3	4	5	6	7	8	9	...	19	...	25
Label	...	3	3	3	6	4	5	7	...	9	...	8

Table 2. Pixel values of the merged label array

Index	...	3	4	5	6	7	8	9	...	19	...	25
Value	...	3	3	3	6	3	3	3	...	3	...	3

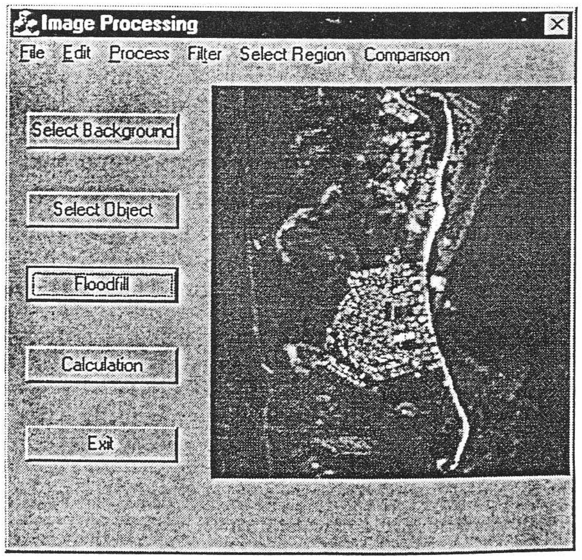

Figure 2. Image of an Island (before labeling algorithm)

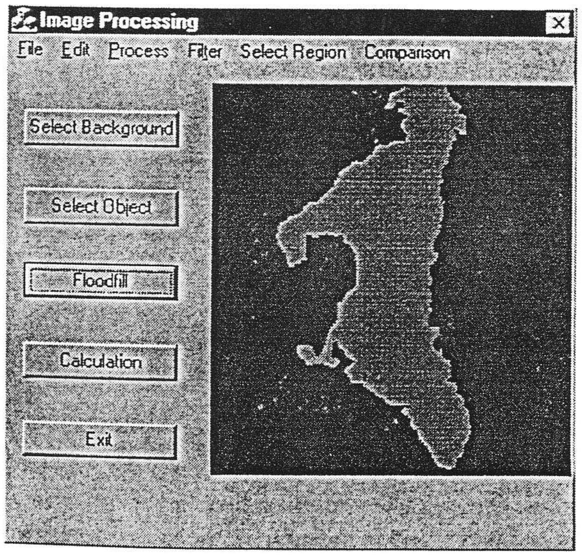

Figure 3. Binary Image of the Island (after labeling algorithm)

4. Object Comparison

Comparison of two regions from two different images is a complicated task because the two regions will differ in shape, color and intensity. Since the main purpose is to quantify the variation between two similar image objects, we focus on comparing the shape of objects. One simple approach is to overlay the regions and calculate the overlapping area between them. Although this idea is straightforward, there are several possible ways of overlaying the regions with different combinations of position and orientation. This is computationally very expensive.

4.1. Global Optimization Problem

To compare two objects, we introduced an incremental strategy in which the first object is fixed and we gradually move and rotate the second object to find the maximum overlap between the two objects. We select one reference point from each object and use the function f (x, y, θ) to represent the overlap between two objects, where (x, y) are the coordinates of the second object's reference point and (0,0) are the coordinates of the first object's reference point. The angle θ is the angle of rotation of the second object with respect to the reference point of the first object. The problem of comparing two objects is to find the combination of parameters that maximize the function f.

Because the maximum of a function f is a minimum of –f, we can apply general mathematical optimization theory to this problem. The definition of the global optimum $\mathbf{X}^*$ of f ($\mathbf{X}$) is that

$$f(\mathbf{X}^*) < f(\mathbf{Y}) \qquad \forall\, \mathbf{Y} \in V(\mathbf{X}),\ \mathbf{Y} \neq \mathbf{X}^*$$

where V($\mathbf{X}$) is the set of feasible values of the control variables $\mathbf{X}$, which is a vector.

A point Y* is a local minimum of f (x) if

$$f(\mathbf{Y}^*) < f(\mathbf{Y}) \qquad \forall\, \mathbf{Y} \in N(\mathbf{Y}^*,\ \eta),\ \mathbf{Y} \neq \mathbf{Y}^*$$

where N($\mathbf{Y}^*$, η) is defined as the set of feasible points contained in the neighborhood of Y*, i.e., within some arbitrarily small distance η of Y*.

4.2. Approaches to Comparing Two Objects

From the above analysis, we can first seek different local minimums (maximums) to find a global minimum (maximum). Then, by comparing these local minimums (maximums), we can get a global minimum (maximum).

There are several deterministic algorithms for finding the local minimum of multivariate functions whose arguments are continuous and on which no restrictions are imposed. Global minimum is an entirely different and more challenging problem. Stochastic methods can be used for the global optimization.

Our approach is different from these algorithms for two reasons:

1. Our application needs real time response, so the global optimization should be obtained in a reasonable time. Calculating the global optimization with these algorithms takes too much time.
2. The global optimization problem here has some different properties. The function changes slowly here and the local region of the global maximum will be not trivially small.

In our approach, the two main steps are as follows:

1. Find the region that contains the global optimal result. We will divide the image into many sub-regions. We will calculate one overlap result with each sub-region and find the maximum within these results. We believe this sub-region will be the region that contains the global optimum if we get the appropriate size of sub-region. The result of this step is shown in Figure 4.
2. If the region obtained in step 1is small, we just need to find a local maximum within the sub-region and regard it as the global maximum. If the region is still large, repeat step 1 until the region is small (small enough to have one local maximum or it is possible to calculate all values in this sub-region). The

global optimum result for this example is presented in Figure 5.

With this algorithm, cost of finding the global maximum will be reduced to a small percentage of the original cost.

Another approach is to ask the user to select two reference points from both objects. If the two objects we compare are similar, the user can select two similar positions in the two images. This means that the user can find the special region – the local optimization of this region will be the global optimum.

When we try to put one object on top of another, we can first put the two reference points together, and then calculate the overlapping area as below.

1. Start with one reference point and move the top image with small increments in the x direction and find a position with the maximum value of overlap area.
2. Start from the maximum point obtained in step one and move the top image within a small range in the y direction to find a maximum overlap again.
3. Repeat steps one and two several times, keeping the range of movement small. Then if there are no any further changes of maximum overlap area, we stop and take this final position as global optimum.

With this approach, if the global optimization is also the starting reference point's local maximum, the result will be the best overlap of the two objects. If global optimization is far from the reference point, by increasing the range in steps 1 and 2, we always can find a global maximum position.

The first approach is an auto-comparison algorithm. Using this algorithm we can efficiently find the global maximum. The second approach selects the reference point and only calculates the local maximum position that works more efficiently than the first approach. Because in most cases the reference point selected by users is close to the global maximum overlapping position, the computer will efficiently find the optimum position in the local region. Figure 4 and Figure 5 show the result of comparing two binary objects and the maximum overlap of these two objects.

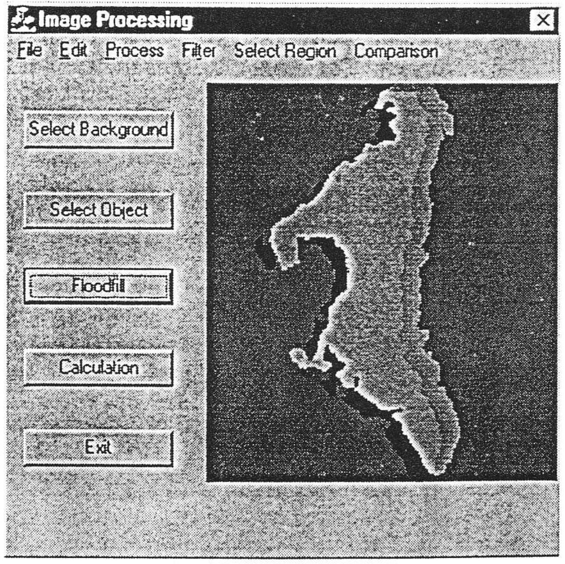

Figure 4. Overlap of two binary objects before optimization. Light blue region is the common part between two objects. Green and dark blue represent separate objects.

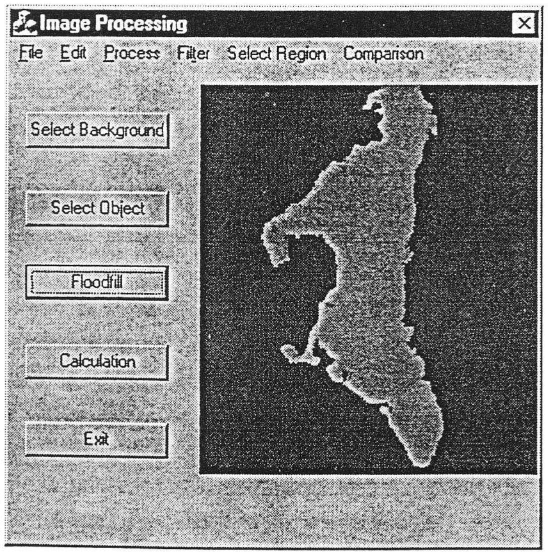

Figure 5. Overlap of two binary objects after optimization. Light blue is the common part between two objects. Green and dark blue represent separate objects.

5. Conclusion

We presented a tool to automatically recognize objects from images and to compute the quantitative difference between image objects. A special application module is implemented that allows polygon boundary selection from an image object with similar properties. This makes the extraction of an object from an image more efficient. Also, a region selection algorithm with a labeling technique is designed to extract an object region based on user chosen properties. Since the cost of this algorithm is proportional to the number of pixels in the input image, it minimizes the processing time for large types of data such as spatial data.

To compare two given objects extracted from spatial images, a feasible approach has been developed and implemented to calculate the maximum overlap position of the two objects so that the areas of common and the difference could be computed. This software has been implemented in Visual C++ and Java.

References

1. Berzins, V. "Accuracy of Laplacian edge detection," *Computer Vision, Graphics, Image Processing*, Vol. 27, No. 2, pp. 195-210, 1984.
2. Canny, J. "A Computational Approach to Edge Detection", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 6, pp. 679-698 Nov. 1986.
3. Foley, J.D., van Dam, A., Feiner, S.K., Hughes, J.F. and Phillips, R.L. Computer Graphics: Principles and Practice, Second Edition in C, Addison-Wesley, 1996.
4. Gonzalez R.C. and Richard E.Woods: "Digital Image Processing", Addison-Wesley, pp. 161-249 and 413-482, 1992.
5. Huertas, A. and Medioni, G. "Detection of Intensity Changes with Subpixel Accuracy Using Laplacian-Gussian Masks", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 5, pp. 651-664, 1986.
6. Jin, Q. Variation Assessment of Large Entities Using Spatial Data, Master's Thesis, August 1999.
7. Mortensen, E.N. and Barrett, W.A. "Interactive Segmentation with Intelligent Scissors", *Graphical Models and Image Processing*, pp. 349-384, Article No. IP980480, 1998.
8. Rishe, N. "Database Design: The Semantic modeling Approach", McGraw-Hill, 1992.
9. Terrafly, version 1.0 CD-ROM, High Performance Database Research Center (http://hpdrc.cs.fiu.edu/), Florida International University, 1998.

Volume I, ISBN: 1-892512-53-x
Volume II, ISBN: 1-892512-54-8
Set, ISBN: 1-892512-55-6