Replicated Shared Object Model for Edge Detection with Spiral Architecture

Xiangjian He, Tom Hintz and Ury Szewcow

School of Computing Sciences
University of Technology, Sydney
PO Box 123, Broadway 2007, Australia

Abstract. Edge detection in computer vision and image processing is a process which detects one kind of significant features appearing as discontinuities in intensities. A parallel edge detection algorithm based on Spiral Architeture is designed in this paper by a Replicated Shared Object Model (RSOM). We use this approach to meet the need of replicated object management, update propagation, underlying communication and consistency maintenance among replicated objects.

1 Introduction

Edge detection plays a key role in the areas of computer vision and image processing etc. It is a time-consuming process, which detects the significant features appearing as discontinuities in light intensities. As an early stage of computation in a large scale computer vision application, an edge map is constructed from its original image to contain major image information which only needs a relatively small amount of memory space to store. If needed, a replica image can be easily constructed from its edge map.

In the past, there have been various edge detection algorithms proposed (e.g. [1], [2], [3] and [4]). One important issue in edge detection is to improve the computation efficiency. The detection is very computationally intensive. The computation is conducted pixel by pixel, and several dozens of arithmetic operations are performed for each pixel. Zhang and Deng [3] show that edge detection using *edge focusing* method (proposed by Bergholm [1]) requires multiple processing iterations from a coarse level to fine levels of resolutions. Their experimental result tells us that an edge detection for a medium size image (of 512×512 pixels processed by 10 iterations) required about 2,400 seconds of computation time on a SPARC station. Even a 40 MHz Intel 860 processor needed 221 seconds to complete. So, fast processing response is a major requirement in many image processing applications, such as in real-time processing where a sequence of image frames should be processed in a very short time. This suggests algorithms on parallel computers. He, Hintz and Szewcow [5] proposed a parallel algorithm based on the Spiral Honeycomb Lie Algebra (SHLA).

SHLA described by Sheridan [6] is a new data structure for computer vision. Image is represented by a collection of hexagons of the same sizes (in contrast with the traditional rectangular representation). The importance of the hexagonal representation is that it possesses special computational features that are pertinent to the vision process. To determine that a pixel is a pixel on the edge, the collection of the information from the neighbouring six pixels are necessary. This leads to necessary communition among processsors handling the pixels or clusters of seven pixels. Therefore, a peer-to-peer model (PTPM) is appropriate for this application. Though the PTPM can be implemented on a client-server style object model, Replicated Shared Object Model (RSOM) is suitable because of the fundamental similarity of structure between PTPM and RSOM.

RSOM is an object model that handles distributed objects in a replicated way. Each application which shares a logical object has a copy of the same object [7]. If the copy of a logical object is updated in an application, all other copies are automatically updated in each application. RSOM deals with replicated object management, update propagation, underlying communition and consistency maintenance among replicated objects.

In this paper, we construct the RSOM for the edge detection on parallel computers.

2 Spiral architecture

An image may be considered as the collection of pixels (picture elements). These elements correspond to the position of the photo receiving cells of the image capturing device. In the case of the human eye, these elements would represent the relative position of the rods and cones on the retina. The geometric arrangement of cones on the primate's retina can be described in terms of a hexagonal grid. This leads to the consideration of an image as the collection of hexagonal cells as displayed in Figure 1.

Each of the individual hexagons is labelled with a unique address as decribed in [6]. This is achieved by describing a process that begins with a collection of seven hexagons. Each of these seven hexagons is labelled consecutively with addresses 0, 1, 2, 3, 4, 5 and 6 as displayed in Figure 2.

Dilate the structure so that six additional collections of seven hexagons can be placed about the addressed hexagons, and multiply each address by 10 (see Figure 3).

For each new collection of seven hexagons, label each of the hexagons consecutively from the centre address as we did for the first seven hexagons (see Figure 4).

The repetition of the above steps permits the collection of hexagons to grow in powers of seven with uniquely assigned addresses. It is this pattern of growth that generates the Spiral. Furthermore, the addresses are consecutive in base seven.

The importance of the Spiral arrangement, apart from its intriguingly beautiful geometry, is that it possesses powerful computational advantages to computer vision [8].

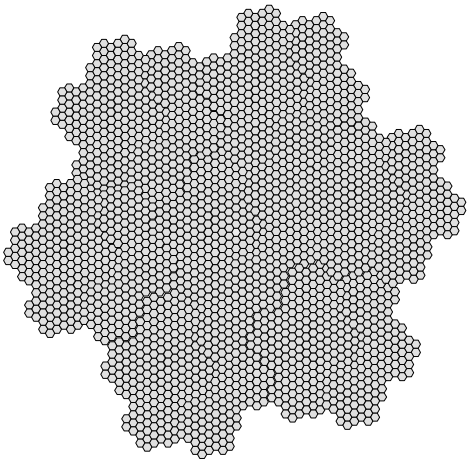

Figure 1: Collection of hexagonal cells.

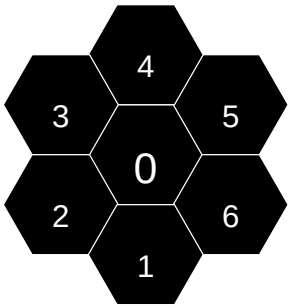

Figure 2: A collection of seven hexagons.

3 Edge Detection Algorithm

Let $f : \Re^2 \to \Re$ be a signal of an image. The scale-space representation $L : \Re^2 \times \Re^+ \to \Re$ is defined such that the representation at 'zero scale' is equal to the original signal, i.e.,

$$L(\cdot; 0) = f(\cdot), \tag{3.1}$$

and the representation at 'coarser scales' is the convolution

$$L(\cdot; t) = g(\cdot; t) * f(\cdot),$$

where $g : \Re^2 \times (\Re^+ - \{0\}) \to \Re$ is the Gaussian kernel [9].

A natural way to define edges from a continuous grey-level image $L : \Re^2 \times \Re^+ \to \Re$ is as the set of points for which the gradient magnitude assumes a maximum in the gradient direction [9]. To give a differential definition of this

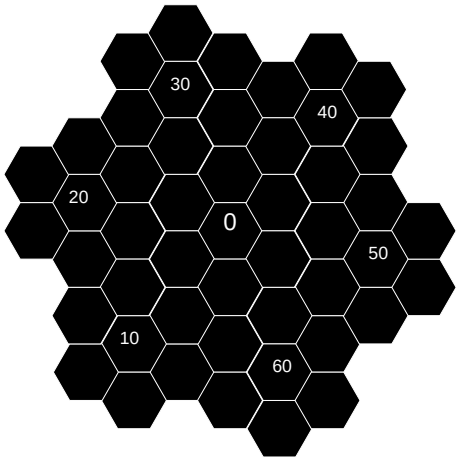

Figure 3: Dilated figure to fit more hexagons.

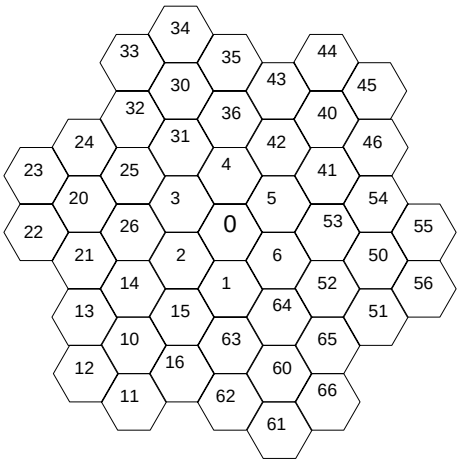

Figure 4: Collection of $7^2 = 49$ hexagons with labelled addresses.

concept, introduce a curvilinear coordinate system (u, v), such that at every point the v-direction is parallel to the gradient direction of L, and at every point the u-direction is perpendicular to the v-direction. Moreover, at any point $P = (x, y) \in \Re^2$, let $\partial_{\bar{v}}$ denote the directional derivative operator in the gradient direction of L at P and $\partial_{\bar{u}}$ the directional derivative operator in the perpendicular direction. Then at P the gradient magnitude is equal to $\partial_{\bar{v}} L$, denoted by $L_{\bar{v}}$, at that point. Assuming that the second and third order directional derivatives of L in the v-direction are not simultaneously zero, the condition for P_0 to be a gradient maximum in the gradient direction may be

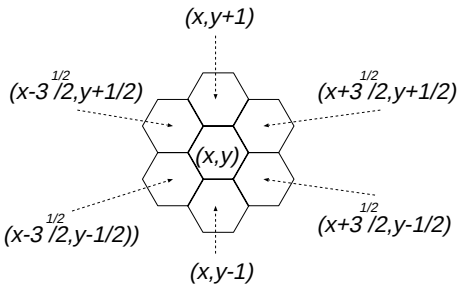

Figure 5: Ordinary coordinates of a cluster of 7 hexagons.

stated as

$$L_{\bar{v}\bar{v}} = 0 \quad \text{and} \quad L_{\bar{v}\bar{v}\bar{v}} < 0. \tag{3.2}$$

Note that

$$\begin{aligned} \partial_{\bar{u}} &= (\sin\beta)\partial_x - (\cos\beta)\partial_y, \\ \partial_{\bar{v}} &= (\cos\beta)\partial_x + (\sin\beta)\partial_y, \end{aligned}$$

where $(\cos\beta, \sin\beta)$ is the normalized gradient direction of L at P_0. Hence, (3.2) is equivalent to

$$\begin{aligned} \tilde{L}_{\bar{v}\bar{v}} &= L_{\bar{v}}^2 L_{\bar{v}\bar{v}} = L_x^2 L_{xx} + 2L_x L_y L_{xy} + L_y^2 L_{yy} = 0, \\ \tilde{L}_{\bar{v}\bar{v}\bar{v}} &= L_{\bar{v}}^2 L_{\bar{v}\bar{v}\bar{v}} = L_x^3 L_{xxx} + 3L_x^2 L_y L_{xxy} + 3L_x L_y^2 L_{xyy} + L_y^3 L_{yyy} < 0. \end{aligned}$$

Given discrete data, note that the six neighbouring hexagons of the hexagon at (x, y) have ordinary coordinates $(x, y-1)$, $(x-\sqrt{3}/2, y-1/2)$, $(x-\sqrt{3}/2, y+1/2)$, $(x, y+1)$, $(x+\sqrt{3}/2, y+1/2)$ and $(x+\sqrt{3}/2, y-1/2)$ (Figure 5).

The derivatives operators can then be obtained in finite difference form:

$$\begin{aligned} L_x(x,y;t) &= \frac{1}{\sqrt{3}}[L(x+\sqrt{3}/2, y+1/2; t) + L(x+\sqrt{3}/2, y-1/2; t)] \\ &\quad -\frac{1}{\sqrt{3}}[L(x-\sqrt{3}/2, y+1/2; t) + L(x-\sqrt{3}/2, y-1/2; t)] \end{aligned}$$

and

$$\begin{aligned} & L_y(x,y;t) \\ = \; & \frac{1}{2}[L(x+\sqrt{3}/2, y+1/2; t) + L(x-\sqrt{3}/2, y+1/2; t) + L(x, y+1; t)] \\ & -\frac{1}{2}[L(x+\sqrt{3}/2, y-1/2; t) L(x-\sqrt{3}/2, y-1/2; t) + L(x, y-1; t)]. \end{aligned}$$

The second and third order derivatives can be obtained respectively from the first and second order derivatives in the same way.

An edge detection algorithm using the *edge focusing technique* can now be outlined as follows [3]:

1. An edge map is defined as a binary image represented by 0's and 1's with a resolution parameter t, denoted by $E(x, y; t)$. $E(x, y; t) = 1$ if the pixel (x, y) is an edge point, otherwise $E(x, y; t) = 0$.
2. Create an initial coarse-level edge map $E(x, y; t_0)$, using an edge detector based on the Gaussian convolution and the differential representation decribed above in this section.
3. For $i = 1, 2, \cdots$, create the ith-level edge map $E(x, y; t_i)$, by using the same blurring technique, where $t_i = t_{i-1} - \Delta t$. The resolution parameter, t is decreased by Δt at each blurring step. Note that the Gaussian operator only acts on the edge points, i.e., $\{(x, y)|E(x, y; t_{i-1}) = 1\}$ detected in the previous iteration, and their neighbour areas.
4. The edge detection iteration continues until the blurring scale t_i is too small to blur the image.

There are the following important features in the above algorithm.

1. The Gaussian operator removes noise and unnecessary detail at a high expense because the convolution is a complicated computing processing which needs a large number of arithmetic operations.
2. Decreasing t_i to obtain a sharper and clearer edge map is an iterative facusing process.
3. Gaussian convolutions and derivatives etc. are conducted pixel by pixel, the entire edge focusing process is very time-consuming.
4. On the same level of resolution, no operations acting on one pixel will affect operations on the other pixels.
5. On the same level, the calculation of higher order derivative (e.g., 1st order derivative) on a pixel requires lower order derivatives (e.g., 0-th order derivatives) of surrounding six pixels.

The first four features suggest a parallel implementation of the algorithm, which has been proposed in [5]. The last one implies the use of Replicated Shared Objects when we treat each pixel as an object.

4 Class Definition

The C++ class mechanism [10] allows us to define a class of objects (i.e., class of pixels in this paper) as follows.

```
double t;   //resolution parameter, global variable
...
class pixel {
public:
   double L();   //grey-level image
   double *L_1();   //L_1 points to L_x
   double **L_2();   //L_2 points to L_xx
   double E();   //edge map of L
```

```
   ...   //definitions of L, *L_1, **L_2 and E
};
```

To prevent the member functions of an class object being called directly from outside of the object, an interface function is used. The interface function packs or unpacks a call message. Any call to or from an outside object can only be done through interface functions. By separating the member functions of an object from their interface function, the underlying communication activity can be hidden from programmer and the repeated calling problem between the same objects can be avoided. In the case of this paper, the interface function of a pixel (object) is used to call its surrounding six pixels (objects) for their values of E, L and its derivatives. This suggests a new definition of the class as follows.

```
double t;   //resolution parameter, global variable
...
class Pixel {
private:
   double L();   //grey-level image
   double *L_1();   //L_1 points to L_x
   double **L_2();   //L_2 points to L_xx
   double E();   //edge map of L
   ...   //definitions of L, *L_1, **L_2 and E
public:
   void get();   //get info. of surrounding 6 pixels,
         //interface function
   ...   //definition of get()
};
```

Each pixel (or hexagon) with address i (an integer of mode 7) corresponds to a class object declared as

```
   Pixel pixel_i;
```

5 Replicated Shared Object Model

We are now able to construct the RSOM for the parallel algorithm of edge detection.

The outline of the parallel algorithm of edge detection using RSOM is as follows:

1. The master node assigns one or more equally partitioned image subregions of an initial image data file to each corresponding slave node. The partitioned image is a collection of Pixel class objects.
2. Include the surrounding six pixels (or objects) of each pixel into the slave node if they are not already in the node. These newly included objects are shared by at least two objects in different slave nodes. This means they are replicated shared objects.

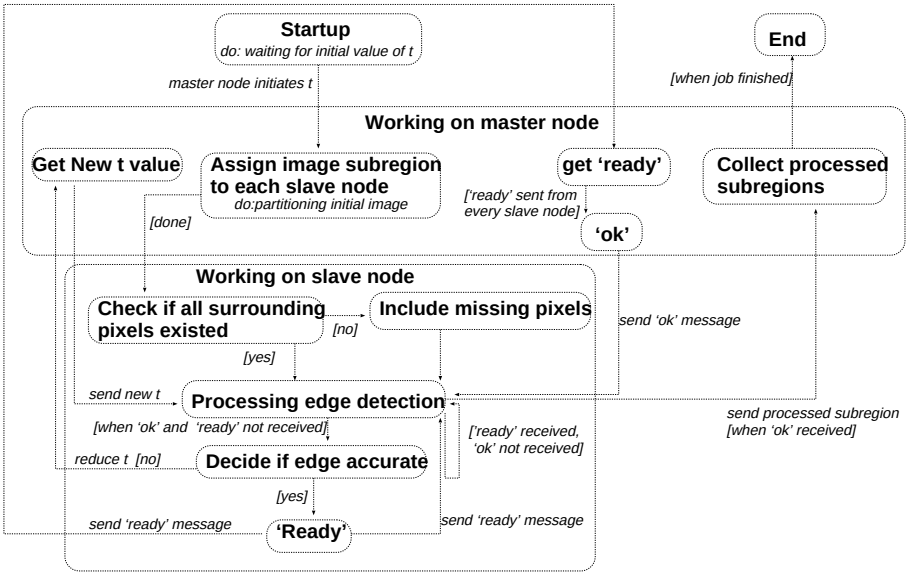

Figure 6: Replicated shared object model.

3. Each slave node performs edge detection in the image subregions by Gaussian operator.
4. Each slave node decides if the edge points of the corresponding subregion are sufficiently accurate and clear. If yes, issues 'ready' message to master node and goes to step 5. Otherwise, decreases the global parameter t and the computation iterates back to Step 3.
5. Repeatedly does Step 3 with any smaller value of the parameter t until receives 'ok' message from the master node, which is the case shown in the nexe step.
6. When the master node has received 'ready' message from all slave nodes, it issues 'ok' message to all slave nodes.
7. The master node collects the processed subregions from all slave nodes, and updates the entire image data file. Then, the edge detection precess completes.

A dynamic model [11] of RSOM is shown in Figure 6.

6 Discussion

The fact that different nodes sharing the same objects have replicated copies of the objects make it unnecessary to maintain the shared objects consistently. This is because the replicated objects do not have to be kept in the same state though all objects in the same node are assigned the same value of the parameter

t at each resolution level. This feature improves greatly the efficiency of the edge detection.

To increase the processing speed, each slave node can delete its object (or pixel) when the surrounding six pixels are not edge points at a resolution level. Each slave node must also creat the object of which one of surrounding pixels is edge point and belongs to the subregion corresponding to the node.

7 Conclusion

In this paper, we have constructed RSOM for an efficient parallel algorithm of edge detection. This has greatly improved the efficiency of detection process.

References

1. F. Bergholm, "Edge facusing", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No.6, pp.726-741, 1987.
2. J. F. Canny, "A computational approach to edge detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, pp.679-698, 1986.
3. X. Zhang and H. Deng, "Distributed image edge detection methods and performance", *Proc. 6th IEEE Symposium on Parallel and Distributed Processing*, IEEE CS Press, October 1994.
4. X. He and T. Hintz, "Application of Spiral Architecture to edge detection for object recognition", *Proc. Pan-Sydney Workshop on Visual Information Processing*, University of Sydney, Australia, pp.90-95, November 1996.
5. X. He, T. Hintz and U. Szewcow, "Parallel Algorithm on Edge Detection with Spiral Architecture", submitted.
6. P. Sheridan, *Sprial Architecture for Machine Vision*, Ph.D Thesis, University of Technology, Sydney, Australia, 1996.
7. T Lee and C. H. Park, "Management of Replicated Shared Object for Distributed Collaborative Applications", *Proc. Interbational Conference on Parallel and Distributed Processing Techniques and Applications*, Sunnyvale, California, pp.199-202, August 1996.
8. P. Sheridan, T. Hintz and D. Alexander, "Geometric invariance on a hexagonal grid", to appear.
9. T. Lindeberg, *Scape-Space Theory in Computer Vision*, Kluwer Academic Publishers, London, 1994.
10. S. B. Lippman, *C++ Primer*, Addison-Wesley Publishing Company, Sydney, 1995.
11. B. Henderson-Sellers, *A Book of Object-Oriented Knowledge, Printice Hal, 1992.*