Available online at www.sciencedirect.com
SCIENCE DIRECT
Image and Vision Computing 25 (2007) 95–102
image AND vision COMPUTING
www.elsevier.com/locate/imavis

A parallel application for 3D reconstruction of coronal loops using image morphing

Lilian N. Faria [a,*], Nelson D.A. Mascarenhas [a], Célio E. Morón [a], José H. Saito [a], Reinaldo R. Rosa [b], Hanumant S. Sawant [c]

[a] Departamento de Computação, Universidade Federal de São Carlos—UFSCar, C.P. 676 CEP 13565-905 São Carlos, SP, Brazil
[b] Divisão de Astrofísica—DAS, Laboratório Associado de Computação e Matemática Aplicada—LAC/INPE
[c] Instituto Nacional de Pesquisas Espaciais—DAS/INPE, C.P. 515 CEP 12201-970 São José dos Campos, SP, Brazil

Received 26 May 2004; received in revised form 29 July 2005; accepted 18 December 2005

Abstract

This paper presents an approach for 3D reconstruction of X-ray tomographic images of coronal loops, observed on the solar atmosphere by the Japanese satellite Yohkoh. In this approach, the intermediate cross-sections images of the magnetic loop are generated with image morphing controlled by a Bezier curve in arc shape. Due to the high computational costs involved in the processing and visualization of solar images, a parallel application for 3D reconstruction of a coronal loop was implemented to execute in the Atlas parallel system.

Keywords: Coronal loops; 3D Reconstruction; Image processing; Image morphing; Parallel system

1. Introduction

Solar images with high spatial and temporal resolution, captured by satellites equipped with X-ray telescopes, show that the dynamic structures of plasma observed on the active regions of the solar corona—the outermost layer of the solar atmosphere—are sources of intense X-rays emissions [16,17]. Among these structures, the coronal loops are associated with closed magnetic field lines that connect magnetic regions on the solar surface, generating a plasma flow inside a tube with arc shape. The coronal loops can be isolated, as shown in Fig. 1, or organized in a more complex system of loops. Isolated loops come in a variety of shapes and sizes, spanning thousand of kilometers in the solar corona.

Frequently, X-ray images of coronal loops show twisted structures with sigmoidal shape (S or inverted S shape), as shown in Fig. 2. This occurs due to a continuous motion of fluid rotation at both ends of the loop that generates torsional waves that propagate along the tube.

Scientists at NASA found out that when the coronal loop becomes unstable after several rotations, it is quite probably to occur a solar explosion. This explosion (known as solar flare) is a sudden release of great amount of stored magnetic energy in the solar corona, which can be observed as an intense and sudden brightness in intensity of X-rays (Fig. 3). The solar explosions may cause serious perturbations in terrestrial communication systems, damage satellites and energy transportation system, as well as the emission of extremely dangerous radiation levels for astronauts in space missions.

Researchers recognize the importance of the study of the coronal loops dynamics to forecast the solar explosions. At the National Institute for Space Research (INPE), scientists have been developing a detailed study of the spatio-temporal dynamics of the coronal loops [11,12]. However, this study is performed with bi-dimensional images without any information about the 3D structure of the loops. In this way, physical and geometric parameters of the magnetic loop are subject to errors due to the spatial limitation imposed by bi-dimensional images. Motivated by this need, efforts have been made in order to allow the tri-dimensional reconstruction of the coronal loops from tomographic images obtained through satellites.

The X-ray tomographic images below, captured by telescopes on board of the Yohkoh satellite, show the energy emission of a magnetic loop in two different depths of the solar atmosphere. The loop-bottom emits hard X-ray (HXR), that is,

* Corresponding author.
E-mail addresses: lilian@dc.ufscar.br (L.N. Faria), nelson@dc.ufscar.br (N.D.A. Mascarenhas), celio@dc.ufscar.br (C.E. Morón), saito@dc.ufscar.br (J.H. Saito), reinaldo@lac.inpe.br (R.R. Rosa), sawant@das.inpe.br (H.S. Sawant).

0262-8856/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2005.12.007

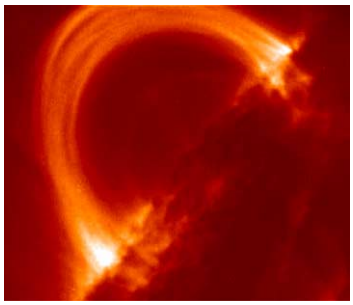

Fig. 1. Coronal loop.

X-ray of higher energy, detected by the Hard X-Ray Telescope. The loop-top emits soft X-ray (SXR) of lower energy that is detected by the Soft X-Ray Telescope. Fig. 4 shows the X-ray images with a resolution of 128×128 pixels, with information of the bottom and top of the loop, respectively.

As there is no information about the tri-dimensional structure of the loop between the top and bottom images, the reconstructed loop using traditional methods for 3D reconstructions does not present the shape of an arc. This occurs because the shape of the transversal-sections between the top and bottom images must gradually be modified to generate the arc [7].

We tried to generate the intermediate images of the loop using an image morphing method. However, as the morphing methods are controlled by linear transition functions, it was impossible to reconstruct the arc. In order to solve this problem, we developed a method for tomographic reconstruction of coronal loops, using image morphing with a transition function controlled by a Bezier curve.

Due to the high computational costs involved in the processing and visualization of solar images, in real-time, a high performance computer system becomes necessary to obtain the forecast of the solar explosions. In a joint effort between the Department of Computer Science at Federal University of São Carlos (UFSCar), the Astrophysics Division (DAS) and the Associated Laboratory for Computing and Applied Mathematics (LAC) at INPE, a parallel system (financed by FINEP-Studies and Projects Funding Agency) is being developed with capacity to support realistic applications, involving a reasonable amount of parallel processing [9]. An Atlas system based on Digital Signal Processors (DSPs), developed by Eonic Solutions GmbH, is composed by one host PC Pentium, four DSPs and the Virtuoso real-time operating system.

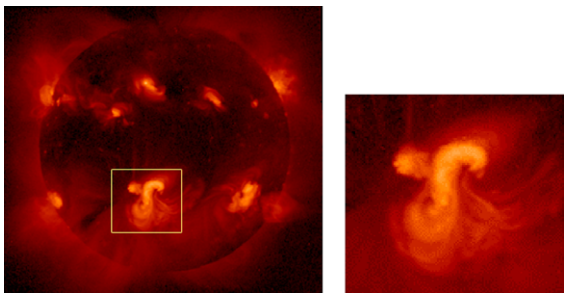

Fig. 2. Twisted coronal loop.

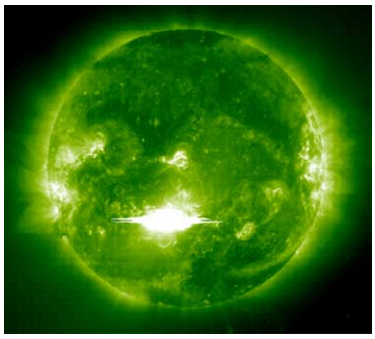

Fig. 3. Solar flare.

In this paper, we present a parallel application for 3D reconstruction of a coronal loop, in order to execute in the Atlas parallel system.

The paper is organized as follows: in Section 2, we briefly give an overview of the morphing algorithm. Section 3 describes the 3D reconstruction method using image morphing controlled by a Bezier curve. Section 4 describes the Atlas parallel system used for the development of the parallel application. Section 5 presents the parallel application. Experimental results are discussed in Section 6. Finally, Section 7 presents the conclusions about this work.

2. Image morphing

Image metamorphosis or morphing is the gradual transformation of one image into another one by the generation of a sequence of intermediate images [3,20]. Many image-morphing algorithms have been proposed [1,4,13]. In feature-based morphing algorithms [1], pairs of corresponding points in the two images are specified to compute mapping functions that define the spatial correspondence between all pixels in both images.

The morphing process consists of image warping, so that the two images have the same shape, and color interpolation with cross-dissolve between the deformed images. Cross-dissolving is the process of blending the colors of two images in order to obtain colors in an intermediate image.

Image warping is a spatial transformation technique that maps each pixel of an image into a new position of the other image [19]. In order to specify the deformation from the source image to the destination one, we need a set of pairs of corresponding points $\{(\mathbf{p}_i,\mathbf{q}_i)|\mathbf{p}_i,\mathbf{q}_i \in R^2,\ i \in 1,\ldots,n\}$, where $\mathbf{p}_i$ specifies a location in the source image that corresponds to a

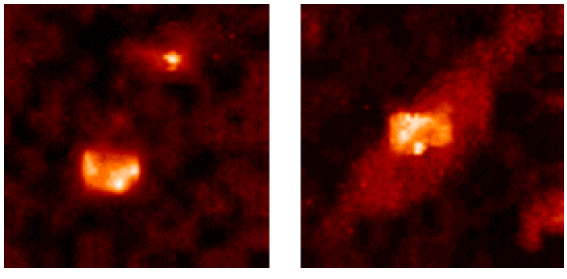

Fig. 4. Bottom (HXR) and top (SXR) images of the loop.

location $\mathbf{q}_i$ in the destination image. These pairs of control points are used to move the pixels at corresponding points from the source position to the destination one and move all other pixels along in a consistent manner [13].

There are several mapping functions for image warping [13,14]. The problem of image deformation using the inverse distance weighted interpolation method originally proposed by Shepard [15] can be formulated as follows.

Given n pairs $(\mathbf{p}_i,\mathbf{q}_i)$ of control points, we need to find at least one continuous function $f:\mathrm{R}^2\rightarrow\mathrm{R}^2$ with $f(\mathbf{p}_i)=\mathbf{q}_i$, $i=1,\ldots,n$. For each pair of corresponding points $(\mathbf{p}_i,\mathbf{q}_i)$, a local interpolator $f_i:\mathrm{R}^2\rightarrow\mathrm{R}^2$ with $f_i(\mathbf{p}_i)=\mathbf{q}_i$ is determined. Linear local interpolators are normally used to move the control points by a linear transformation [13]. The movement of neighbour pixels is controlled by a mapping function that depends on the distance between the pixel to each of the control points of the image. Therefore, pixels more distant of a control point are not affected by the displacement of this point.

The mapping function $f:\mathrm{R}^2\rightarrow\mathrm{R}^2$ is a weighted average of the local interpolator functions, with weights that depend on the distance from the pixel to its respective control points. The inverse distance weighted function is given by:

$$f(x,y)=\sum_{i=1}^{n} w_i(x,y)f_i(x,y).$$

The weight function $w_i:\mathrm{R}^2\rightarrow\mathrm{R}$ depends on the inverse distance from a pixel $\mathbf{p}(x,y)$ to the control point $\mathbf{p}_i(x_i,y_i)$, normalized by the sum of the inverse distances to all control points:

$$w_i(x,y)=\frac{d_i(x,y)}{\sum_{j=1}^{n} d_j(x,y)},$$

where $d_i(x,y)$ is the inverse euclidean distance from a pixel $\mathbf{p}$ to the control point $\mathbf{p}_i$.

The weight function w_i must satisfy the conditions:

(1) $w_i\geq 0$, and $\sum_{i=1}^{n} w_i(x,y)=1$;

(2) $w_i(x_i,y_i)=1$, with $i\neq j,i,j=1,\ldots,n$.

Fig. 5 shows the mapping process of a pixel $\mathbf{p}(x,y)$ to a new position $\mathbf{p}'(x',y')$. The position (x',y') is calculated by a mapping function defined as $f(x,y)=w_0f_0+w_1f_1+w_2f_2$, with $f_0(\mathbf{p}_0)=\mathbf{q}_0$, $f_1(\mathbf{p}_1)=\mathbf{q}_1$ and $f_2(\mathbf{p}_2)=\mathbf{q}_2$. The weights satisfy the conditions $w_1>w_0>w_2$ and $w_0+w_1+w_2=1$.

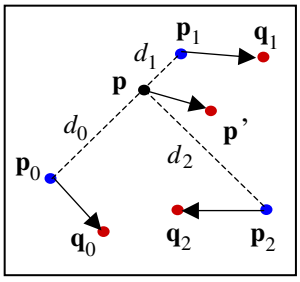

Fig. 5. Mapping of a pixel with inverse distance weighted interpolation method.

In order to obtain the metamorphosis of one image into another, we generate a progressive deformation of the source image and inverse deformation of the destination image. At the same time, a cross-dissolve of the deformed images generates the intermediate images. Therefore, the influence of the source image is attenuated while the influence of the destination image increases.

3. Reconstruction of 3D coronal loops using image morphing with Bezier curves

In this reconstruction method, the deformation of the top and bottom images of the loop is controlled by a Bezier curve to approximate the magnetic field in arc shape.

We need to specify four control points that define the 3D Bezier curve. In Fig. 6, points P_0 and P_3 correspond, approximately, to the loop footpoints observed in the bottom image. Points P_1 and P_2 are adjusted to define the shape of the magnetic loop.

The parametric polynomial function, $P(t)$, of a cubic Bezier curve is expressed by

$$P(t)=(1-t)^3P_0+3t(1-t)^2P_1+3t^2(1-t)P_2+t^3P_3,$$

where t is the normalized parameter of the curve, with values range between 0 and 1; P_0, P_1, P_2 and P_3 are control points with cartesian coordinates x, y and z.

For each point P(t) of the curve, the tangent vector is

$$\vec{P}(t)=\frac{P'(t)}{||P'(t)||}$$

where $P'(t)$ is the first derivative at the point $P(t)$.

We perform the image morphing using the inverse distance weighted interpolation method to deform the original images, and a cross-dissolve of the deformed images to generate the intermediate images [5,6]. Two points Pc_0 and Pc_1 with the same coordinate z in the Bezier curve, are used as control points in a slice, for the partial deformation of the top and bottom images.

For each control point Pc_i of a specific slice, a local interpolator $f_i:\mathrm{R}^2\rightarrow\mathrm{R}^2$ is specified. We defined the interpolator f_0 and f_1 at the control points Pc_0 and Pc_1 by

$$f_0=1-\vec{P}\mathrm{c}_0 \quad \text{and} \quad f_1=1-\vec{P}\mathrm{c}_1,$$

where $\mathbf{Pc}_0$ and $\vec{P}\mathrm{c}_1$ are the tangent vector to the Bezier curve at the points Pc_0 and Pc_1. Fig. 7 shows the warping parameters using the Bezier curve.

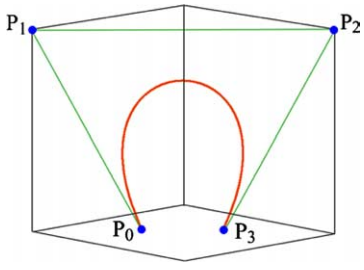

Fig. 6. Bezier curve for approximation of the tri-dimensional loop.

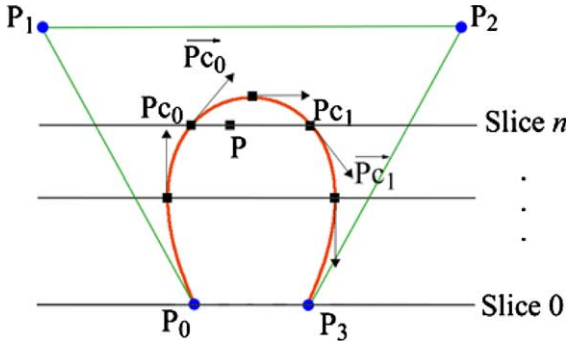

Fig. 7. Warping parameters represented by a Bezier curve.

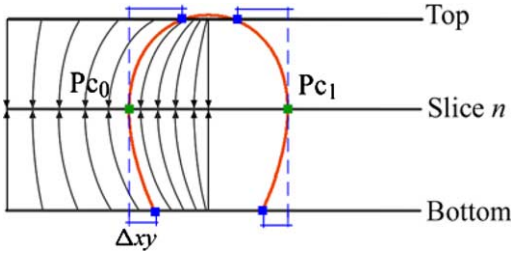

Fig. 8. Mapping of pixels in the warping process with Bezier curve.

The inverse distance weighted interpolation function $f(x,y)$ is a weighted average of the local functions f_0 and f_1, given by:

$$f(x,y) = w_0(x,y)f_0(x,y) + w_1(x,y)f_1(x,y).$$

On a slice, the weight function w_i:$R^2 \rightarrow R$ depends on the inverse distance from a pixel $P(x,y)$ to the control point $Pc_i(x_i,y_i)$, normalized by the sum of the inverse distances of all control points. So, the weight w_0 is given by:

$$w_0(x,y) = \frac{d_0(x,y)}{d_0(x,y) + d_1(x,y)},$$

where $d_0(x,y)$ and $d_1(x,y)$ are the inverse distances from the pixel $P(x,y)$ to the control points Pc_0 and Pc_1, respectively.

In order to synthesize the deformed images, each pixel at the position (x,y) of the top or bottom image is moved to a new position (x',y') at the intermediate slice n, given by:

$$(x',y') = (x,y) + \alpha(\Delta x,\Delta y),$$

being α the inverse distance weighted interpolation function, and $(\Delta x,\Delta y)$ a displacement that depends on the nearest control point.

The displacement $(\Delta x,\Delta y)$ for each point at the slice is defined as the distance xy between the nearest control point in the slice n and the corresponding control point in the bottom image, for deformation of the bottom image (or in the top image, for deformation of the top image).

As shown in Fig. 8, the control point in the bottom image is mapped to a new position in the slice n, with a displacement Δxy. The pixels that are most distant from the control point are displaced with a smaller Δxy, weighted by the interpolation function. The same procedure is used for the top image.

Simultaneously, we generate the intermediate images of the loop with a cross-dissolve of the deformed top and bottom images. Fig. 9 illustrates this process, showing some deformed slices of the top and bottom images, their respective control points and the intermediate slices generated.

Due to the need to obtain the 3D reconstruction of the coronal loops within a reasonable time, we implemented the reconstruction method to be executed in a parallel system with four processors.

4. The parallel system

The ATLAS™ system (Fig. 10), from Eonic Solutions GmbH, includes hardware and software to implement and execute applications that need high performance. This system is composed by one host PC Pentium with Windows NT and two DSP boards [2]. Each board is equipped with two processors ADSP-21160, 72 MHz (Hammerhead SHARC™) from Analog Devices. These DSPs are high performance signal processors for communications, graphics, and imaging applications, which combine floating-point operations with multiprocessing support. Other ATLAS™ boards can be added to the system.

ATLAS™ system is shipped together with the fully installed Virtuoso™ real-time operating system (RTOS) from Wind River Systems, Inc.

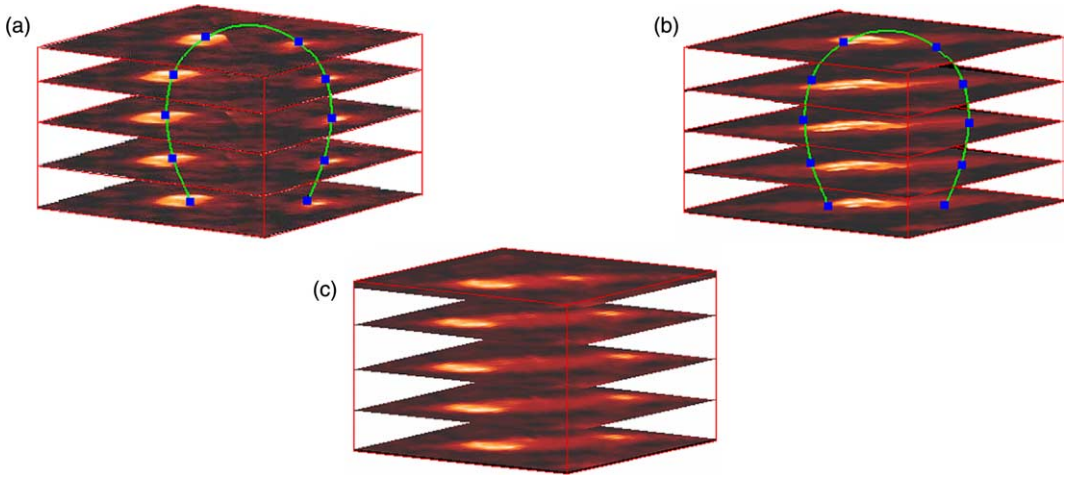

Fig. 9. Image morphing process: warping + cross-dissolve.

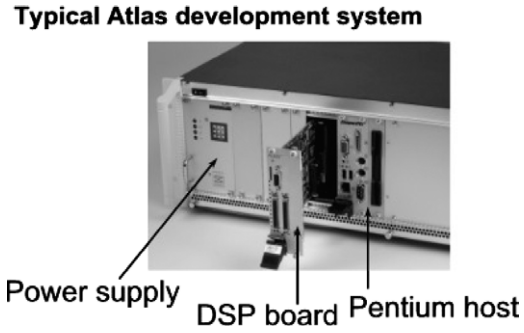

Fig. 10. Atlas parallel system.

4.1. Virtuoso real-time operating system

The Virtuoso kernel (Virtual Single Processor Programming System) is an operational system that concentrates only the objects and services necessary for the development of real-time applications in multi-processor systems [18]. Each microkernel object has specific attributes and supports a set of services.

The main objects of Virtuoso are: tasks, semaphores, mailboxes, queues, channels, memory partitions, resources and timers.

Task. A task is a program module that exists to perform a defined function or a set of functions. A task is independent of other tasks but may establish relationships with them.

Semaphore. Semaphores are used to synchronize two tasks and/or events. A signaling task will signal a semaphore while there is another task waiting on that semaphore to be signaled. One can wait on a semaphore with a time-out or return from the waiting time if no semaphore is signaled. This can be useful to make sure that the task does not get blocked.

Mailbox. Messages are used between a sender and a receiver task. This is done using a mailbox. The mailbox is used as an intermediate agent that accepts messages. Mailboxes work with arbitrary sizes and allow a selective transportation between sender and receiver.

Queue. Queues are also used to transfer data from a task to another one but here the data with fixed size is transferred in a buffered and time-ordered way. The advantage is that no further synchronization is required between the enqueuing and the dequeuing task, allowing the enqueuer to carry on.

Channel. A channel consists of queued writer(s) and reader(s) and an optional channel buffer. In the unbuffered case, data with arbitrary size will flow directly from writers to readers. When using the option of channel buffers, data will probably first be copied to the buffer before being finally transferred to the reader. Channels should be thought of as software 'pipes' that allow one task to put data in and another one to take it out. In addition, host channels allow communication between a task running in a DSP and an external program running in the host.

Memory. In any system, memory is a resource for which tasks compete.

Resource. The resource protection calls are needed to assure that the access to resources is done in an atomic way. Unless the processor can provide real physical protection, the locking and unlocking of a resource is in fact a convention that all tasks using a resource must follow.

Timer. This class of calls allows an application task to use a timer as part of its function. From there on, the timer can be started to generate a timed event at a specified point in time (one shot) or at an interval (cyclic).

The application developed using Virtuoso is divided into tasks; that is, independent program modules that can interact with other tasks through communication and synchronization services. Tasks can be associated to different priorities. After defining the objects and application code, the programmer distributes these objects among the more convenient processors until the real-time requirements are met.

4.2. Visual environment for the development of parallel real-time programs

In order to offer support for the development of parallel applications with Virtuoso, a tool Visual Environment for the Development of Real-Time Parallel Programs [8,10] was developed at the Department of Computer Science at UFSCar. The first version of the tool is available for download from http://www.dc.ufscar.br/~tev and was released as Teaching Environment for Virtuoso (TEV).

In the Visual Environment, applications are built through the construction of a graphical model. This model is represented by a graph, where nodes denote the data structures that compose a parallel program (tasks, signals, resources, mailboxes, etc.), and arrows denote the communication and synchronization operations between the structures. The information in the graphical model can be complemented with code written by the user. Based on this information, the C/C++ source code of the application is automatically generated.

5. Parallel program for 3D reconstruction of coronal loops

The application for 3D reconstruction of coronal loops is composed by a main program, written in Interactive Data Language (IDL), running in the host PC of the Atlas system, and a 3D reconstruction parallel program running in the four DSPs. The communication between these programs is obtained through a bi-directional pipe (Fig. 11).

The main program offers a user graphical interface and an interface for 3D visualization of the coronal loop.

The 3D reconstruction parallel program, implemented in the programming model based on tasks and channels is divided into five tasks (MASTER, TASK1,…,TASK4). Task MASTER distributes the original images of the loop between the tasks TASK*i*, where each task TASK*i* executes in a different processor, in order to simultaneously generate a subgroup of intermediate images.

The tasks are interconnected by channels (PIPE, CHANNEL1,…,CHANNEL4). The external channel (PIPE) allows a bi-directional communication between the main program at host PC and the tasks on the DSPs of the parallel machine. The

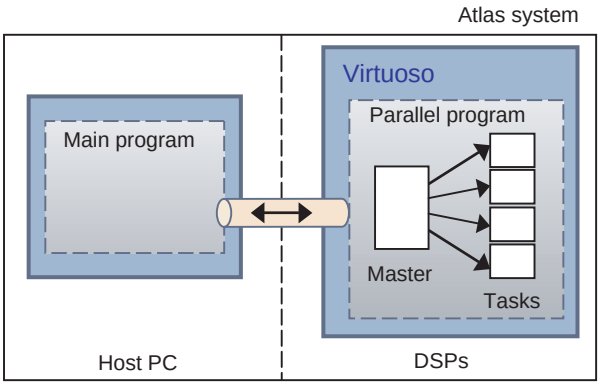

Fig. 11. Structure of the parallel application in the Atlas system.

other channels perform the inter-processor or intra-processor communication between tasks. Fig. 12 illustrates the data flow of the 3D reconstruction application.

When the main program is executed, a pipe for external communication is created, and the parallel program initiates the execution. Task MASTER opens a connection with the pipe and waits the main program to send the original images of the loop.

When the user requests a 3D reconstruction, the main program sends the original images for task MASTER, who in turn, distributes these images between the other tasks using the internal channels. Each task TASK*i* waits until receiving the original images to initiate the processing.

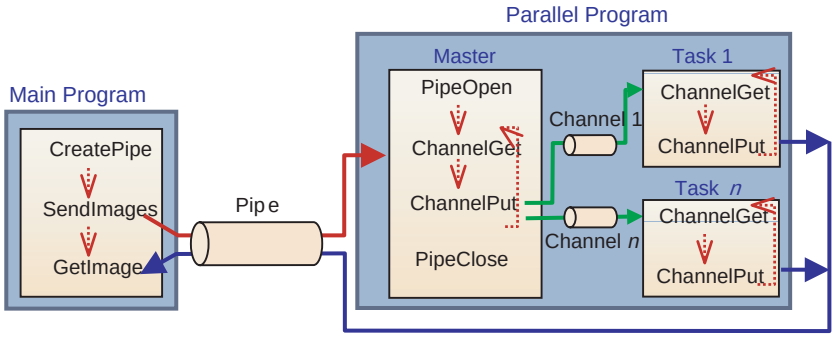

Fig. 12. Flow data of the 3D reconstruction application.

Master

```
❶ Open PIPE channel;
  Do {
❷   Get original images from the external program through PIPE;
❸   Send original images to the TASK1 through CHANNEL1;
❹   Send original images to the TASK2 through CHANNEL2;
❺   Test semaphores END1 and END2;
  } while( )
❻ Close PIPE channel;
```

Task 1

```
while (true){
①   Get original images from the MASTER through
    CHANNEL1;
    for(…){
②       Send the i-th intermediate image to the external
        program through PIPE channel;
    }
③   Signal semaphore End1;
}
```

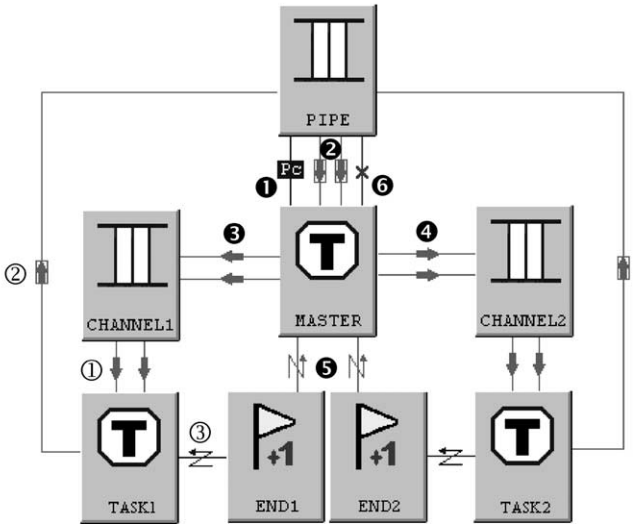

Fig. 13. Graphical representation of the parallel program in the Visual Environment.

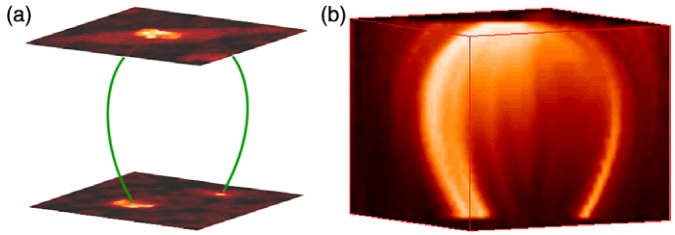

Fig. 14. 3D Bezier curve and reconstructed loop using image morphing.

During the generation of the intermediate images subgroup, tasks TASK*i* send the intermediate images to the main program through the communication pipe. The main program stores the intermediate images to obtain a volume dataset. After the 3D reconstruction, the tri-dimensional structure of the coronal loop can be visualized through an interface of volume visualization.

After that, new requests of 3D reconstruction can be executed. Finally, when the main program is finished, the parallel program closes the pipe and ends the execution.

Fig. 13 shows a simplified graphical representation of the parallel program in the Visual Environment, omitting TASK3, TASK4 and other objects such as memory pool.

After defining all the objects and writing the parallel application code, the microkernel objects can be easily distributed between the processors of the parallel machine, using the object properties editor offered by the Visual Environment. Tasks MASTER and TASK1 were mapped to DSP1, while tasks TASK2, TASK3 and TASK4 were mapped to DSP2, DSP3 and DSP4, respectively.

In the graphical description of the parallel program, the microkernel objects—tasks, semaphores, channels, and so on—and the communication and synchronization service primitives offered by the Virtuoso, are graphically represented by rectangles and connection lines, respectively. The Visual Environment automatically generates the corresponding code for the connections. In addition, specific complementary code of the tasks can be written in the C/C++ language, using the environment's text editor.

The primitive pairs *ChannelPut*() and *ChannelGet*() carry out the communication between two tasks through channels, while the primitives *SemaSignal*() and *SemaTest*() execute the synchronization between tasks using semaphores.

A synchronization mechanism between task MASTER and each task TASK*i* through the semaphores (END1…, END*n*) becomes necessary to guarantee the correct functioning of the

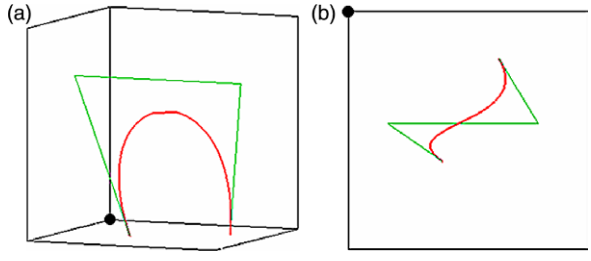

Fig. 15. Bezier curve with a sigmoid shape.

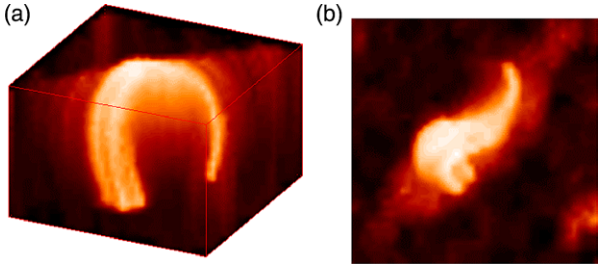

Fig. 16. Hypothetical sigmoid loop reconstructed from the X-ray tomographic images.

3D reconstruction parallel program. When task TASK*i* finishes the generation of the subgroup of intermediate images, it signals the semaphore to indicate that the end of processing has been reached. Task MASTER verifies the signal of each semaphore. When all the semaphores are signaled, task MASTER is ready to receive other original images for 3D reconstruction. This synchronization prevents that task MASTER wrongly receives an intermediate image as if it was an original image of the loop.

After defining all the objects and writing the parallel application code, the microkernel objects can be easily distributed between the processors of the parallel machine, using the object properties editor offered by the Visual Environment. Tasks MASTER and TASK1 were mapped to DSP1, while tasks TASK2, TASK3 and TASK4 were mapped to DSP2, DSP3 and DSP4, respectively.

6. Results

We tested the 3D reconstruction method with the original X-ray images of the solar atmosphere, shown in Fig. 4. A Bezier curve was defined to reconstructed the structure of the loop (Fig. 14a). The control points locations of the Bezier curve are manually chosen in order to approximate the shape and size of the loop. The 3D loop reconstructed is shown in Fig. 14b.

An advantage of the method for 3D reconstruction with Bezier curve is the capacity to obtain loops of various shapes and sizes, being enough to define the four control points for the cubic Bezier curve.

A twisted loop with a sigmoid shape can be easily reconstructed from the same top and bottom images. Fig. 15 shows the Bezier curve that defines the sigmoid shape for reconstruction of the twisted loop shown in Fig. 16.

The 3D reconstruction parallel program was executed for performance analysis in the parallel machine with four processors. Table 1 shows the execution time, the efficiency

Table 1
Execution time (in seconds), efficiency and speedup of the parallel program

Number of processors	Time (s)	Efficiency (%)	Speedup
1	75.86	100.00	1.0
2	38.36	98.88	1.98
4	19.28	98.37	3.93

and speedup of the 3D reconstruction program as the number of processors is increased.

Due to increasing overhead and the communication bottleneck, the efficiency tends to decrease and the speedup does not increase linearly as the number of processors is increased. Nevertheless, the estimated values show that the algorithm is scalable allowing an increase in the numbers of processors.

7. Conclusions

We presented a method for 3D reconstruction of coronal loops that generates intermediate slices between the top and bottom images using image morphing. This reconstruction method is controlled by a Bezier curve that defines the tridimensional structure of the loop.

Despite the insufficient number of tomographic images for 3D reconstruction of the loop, the reconstruction method presents a satisfactory visual result, allowing the reconstruction of loops of different shapes and sizes.

Motivated by the need of obtaining the 3D reconstruction of the coronal structures in real-time, a parallel program was implemented to execute in the Atlas parallel system. Preliminary results of the scalability analysis of the parallel program show that the number of processors can be increased in order to diminish the processing time.

The method for 3D reconstruction of coronal loops presented in this paper is the first step to obtain the forecast of the solar explosions from the three-dimensional dynamics of the loop. In future, the parallel program implemented for 3D reconstruction of loops will be used to reconstruct a sequence of tomographic images in constant evolution over time. In this way, a study of the spatio-temporal dynamics of the loop in three-dimensions will be possible.

Acknowledgements

We want to acknowlege to CAPES for the granted scholarship.

References

[1] T. Beier, S. Neely, Feature-based image metamorphosis, Computer Graphics, Proceedings SIGGRAPH 26 (2) (1992) 35–42.

[2] EONIC, Atlas System User Guide: Atlas2-HS V1.1, Eonic Solutions, 2001.

[3] J. Gomes, et al., Warping and Morphing of Graphical Objects, Morgan Kaufmann, São Francisco, CA, 1998.

[4] S. Lee, S. Sung, G. Wolberg, Image metamorphosis using snakes and free-form deformations, SIGGGRAPH'95, 1995, pp. 439–448.

[5] S. Lee, et al., Image metamorphosis with scattered feature constraints, IEEE Transactions on Visualization and Computer Graphics 2 (4) (1996) 337–354.

[6] S. Lee, et al., Image morphing using deformation techniques, The Journal of Visualization and Computer Animation 7 (1) (1996) 3–23.

[7] B. Luo, E.R. Hancock, Slice interpolation using the distance transform and morphing, Proceedings of the IEEE 13th International Conference on Digital Signal Processing, 1997, pp. 1145–1149.

[8] C.E. Morón, et al., A visual environment integrating design, implementation and debugging in parallel real-time systems, 12th Brazilian Symposium on Computer Architecture and High Performance Computing, SBAC-PAD, Brazil, October 2002, pp. 313–319.

[9] C.E. Morón, et al., Parallel architecture using DSPs, Proceedings of the Ninth Brazilian Symposium on Computer Architecture and High Performance Computing, SBAC-PAD'97, Brazil, 1997, pp. 605–608.

[10] J.R.P. Ribeiro, N.C. Silva, C.E. Morón, A visual environment for the development of parallel real-time programs, Lecture Notes in Computer Science 1388 (1998) 994–1014.

[11] R.R. Rosa, et al., Phenomenological dynamics of coronal loops using a neural network approach, Advances in Space Research 25 (9) (2000) 1917–1921.

[12] R.R. Rosa, A.S. Sharma, J.A. Valdivia, Characterization of asymmetric fragmentation patterns in spatially extended systems, International Journal of Modern Physics C 10 (1) (1999) 147–163.

[13] D. Ruprecht, H. Müller, Deformed cross-dissolves for image interpolation in scientific visualization, The Journal of Visualization and Computer Animation 5 (3) (1994) 167–181.

[14] D. Ruprecht, H. Muller, Image warping with scattered data interpolation, IEEE Computer Graphics and Applications 15 (2) (1995) 37–43.

[15] D. Shepard, A two dimensional interpolation function for irregularly spaced data, in: Proceedings of the 23rd National Conference on ACM, 1968, pp. 517–524.

[16] S. Tsuneta, J.R. Lemen, Dynamics of the solar coronal observed with the Yohkoh soft X-ray telescope, Physics of Solar and Stellar Coronae (1993) 113–130.

[17] Y. Uchida, New aspects of solar coronal physics revealed by Yohkoh, Physics of Solar and Stellar Coronae (1993) 97–112.

[18] VIRTUOSO™, User Guide for Version 4.2, Wind River Systems, 2001.

[19] G. Wolberg, Digital Image Warping, IEEE CS Press, Los Alamitos, CA, 1990.

[20] G. Wolberg, Image morphing: a survey, The Visual Computer Journal 14 (1998) 360–372.