Parallel Interactive Virtual Machining on Shared Memory Multiprocessors

N. Mahesh and S. Manohar

Department of Computer Science and Automation
Indian Institute of Science
Bangalore, INDIA

Abstract

Interactive sculpting is the process by which a designer can impose free-form shape changes on the object being designed. It has potential applications in the fields like computer aided geometric design and rapid prototyping. Our approach to interactive sculpting is through the intermediate step of interactive virtual machining (IVM). IVM is a subset of interactive sculpting in that the degrees of freedom available to the sculptor are reduced. The goal of our work is to demonstrate the feasibility of interactive virtual machining on a shared memory multiprocessor workstation. Our prototype IVM system uses a voxel based approach. It provides common machining tools to the user and uses Minkowski operations to implement those tools.

Parallel implementation of the virtual machining tools has shown that sculpting with 256^3 voxel array is possible with frame rate of around 20 frames/sec. We present the parallel algorithms for virtual machining tools and the results of implementation on a shared memory multiprocessor environment.

1 Introduction

Interactive sculpting is the process by which a designer can impose free-form shape changes on the object being designed. It has potential applications in the fields like computer aided geometric design and rapid prototyping. Critical problems associated with interactive sculpting are:

- The design of powerful user interfaces necessary to enable a designer to modify a 3-D object using a 2-D display.
- The design of efficient algorithms for performing the sculpting operations at interactive rates. and
- Displaying the results of sculpting at interactive rates.

We attempt to overcome these problems by approaching interactive sculpting through the intermediate step of interactive virtual machining (IVM). IVM is a subset of interactive sculpting in that the degrees of freedom available to a designer are reduced. In other words, in IVM the designer is provided with simple sculpting tools, resembling real life machining tools, like milling, turning and thread cutting. This simplifies the user interface requirements while reducing the range of shapes that can be created. However, unlike in real machining, the designer need not have to worry about the real life constraints like tool colliding with the block or require that the object should be cylindrical for turning etc. Hence a much richer range of shapes can be created by IVM.

So far, voxel based modeling and sculpting have not received sufficient attention due to the prohibitive memory and computational costs associated with it. But with the decreasing hardware costs and progressing research in high performance computing, these problems are fading away. Hence, we have chosen a voxel based approach which simplifies the computational difficulty of virtual machining operations. Minkowski operations [1] are used in implementing the virtual machining tools. To achieve interactive rates however, we need to resort to parallel implementation which is the focus of this paper.

The last problem of displaying the results of sculpting at interactive rates can be tackled with a dedicated rendering pipeline for the computed volume buffer and/or resorting to parallel volume rendering [2] and we assume these in our results. Research in hardware support for volume rendering is progressing rapidly [3] and we expect real time volume rendering to become available as a standard part of commercial workstations in the near future.

We explain Minkowski operations in section 2. Sequential algorithms for virtual machining tools are dealt with in section 4 and their parallel versions are dealt with in section 5. Section 6 gives the implementation details and results. The final section outlines the areas of ongoing research.

2 Minkowski operations

Minkowski operations [1] is the basic algorithmic tool used in implementing the virtual machining tools . Brief description of them is as follows.

Minkowski addition of two sets A and B in R^d is defined as

the union of sets obtained by positioning one of them, say B, at every point of the other, say A. i.e. the set of points obtained by vectorially adding each point in A with each point in B.(Fig 1)
Mathematically, if A_p denotes the translate of a set A by the vector p, i.e., $A_p = A \oplus \{p\}$, then,

$$A \oplus B = B \oplus A = \bigcup_{a \in A} B_a = \bigcup_{b \in B} A_b$$

which is same as,

$$A \oplus B = \{a + b : a \in A, b \in B\}$$

where $\oplus$ stands for Minkowski addition.
Minkowski decomposition of two sets A and B in R^d is defined as

$$A \ominus B = \bigcap_{b \in B} A_{-b} = \bigcap_{b \in B'} A_b$$

where $\ominus$ stands for Minkowski decomposition operation. The set B' = { -b : b $\in$ B } is generally known as the symmetrical set of B with respect to the origin.

3 Virtual machining tools

Our prototype IVM system provides the designer with the following virtual machining tools.

Simple Milling : This operation refers to the cutting/pasting with the tool along a straight line segment from/to clay.

If C is the clay and S is the swept volume of the tool along the line segment, then

$$Result = \begin{cases} C - S & \text{for cut operation} \\ C \cup S & \text{for paste operation} \end{cases} \quad (1)$$

Contour Milling : This tool refers to, cutting/pasting with the tool along the contours- curves, from/to clay. Contours are represented using B-Spline curves.

Turning : This tool refers to, cutting/pasting with the tool along circular paths, from/to clay.

Thread Cutting : This tool refers to, cutting/pasting with the tool along helical paths, from/to clay. It will useful in creating screws/nuts like shapes.

Sequential version of our prototype IVM system has been implemented. Fig 2 shows some of the sculpted objects. Both octree based [4] and voxel array based IVM systems have been implemented. We limit our discussion to the voxel array based implementation of IVM.

4 Sequential algorithms for virtual machining tools

For the following discussion assume that

Clay - 3-D array of booleans of size C_x x C_y x C_z
Tool - 3-D array of booleans of size T_x x T_y x T_z

'Clay' is the object voxel array being sculpted and 'Tool' is the tool voxel array used for sculpting.

4.1 Simple Milling

From equation (1) it is clear that the problem is to find the swept volume 'S' of the tool along a straight line segment. Computing this volume boils down to the computation of the Minkowski sum of the tool with the line segment. Because of our choice of voxel based modeling this computation is greatly simplified. Summary of the algorithm is given below.

```
do_simple_milling( Clay, Tool, line-segment, operation )
{
    S = compute_sweep( Tool, line-segment )
    if( operation is cut )
        Result = Clay - S = Clay - (Clay ∩ S)
    else    //operation is paste
        Result = Clay ∪ S
}
```

Note: Boolean operations like union and intersection are trivial on voxel arrays.

```
compute_sweep( Tool, line-segment )
{
    Result = φ      // Empty voxel array
    for( each boundary voxel V=(Vx,Vy,Vz) in Tool )
    {
        S = compute_sweep_for_one_voxel(V, line-segment)
        Result = Result ∪ S
    }
}

compute_sweep_for_one_voxel( V, line-segment )
{
    Result = φ      // Empty voxel array
    for( each point (lx,ly,lz) on line-segment )
    {
        V_shifted = (Vx,Vy,Vz) + (lx,ly,lz)
        Result = Result ∪ V_shifted
    }
}
```

4.2 Contour Milling, Turning, and Thread Cutting

For all these tools, the curves involved are divided into smaller line segments and simple milling is applied to each segment to get the result.

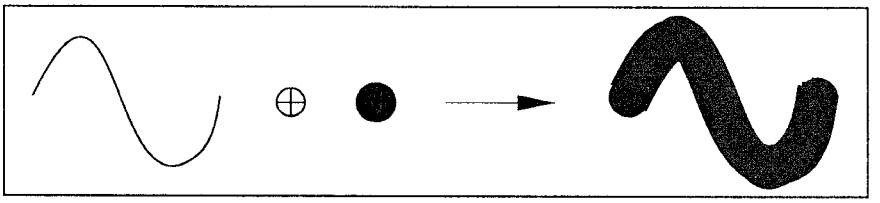

Figure 1: **An example of Minkowski Addition**

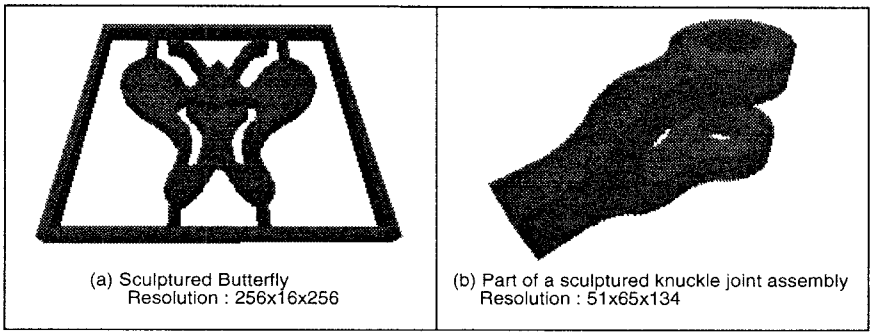

Figure 2: **Examples of objects sculpted using our prototype IVM system**

5 Parallel version of algorithms for virtual machining tools

Parallelism exists at various levels in the algorithms explained in the previous section. These parallelisms are listed below in the decreasing order of grain size.

1. In *contour milling*: Each line segment can be cut/paste from/to the clay independent of other line segments.
2. In *compute_sweep()*: Computing and inserting the sweep of a boundary voxel of the tool is independent of other boundary voxels of the tool.
3. In *compute_sweep_for_one_voxel()*: Processing for each point on the line segment is independent of the other points on the line.

We shall limit our discussion to the first two cases as we believe that the third case is too fine grained and the overheads involved may outweigh the benefits of parallelization. Summary of the parallel algorithms for the first two cases is given below.

do_parallel_contour_milling(Clay, Tool, Contour)
{
* *Divide the contour in to a set of line segments L such that the lengths of the line segments are nearly equal*
* *Partition L as equally as possible among the processors available*
* *Each processor has to perform* do_simple_milling() *for each of the line segments assigned to it*
}

parallel_compute_sweep(Tool, line-segment)
{
* *Partition the boundary voxels of 'Tool' as equally as possible among the processors available*
* *Each processor has to perform* compute_sweep_for_one_voxel() *for each of the boundary voxels assigned to it*
}

Note that, because of the shared memory implementation, all the processors operate directly on the clay and hence no phase is required for combining the results from all the processors.

6 Implementation details and results

Virtual machining tools library has been implemented in C on a SGI PowerChallenge shared memory multiprocessor machine with 16 R10000 CPUs (194/196 MHz), and 3 GB main memory.

Table 1 shows the frame rates possible for two cases: one with the clay size of 256^3 and the other with 512^3. We assume that a dedicated rendering pipeline is available for

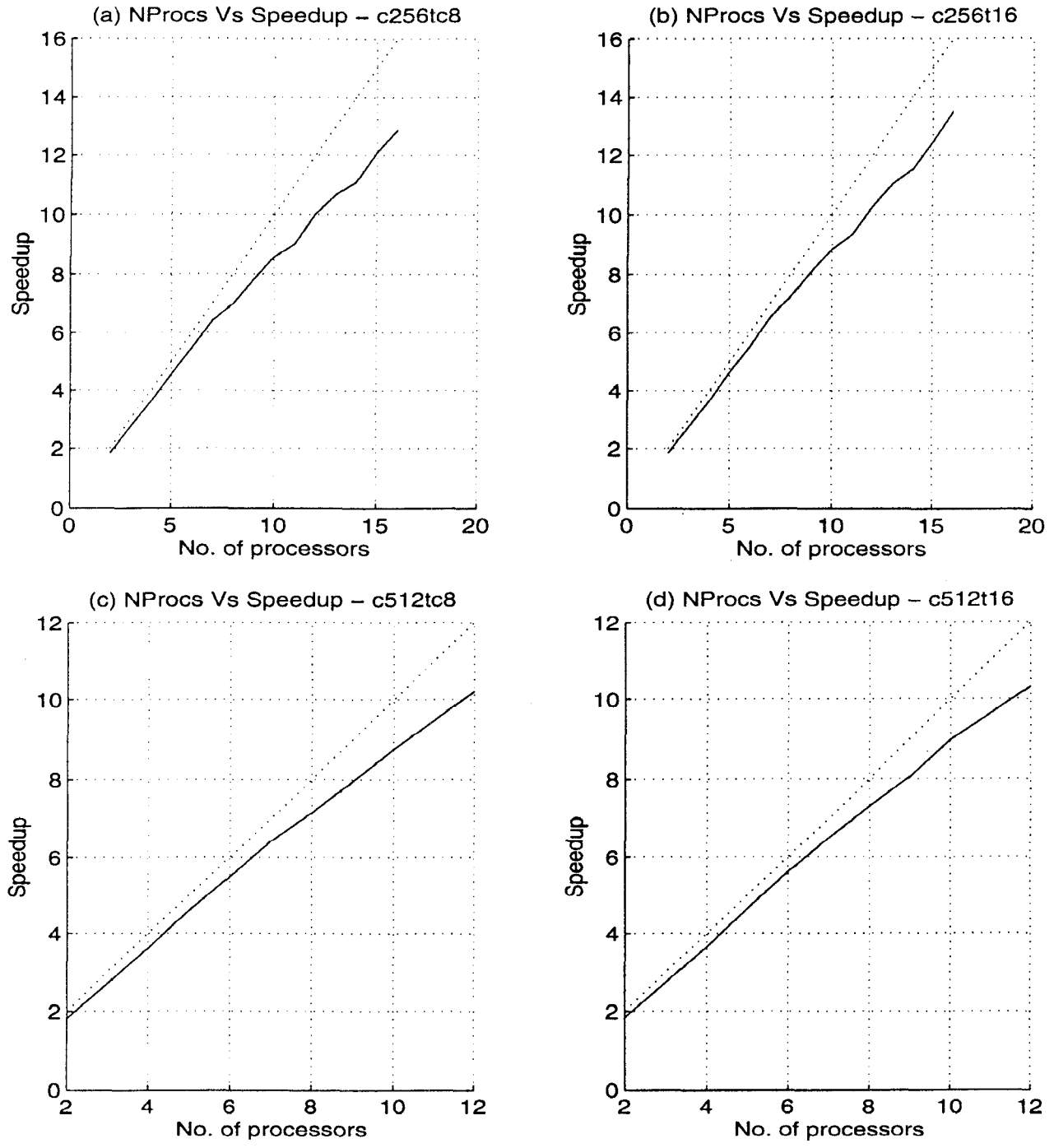

Figure 3: **Speedups achieved for turning operation on a SGI PowerChallenge - shared memory multiprocessor machine.**

dotted-lines :	ideal speedup	
solid-lines :	actual speedup	
(a) clay : 256^3	tool: cylinder r=8, h=8	Turning Radius=120
(b) clay : 256^3	tool: 16^3	Turning Radius=120
(c) clay : 512^3	tool: cylinder r=8, h=8	Turning Radius=248
(d) clay : 512^3	tool: 16^3	Turning Radius=248

No. of procrs.	Sculpting Time(sec)		Frame Rate(frames/sec)	
	clay size:256^3	clay size:512^3	clay size:256^3	clay size:512^3
1	0.668630	1.396508	1.50	0.72
2	0.362576	0.757456	2.76	1.32
3	0.242865	0.510739	4.12	1.96
4	0.183931	0.382726	5.44	2.61
5	0.143941	0.300281	6.95	3.33
6	0.121068	0.248662	8.26	4.02
7	0.102442	0.214898	9.76	4.65
8	0.092376	0.191070	10.83	5.23
9	0.082655	0.173412	12.10	5.77
10	0.075570	0.155718	13.23	6.42
11	0.071597	0.145033	13.97	6.89
12	0.065150	0.135381	15.35	7.39
13	0.060685	—	16.48	—
14	0.058045	—	17.23	—
15	0.053664	—	18.63	—
16	0.049518	—	20.19	—

Table 1: **Frame rates possible for the compute intensive cases in the corresponding clay sizes.**

Computation Involved with clay size of 256^3: Turning with radius = 120 units.
Computation Involved with clay size of 512^3: Turning with radius = 248 units
and the tool size is 16^3 in both the cases

the computed volume buffer. The results are for fairly compute intensive turning operations with 16^3 tool size which has 1352 boundary voxels. We can observe that frame rate of 20 frames/sec is possible for 256^3 clay size, with 16 processors, which is close to interactive rates. We are working on higher resolution voxel arrays. With the results available so far, frame rate of around 7 frames/sec is possible for 512^3 clay size, with 12 processors. Extrapolating the results, we expect frame rate around 10-12 frames/sec, with 16 processors. The graphs in Fig 3 show the speedups for various turning operations explained therein. From the graphs it is clear that the speedup curves have not yet saturated. So, we can expect higher frame rate by increasing the number of processors.

Our on going experiments on parallel implementation of other virtual machining tools on a SGI PowerChallenge shared memory multiprocessor machine with 6 R8000 CPUs (90/75 MHz), and 1 GB main memory, show promising results. For simple milling speedup is in the range of 4.9 to 5.3.

7 Future Work

Our ongoing research focuses on

- Integrating the virtual machining tools with a real time volume rendering system.
- Improving upon the present implementation of virtual machining tools to achieve better load balance and hence better speedups.
- Parallelizing general linear transformations (like scaling, rotation and shearing.) on voxel arrays.

8 Conclusion

We have approached interactive sculpting through the intermediate step of interactive virtual machining (IVM), in that the degrees of freedom available to the designer are reduced. We have found that the sculpting operations are the bottleneck for interactiveness. So, we have developed parallel algorithms for virtual machining tools like milling, turning and thread cutting. Results of the parallel implementation have shown that frame rate of around 20 frames/sec is possible with a clay size of 256^3. Our future work focuses on higher resolution voxel arrays.

Acknowledgments

We thank Dr. C.E. Prakash for running our code on a SGI PowerChallenge shared memory multiprocessor ma-

chine with 16 R10000 CPUs and thank Prof. A. Kaufmann, SUNY, Stony Brook, for permitting the use of the machine.

References

[1] P.K. Ghosh, *"A Unified Computational Framework for Minkowski Operations"*, Computer & Graphics, Vol.17, No.4, pp.357-378, 1993.

[2] Philippe Lacroute, *"Real-Time Volume Rendering on Shared Memory Multiprocessors Using the Shear-Warp Factorization"*, Parallel Rendering Symposium '95, pp.15-22, October 1995.

[3] S. Manohar and C.E. Prakash, *"Hardware architecture for voxelization based volume rendering of unstructured grids"*, Tenth Eurographics Workshop on graphics hardware, editor: W. Straber, pp.103-115, Aug 1995.

[4] S.U. Sethia, *"Interactive volume sculpting"*, Masters project report, Department of Computer Science and Automation, Indian Institute of Science, India, Jan 1996.