T-Buffer: Fast Visualization of Relativistic Effects in Spacetime

Ping-Kang Hsiung*
Robert H. Thibadeau†
Michael Wu‡
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

We have developed an innovative ray-tracing simulation algorithm to describe *R*elativistic *E*ffects in *S*pace*T*ime ("REST"). Our algorithm, called *REST-frame*, models light rays that have assumed infinite speed in conventional ray-tracing to have a *finite* speed in spacetime, and uses the non-Newtonian Lorentz Transformation to relate measurements of a single event in different inertial coordinate systems (*inertial frames*). Our earlier work [5][6][7] explored the power of *REST-frame* as an *experimentation tool* to study the rich visual properties in natural world modeled by Special Relativity. Non-intuitive images of the anisotropic deformation ("warping") of space, the intensity concentration/spreading of light sources in spacetime, and the relativistic Doppler shift were visualized from our simulations.

REST-frame simulations are computationally expensive. Several hours of CPU time may be needed to generate one intricate image on a relatively powerful DECStation 3100. This high simulation cost of *REST-frame* precludes its application in interactive, real-time graphics environments.

In this paper, we report a scanline based *REST-frame* rendering method that provides a faster alternative to the original ray-tracing based *REST-frame* implementation. This new method operates in the spirit of the classical Z-buffer in computer graphics[2] and the inter-inertial frames point-mapping method investigated in physics in the early 1960's[14][12], and determines the visibility of points in spacetime by their spatial *and* temporal visibility. Specifically, all spacetime event points that are *potentially* visible from the viewpoint at the imaging time are geometrically projected in three dimensional (3D) space to the image plane pixel buffer. Multiple points with a same pixel affiliation are sorted by their *time distance* from the imaging time, and the most recent spacetime point is displayed.

This method, which we call *"Time-Buffer"* or *"T-Buffer"*, offers a significant speed improvement over the original *REST-frame* in software, and permits a dedicated Z-buffer-type hardware implementation that promises interactive, real-time relativistic effects in simulations on a contemporary graphic workstation.

Motion blur in real world images caused by the non-infinitesimal exposure time of image-taking can be simulated by *"Stochastic T-Buffer"*, which perturbs the *time component* of the scan-converted spacetime events that are potentially visible. The classical A-Buffer technique[1] that models translucency also can be adapted easily in T-Buffer. The limitation of T-Buffer is its inability to model specular reflection and refraction in optics,[1] which our original *REST-frame* implementation simulates completely.

*Department of Electrical and Computer Engineering, Carnegie Mellon University. (412) 268-2524. pkh@vap.vi.ri.cmu.edu

†Imaging Systems Laboratory, The Robotics Institute, Carnegie Mellon University. rht@vi.ri.cmu.edu

‡Department of Electrical and Computer Engineering, Carnegie Mellon University. mw2u@andrew.cmu.edu

This research was partially supported by the Inland/Fisher-Guide Division of General Motors Corporation and the Imaging Systems Laboratory, The Robotics Institute, Carnegie Mellon University.

1 Introduction

In the conventional rendering algorithms, light had always been regarded as if it traveled with infinite speed, and Galilean-Newtonian transformation was used to model relative motion between dynamic systems and the observer. When the scene objects and the observer (or the camera plate) are in relative motion at speeds comparable to light speed, Special Relativity requires the time information to be interwoven with the spatial coordinates in defining the vision formation process. Light speed must be treated as *finite*, and inertial reference frames[2] ("frames" in short) are to be connected by the Lorentz Transformation.

In [5], we first treated the subject of visualizing the spacetime world of Special Relativity with the application of an innovative ray-tracing technique *REST-frame*. Objects were assumed to make one dimensional (1D) motion relative to the observer. The simulation of 3D relativistic motion was later completed[6]. Extension to simulate kinematic systems containing objects of different relativistic velocities was reported in [7], which also included our initial investigation of the relativistic Doppler shift

[1] although it does render diffusive reflection and shadow casting correctly.

[2] A reference system, or *reference frame*, is *inertial* if it is nonaccelerating.

effects.

Our original *REST-frame* implementation took a ray-tracing based simulation approach.[3] It gave good turn-around times for simple scenes, but became expensive in computation for more complex scenes. In a 3D lattice scene that contained 432 diffusive cylinders, 1731 reflective spheres and 12 light sources, some simulations took over two hours on a DECStation 3100 workstation to generate non-antialiased images of 512 by 512 in resolution, or a rendering rate of roughly 450 rays per second. Whereas *REST-frame* simulations reveal intricate images of complete and accurate optical phenomena of reflection, refraction and shadow casting under relativistic condition, their cost in time precludes most real-time, interactive graphics applications to benefit from it; a faster, perhaps less complete, visualization method may be more desirable in the latter environments. In this paper, we report one such method — T-Buffer, and discuss its speed advantage and some other features.

2 Approach

In order to model the physics of high speed motion, the *REST-frame* technique synthesizes the visual effects in spacetime by incorporating the true physics of Special Relativity and finite light-speed in its simulations. Specificially, it includes the two postulates of Special Relativity[11][13][9]:

1. Non-existence of preferred reference system (*"The Principle of Relativity"*): the laws of physics must be the same for observers in all inertial reference systems.
2. Constancy of speed of light: c is constant in a vacuum in all inertial frames and is independent of the motion of a light source relative to the observer.

According to these postulates, the *measured* space and time coordinates are dependent upon the reference frame from which the measurement is conducted; and the Lorentz Transformation equations relate measured spacetime coordinates between inertial reference frames.

2.1 Principle

A block diagram of our T-Buffer implementation of *REST-frame* is presented in figure (1). We assume the image plane to be stationary in a frame S, and the objects to move *in unison* with respect to S at a velocity $\vec{V} = (u, v, w)$. Of the infinite frames in which the objects are stationary, we can find one frame S' that has its axes X', Y' and Z' coincide with the S frame axes X, Y and Z, respectively, at time $t = t' = 0$. We call S the imaging frame or the camera frame, and S' the object frame. The measurements of every spacetime event[4] e in S and S' can be connected through the Lorentz Transformation[9]:

$$\vec{X'} = \vec{X} + [\frac{(\gamma - 1)}{\| \vec{V} \|^2}(\vec{X} \cdot \vec{V}) - \gamma t]\vec{V}$$

$$t' = \gamma(t - \frac{\vec{X} \cdot \vec{V}}{c^2}) \quad (1)$$

in which $\| \vec{V} \|^2 = u^2 + v^2 + w^2$, and $\gamma = 1/\sqrt{1 - \frac{\|\vec{V}\|^2}{c^2}}$. Vector $\vec{V}$ is sometimes replaced by vector $\vec{\beta} = (\beta_x, \beta_y, \beta_z) = (u/c, v/c, w/c)$. The Inverse Lorentz Transformation is

$$\vec{X} = \vec{X'} + [\frac{(\gamma - 1)}{\| \vec{V'} \|^2}(\vec{X'} \cdot \vec{V'}) - \gamma t']\vec{V'}$$

$$t = \gamma(t' - \frac{\vec{X'} \cdot \vec{V'}}{c^2}) \quad (2)$$

In which vector $\vec{V'} = -\vec{V}$. As a shorthand, we will write eq. (1) as $e'_{S'} = Le_S$ and eq. (2) as $e_S = L^{-1}e'_{S'}$. L and L^{-1} are the Lorentz Transformation operator and its inverse, respectively, and e_S and $e'_{S'}$ stand for the spacetime descriptions of an event in S and S', respectively.

Let us denote the viewpoint (the camera position) as $(x_{from}, y_{from}, z_{from})$ and the imaging time as t_{from} in S. Together, they form the *imaging event* e_{from_S}. If we perform the Lorentz Transformation to the imaging event, we get its S' description $e'_{from_{S'}}$:

$$e'_{from_{S'}} = [x'_{from}, y'_{from}, z'_{from}, t'_{from}]_{S'} = L\ e_{from_S} \quad (3)$$

According to the second postulate of Special Relativity ("constancy of light-speed"), all events $e'_{v_{S'}} = [x', y', z', t']_{S'}$ that are potentially visible from the imaging event $e'_{from_{S'}}$ in *temporal* sense must satisfy

$$\| e'_{v_{S'}} e'_{from_{S'}} \|_{3D} = c|t' - t'_{from_{S'}}| \quad (4)$$

That is, all potentially visible events $e'_{v_{S'}}$ satisfy the time constraint

$$t' = t_{from_{S'}} - \frac{1}{c}\sqrt{(x' - x'_{from})^2 + (y' - y'_{from})^2 + (z' - z'_{from})^2} \quad (5)$$

For each spatial point (x', y', z') in S', equation (5) dictates the time at which event $e_{v_{S'}} = [x', y', z', t']$ must have occurred in S' in order for it to reach the camera plate (at light speed c) at time t_{from_S}. For every point on the object surfaces,[5] this equation gives the specific emission time in S' of the photons from the (steadily illuminated or illuminating) surface point that precisely make it to the viewpoint[6] at the imaging time.

For the final geometric visibility test, every such potentially visible event $e'_{v_{S'}}$ is transformed into the camera frame S by applying eq. (2) to it:

$$e_{v_S} = [x, y, z, t]_S = L^{-1}\ e'_{v_{S'}} = L^{-1}\ [x', y', z', t']_{S'} \quad (6)$$

In S, the spatial coordinates of e_{v_S} is perspectively projected to the image plane and its associated "Time-Buffer" that registers a time for each pixel. The value t of e_{v_S} is then compared against the Time-Buffer time of the pixel onto which e_{v_S} is spatially projected, and *a larger t replaces the smaller t* in the Time-Buffer. The replacement rule is based on the observation that t represents the time in the past that a potentially visible event occurred, and a 3D point in a more recent past (a larger t) obscures all points in the more remote past (the smaller t's), if the point is opaque.[7]

[3]Ours is a ray-tracer that uses the hierarchical bounding box[8][4] method to accelerate ray-object intersection test.

[4]We use the symbol (x, y, z) for 3D positional coordinates and $[x, y, z, t]$ for a spacetime event point. When we designate a specific reference frame S, we use $(x, y, z)_S$ and $[x, y, z, t]_S$. Individually, each component is written with a subscript S (e.g. t_S). We also use $\vec{X'} = (x', y', z')_{S'}$ and $\vec{X} = (x, y, z)_S$.

[5]Note that object points are stationary in S'.

[6]which is measured as in motion in S'.

[7]This replacement rule is the opposite of that of the conventional Z-buffer, but is consistent with the geometric interpretation of the latter.

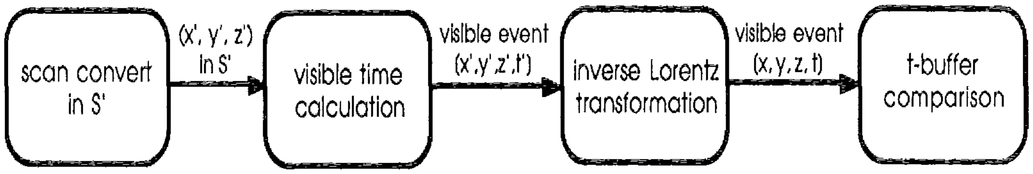

Figure 1: T-Buffer algorithm block diagram

2.2 Algorithm

In summary, the T-Buffer algorithm works as follows (refer to figure (1)):

1. Initialization:
   (a) Initialize the Time-Buffer array to -HUGE_VAL.
   (b) Use eq. (3) to determine the imaging event in S', $[x'_{from}, y'_{from}, z'_{from}, t'_{from}]_{S'}$.
2. T-Buffer visibility test:
   (a) For each object in S', scan-convert its surface to obtain spatial points $(x', y', z')_{S'}$.
   (b) From eq. (5), calculate the visible time t' associated with each point $(x', y', z')_{S'}$. This gives the *potentially visible* event $[x', y', z', t']_{S'}$ in spacetime corresponding to $(x', y', z')_{S'}$.
   (c) Transform each event $[x', y', z', t']_{S'}$ to its S coordinates $[x, y, z, t]$ using eq. (6).
   (d) Use the time t in $[x, y, z, t]$ to do Time-Buffer comparison/replacement. When a replacement occurs, the ID of the scanned object is also stored in the corresponding image pixel memory.
3. Image rendering: For each image pixel, retrieve the object ID and render the pixel according to the cosine law of diffusive reflection.

2.3 Extensions

2.3.1 Shadow

Shadow casting can be easily added to T-Buffer by applying shadow-buffering technique[10]. The light sources in each *REST-frame* simulation can be stationary in either the camera frame S or the object frame S'. These two possible configurations result in different relative speeds for the "mirror light sources" — the imaginary light sources reflected off object surfaces — to the imaging event. Note that in either case, the speed of light is non-additive, and the correct treatment is naturally accounted for in our algorithm.

2.3.2 Motion Blur

Motion blur can be simulated by Stochastic T-Buffer that adds perturbation to the time component of every scan converted event $[x', y', z', t']_{S'}$ (figure (2)). The foundation for time perturbation is the inclusion in the Lorentz Transformation (and its inverse) of the relative motion between objects and image plane. Consequently, a time perturbation in S' is correctly transformed into a spatial displacement in the imaging frame S.

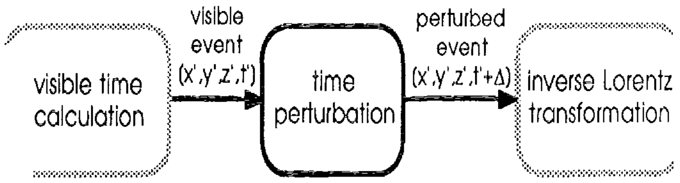
visible time calculation
visible event (x',y',z',t')
time perturbation
perturbed event (x',y',z',t'+Δ)
inverse Lorentz transformation

Figure 2: Motion blur modification of T-Buffer

3 Experiments

In this section, we show images generated by our software T-Buffer implementation, and evaluate the T-Buffer performance.

3.1 T-Buffer images

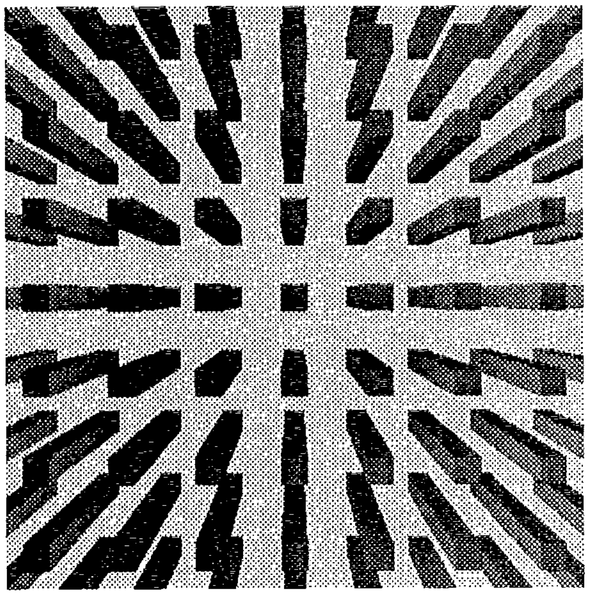

Figure 3: Array of bars at stationary ($\vec{\beta} = (0, 0, 0)$)

Figures (3), (4), (5), (6), and (7) are images generated by T-Buffer. For comparison of image quality, we show in Figure (8) an image produced by our previous ray-tracing based implementation under the same viewing condition as in Figure (7). The reflective highlight apparent in Figure (8) is not reproduced by T-Buffer in (7), although this difference is not essential.

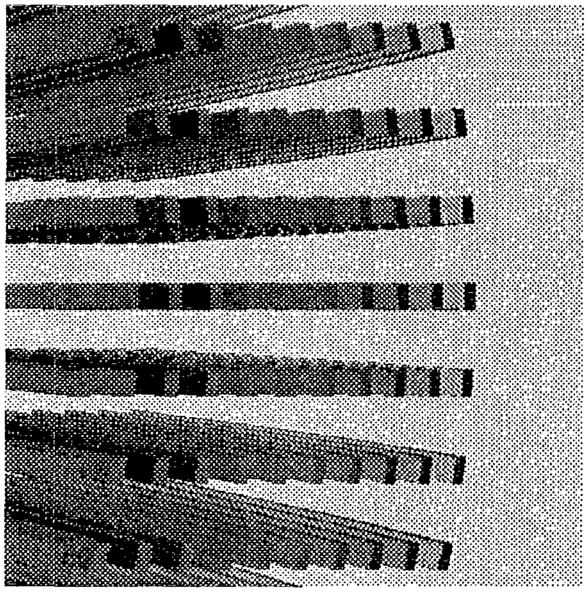

Figure 4: Array of bars at $\vec{\beta} = (0.9, 0, 0)$

	teapot (0.0c)	teapot (0.9c)	bars (0.9c)
Polygons	9121	9121	678
Scan time	1.366 S	1.950 S	10.883 S
T-Buffer time	7.184 S	12.533 S	146.05 S
Shading time	3.883 S	4.450 S	6.433 S
Polygons/sec	733.6	481.8	4.15
Memory usage	17M Byte	17M Byte	16M Byte

Table 1: Basic characterization (with shadow)

3.2 Performance evaluation

Table 1 shows the basic performance characteristics of T-Buffer on an Apollo DN-10000 system with 32M Bytes of memory. Both "teapot" and "bars" in our simulations had 4 light sources. Shadow casting was not included in the simulations. The T-Buffer time in the table refers to the T-Buffer comparison/replacement time. Note the extremely low efficiency in the "bars" column. The bars scene refers to the bar array shown in Figure (3), (4) and (5), which has a high percentage of polygons that are invisible or close to orthogonal to the final image plane, and thus has many wasted scan-conversion and T-Buffer operations.

To compare T-Buffer with our ray-tracing based *REST-frame* implementation, we ran both programs on a same set of simulation tasks. Some representative timing data are summarized in Table 2 and 3 (We rate the ray-tracing version based on polygon/second to compare with T-Buffer). Each timing datum in Table 2 is the sum of its corresponding scan time, T-Buffer time and shading time. In favorable cases, the speed improvement of the software T-Buffer over our ray-tracing based *REST-frame* is 6-7 times.

	teapot (0.0c)	teapot (0.9c)	bars (0.9c)
No shadow	12.43 S	18.93 S	163.37 S
Poly/sec	733.6	481.8	4.15
Shadowed	35.72 S	56.17 S	334.8 S
Poly/sec	255.4	162.4	2.03

Table 2: Performance comparison (T-Buffer)

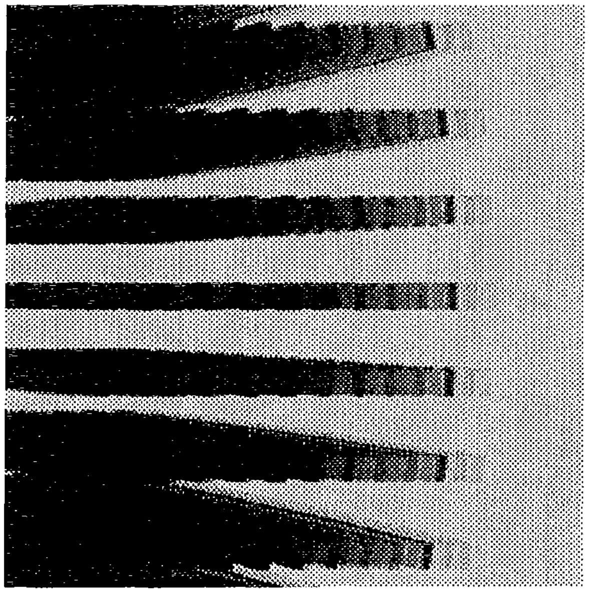

Figure 5: Bar array at $\vec{\beta} = (0.9, 0, 0)$, with 0.05 second shutter time

	teapot (0.0c)	teapot (0.9c)	bars (0.9c)
No shadow	78.00 S	85.00 S	200.0 S
Poly/sec	116.94	107.31	3.39
Shadowed	218.3 S	258.0 S	769.3 S
Poly/sec	41.78	35.35	0.88

Table 3: Performance comparison (Ray-traced)

4 Discussion and future work

4.1 Execution efficiency

Our preliminary T-Buffer implementation can be further optimized for better time and memory efficiencies. Two possible improvements are to perform scan-conversion in screen (pixel) space, rather than in object space in our current coding, and to employ orthographic projection, instead of perspective projection, to screen space.

4.2 Hardware acceleration

Hardware acceleration of T-Buffer can easily be realized. With some minor design modifications, an existing hardware Z-buffer circuitry can be converted into a T-Buffer engine. The changes involve adding the t' calculation circuit that computes eq. (5) (the second step in fig. (1)), and inverting the comparison/replacement rule for buffered values.

4.3 Multiple velocity system

To extend the T-Buffer method to simulate systems of objects with multiple velocities, it is necessary to consider multiple inertial frames S'_1, S'_2, ..., S'_n. Any object or group of objects which travels at a unique velocity with respect to imaging frame

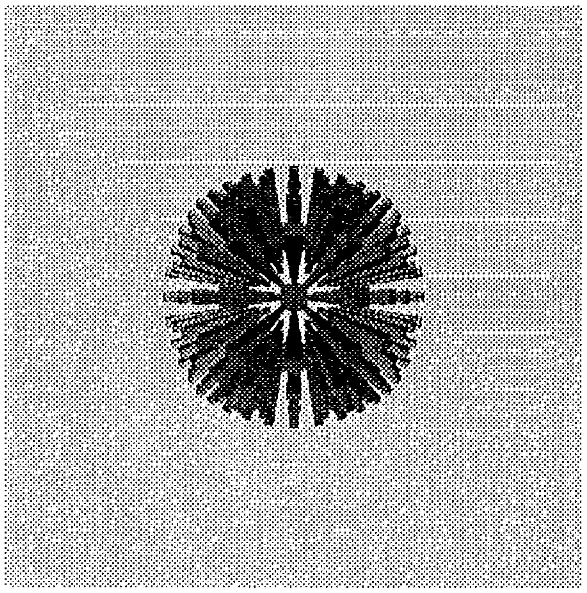

Figure 6: Bar array traveling towards viewer at 0.99c

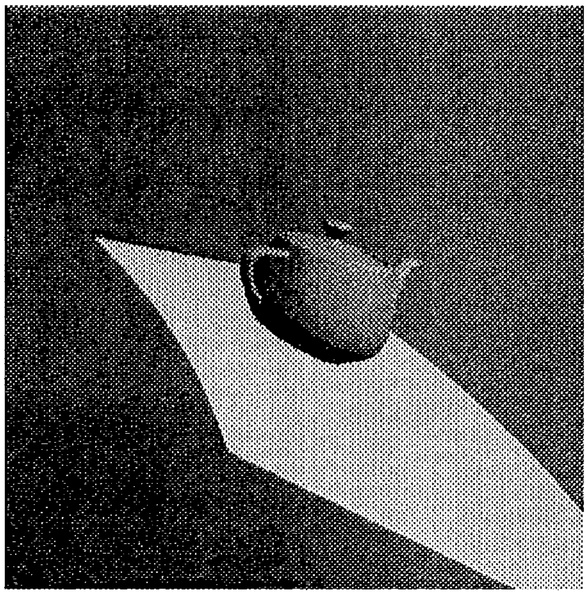

Figure 7: Teapot at $\vec{\beta} = (0.9, 0, 0)$

S may be placed in its own proper frame S'_i, and be related to S by a unique Lorentz Transformation L_i. The object space is thus divided into a series of co-existent inertial frames. All objects in every frame are scan-converted, and the events transformed to S to perform the final T-Buffer operation.

4.4 Features and limitation

The classical A-Buffer technique[1] that models translucency also can be adapted in T-Buffer. The limitation of T-Buffer is its inability to model specular reflection and refraction in optics.

4.5 Future work

We plan to test T-Buffer on a hardware graphics accelerator that we are constructing. At the heart of this accelerator is the new Intel processor *i860*[3]. We plan to explore the fast floating point processing facilities as well as the Z-buffer hardware support on this processor.

5 Conclusion

The *REST-frame* simulation technique fills in a void in past research, and provides one opportunity for exploring the historical fascination with visualizing Special Relativity effects that exist in physics as well as in many segments of our popular cultures.

Our previous ray-tracing based *REST-frame* implementation generated very high quality images that incorporated fine optical effects of reflection, refraction and shadow casting, but took long times to complete. In some time-critical applications, e.g. flight simulation, computer animation and video games, absolute realism and optical precision that this earlier *REST-frame* implementation offers is not essential. Rather, the emphasis is the speed of simulations — preferably at real-time.

The T-Buffer technique this paper presents provides a desirable solution to such applications by optimizing image synthesis speed at the expense of lower optical complexity in the resulting images. The advantage of T-Buffer over our previous implementation is twofold:

1. The software T-Buffer implementation runs over six times faster than the earlier *REST-frame* in favorable cases.
2. Furthermore, it can be mapped onto the well-developed Z-buffer based *rendering pipeline* that resides in most of the contemporary graphics workstations.

The availability of this latter hardware option makes the ultimate *real-time* simulation and animation of relativistic effects technologically feasible.

6 Acknowledgments

We would like to thank Robert H. P. Dunn and John Zsarnay for their generous assistance to our work. We also thank Kathryn Porsche for her insightful suggestions that contributed to this presentation. Finally, we are grateful for the valuable comments provided by our reviewers.

References

[1] L. Carpenter. The A-buffer, an antialiased hidden surface method. *Computer Graphics (SIGGRAPH)*, 103–108, 1984.

[2] E. Catmull. Computer display of curved surfaces. In *Proc. IEEE Conf. Computer Graphics Pattern Recognition Data Struct.*, page 11, May 1975.

[3] Intel Corp. *i860 64-Bit Microprocessor Programmer's Reference Manual.* Intel Corp., 1989.

[4] Andrew Glassner. *An Introduction to Ray-Tracing.* Academic Press Limited, 1989.

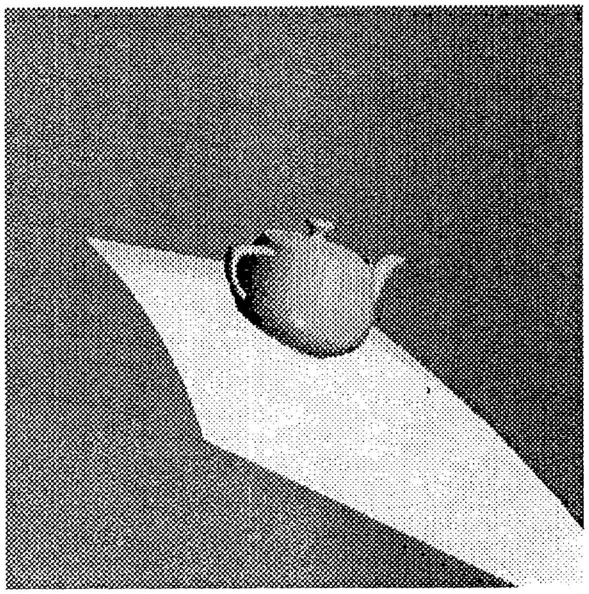

Figure 8: Teapot at $\vec{\beta} = (0.9, 0, 0)$ (Ray-tracing based simulation)

[5] Ping-Kang Hsiung and Robert H. P. Dunn. Visualizing relativistic effects in spacetime. In *Proceedings of the Supercomputing '89 Conference*, Nov. 13-17, 1989.

[6] Ping-Kang Hsiung and Robert H. Thibadeau. Spacetime visualization of 3D relativistic motion. *Unpublished document*, Oct., 1989.

[7] Ping-Kang Hsiung and Robert H. Thibadeau. Spacetime visualization of relativistic effects. In *ACM 1990 Computer Science Conference (to appear)*, Feb. 20-22, 1990.

[8] T.L. Kay and J.T. Kajiya. Ray tracing complex scenes. *Computer Graphics (SIGGRAPH)*, 269–278, Aug. 1986.

[9] C. Møller. *The Theory of Relativity*. Oxford University Press, 1960.

[10] N. Magnenat-Thalmann and D. Thalmann. *Image synthesis: theory and practice*. Springer-Verlag, 1987.

[11] Robert Resnick. *Introduction to Special Relativity*. Rensselaer Polytechnic Institute, 1968.

[12] G. D. Scott and M. R. Viner. The geometrical appearance of large objects moving at relativitic speeds. *American Journal of Physics*, 18(2):109–144, Jan 1965.

[13] E. Taylor and J. Wheeler. *Spacetime Physics*. M.I.T. / Princeton, 1966.

[14] Sten Yngström. Observation of moving light-sources and objects. *Arkiv för Fysik*, 367, 1962.

Color images for this paper can be found in the color plate section.

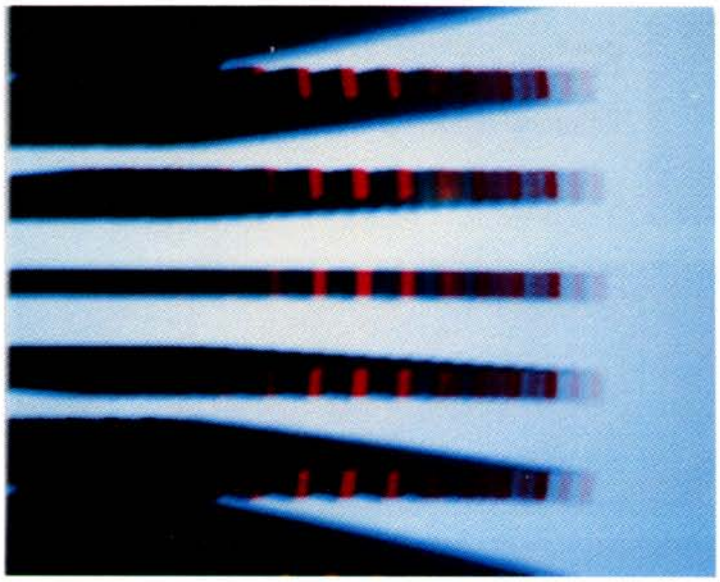

Plate 1: Bar array at $\vec{\beta} = (0.9, 0, 0)$, with 0.05 second shutter time

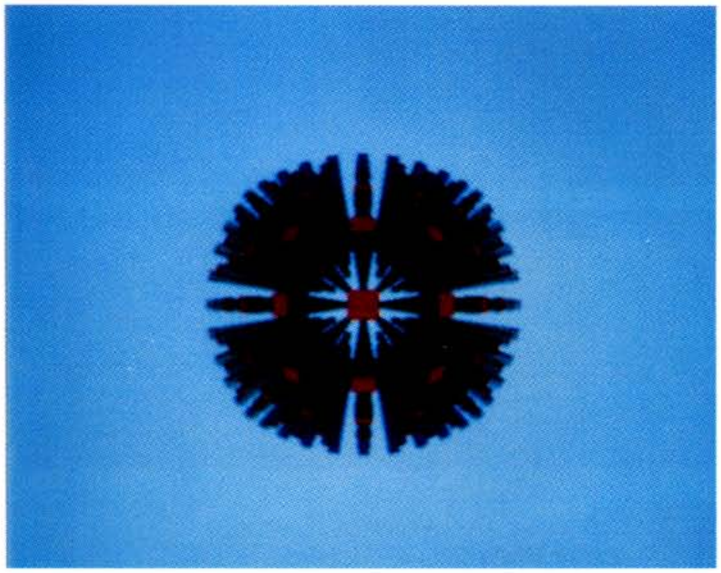

Plate 2: Bar array traveling towards viewer at 0.99c

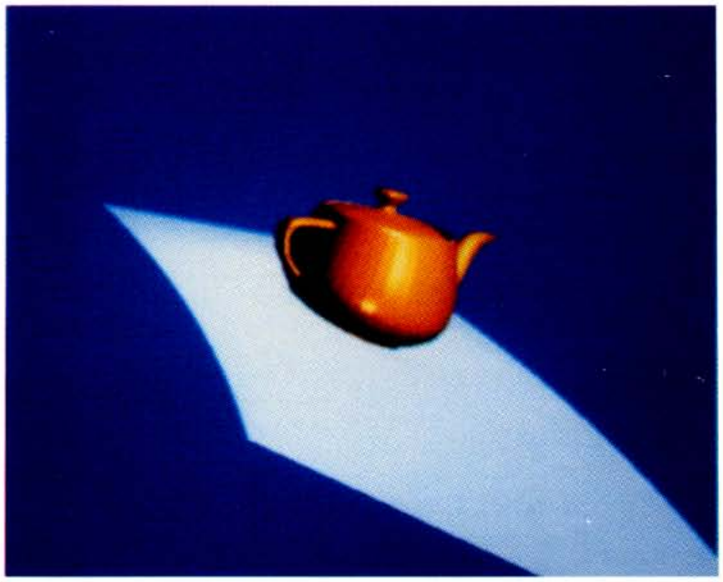

Plate 3: Teapot at $\vec{\beta} = (0.9, 0, 0)$ (Ray-tracing based simulation)

Hsiung, Thibadeau and Wu, "T-Buffer: Fast Visualization of Relativistic Effects in Spacetime".