On Coherence in Volume Rendering

Forschungsseminar Visualisierung
WS 98/99

Anna Vilanova i Bartrolí and Katja Bühler

February 1, 1999

Contents

Articles on Coherence in Volume Rendering

- General

  [Yagel '96],[Groeller, Purgathofer '95]

- Pixel Space Coherence (Katja)

  [Bergman et al. '86],[Levoy '90b]

- Object Space Coherence (Katja)

  [Danskin, Hanrahan '92] extension of [Laur, Hanrahan '91](Space Leaping) and [Levoy '90a](Splatting) All Algorithms are based on preprocessing the data storing them in octrees together with certain information about the homogenity of the volumetric area corresponding to each node of the tree.

  Proximity Clouds: [Yagel, Shi '93].

- Frame-to-Frame Coherence (Anna)

  [Brady et al. '97],[Brady et al. '98],[Yagel, Shi '93],[Gudmundsson, Randen '90]

- Inter Ray Coherence/Shear-Wrap (Anna)

  [Lacroute, Levoy '94],[Yagel, Kaufman '92]

- Stereoscopic Images (Anna + Katja)

  [Adelson, Hansen '94],[He, Kaufman '96a]

- Wavelets (Katja)

  [Sánchez, Carvajal '98]

- others

  [Avila et al. '94],[Cohen-Or, Fleishman '95],[Mueller, Yagel '96](Splats, no coherence)

1 Introduction

1.1 Definitions of coherence

- Meyers Online-Lexikon:
  Kohärenz, das Vorhandensein genau bestimmter Beziehungen zw. den Phasen sich überlagernder (interferierender) Wellenzüge; v. a. Laser liefern eine Strahlung bes. hoher Kohärenz.

- From Webster's Revised Unabridged Dictionary (1913)
  Coherence, Coherency, n. [L. cohaerentia: cf. F. coh['e]rence.]

  1. A sticking or cleaving together; union of parts of the same body; cohesion.

  2. Connection or dependence, proceeding from the subordination of the parts of a thing to one principle or purpose, as in the parts of a discourse, or of a system of philosophy; consecutiveness.

  Coherence of discourse, and a direct tendency of all the parts of it to the argument in hand, are most eminently to be found in him. –Locke.

- From WordNet (r) 1.6
  coherency, n.

  1: the state of cohering [syn: coherence, cohesion, cohesiveness] [ant: incoherence]

  2: logical and orderly and consistent relation of parts [syn: coherence]

- "Meister"-Definition:
  Coherence denotes similarities between items or entities

1.2 Motivation

Volume rendering is still a very time consuming process and the cost is multiplied if we have to generate more than one picture like in generation of stereoscopic images or animations. A way to accelerate the rendering process is to take advantage of coherences that may occure in several forms: e.g. homogeneous areas in voxel space or the similarities of two consectutive images of an animation.

1.3 Definitions of different types of coherence relevant for volume rendering

- **Pixel Space Coherence:** Coherence between pixels in image space.

  Acceleration of image processing taking advantage of the asumption that between two similar shaded pixels ly only pixels with similar color. Thus, coherent pixels can be computed with interpolation.

- **Object Space Coherence:** Coherence between voxels in Object space (e.g. empty areas)

  Avoid sampling of in 3D regions having uniform or similar values: Detection of this areas by building up octrees. Samples located in a coherent area are calculated only once. "It is obvious that empty space does not have to be sapmled - its has only to be crossed as fast as possible" (Space leaping) [Yagel '96]

- **Inter Ray Coherence:** In parallel projection all ray have the same slope, thus only a template of a discretisation of such a ray is generated by a 3D line voxelization algorithm.

- **Frame-to-Frame Coherence:** The coherence between successive images in animations.

  Exploiting the coherence of empty space in conscutive images, the ray path can be shorten.

1.4 The given volumetric data and the raycasting algorithm

The volumes are already illuminated and we are left with the task of rendering the 3D grids of RGBα values.

Brute-force Algorithm [Levoy '90a]

Given a scalar valued 3D array of data samples of N^3 Voxels. A Voxel is defined by its position $\boldsymbol{i} = (i, j, k)$, $i, j, k = 1, ..., N$ its value $f(\boldsymbol{i})$ (???), its color $C(\boldsymbol{i})$ and the opacity $\alpha(\boldsymbol{i})$.
The image is assumed to be an array of P^2 Pixels indexed by a vector $\boldsymbol{u} = (u, v)$, $u, v = 1, ..., P$.

Parallel rays are traced into the data from each pixel and a vector of colors and opacities is computed by resampling the data at W evenly spaced locations along the ray and by trilinearly interpolating from the eight voxels surrounding each sample location. Samples are indexed by a vector $\boldsymbol{U} = (u, v, w)$, where (u, v) identifies the ray (the location of the pixel the ray was cast from) and $w = 1, ..., W$ corresponds to the distance along the ray with $w = 1$ being closest to the eye. The Color and opacity of sample $\boldsymbol{U}$ are denoted by $C(\boldsymbol{U})$ and $\alpha(\boldsymbol{U})$ respectively. A fully opaque background is draped behind the dataset and the resampled colors and opacities are composited with each other and with the background to yield the color for the pixel $C(\boldsymbol{U})$.
Let $C_{out/in}(\boldsymbol{u}; \boldsymbol{U})$ and $\alpha_{out/in}(\boldsymbol{u}; \boldsymbol{U})$ denote the color and opacity of pixel $\boldsymbol{u}$ after/before processing sample $\boldsymbol{U}$.

In a front-to-back algorithm color and opacity are calculated in the following way:

$$\hat{C}_{out}(\boldsymbol{u}; \boldsymbol{U}) = \hat{C}_{in}(\boldsymbol{u}; \boldsymbol{U}) + \hat{C}(\boldsymbol{U})(1 - \alpha_{in}(\boldsymbol{u}; \boldsymbol{U}))$$

$$\alpha_{out}(\boldsymbol{u}; \boldsymbol{U}) = \alpha_{in}(\boldsymbol{u}; \boldsymbol{U}) + \alpha(\boldsymbol{U})(1 - \alpha_{in}(\boldsymbol{u}; \boldsymbol{U}))$$

with $\hat{C}_*(*) = C_*(*)\,\alpha_*(*)$.
After all samples along a ray have been processed, the color $C(\boldsymbol{U})$ of the ray is given by

$$C(\boldsymbol{U}) = \frac{\hat{C}_{out}(\boldsymbol{u}; \boldsymbol{W})}{\alpha_{out}(\boldsymbol{u}; \boldsymbol{W})}$$

2 Pixel space coherence

"It is highly probable that between two pixels having identical or similar color we will find another pixel having the same or similar color." [Yagel '96]

2.1 General strategy

Avoid sending of rays for obvious pixels generating colors of "empty" Pixel by interpolation.

2.2 Technics exploiting pixel space coherence

Adaptive image supersampling in

Raytracing [Bergman et al. '86]

Anti-aliasing is performed only if a given threshold of variance of the eight neighbored pixels is exeeded.

Volume rendering [Levoy '90b]

An initial image is generated by casting a uniform but sparse grid of rays into the volume data, interpolating between the resulting colors and resampling at the display resolution. A sampling rate of one ray per four pixels corresponding to a sampling rate of one ray per four voxels, is typical. Subsequent images are generated by discarding interpolated colors, casting more rays and repeating the interpolation and resampling steps. Recursive subdivision and based on color differences is used to concentrate these additional rays in regions of high image complexity and recursive bi-linear interpolation is used to to form images from the resulting non-uniform array of colors.

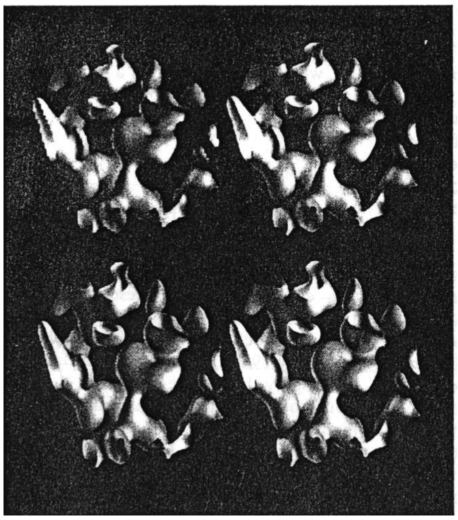
The cost of computing each image in a refinement sequenz is equal to the sum of the costs of recursive subdivision, ray tracing and recursive interpolation.

Algorithm: The voxel array the following algorithm is applied to is already shaded and classified.

1. The image plane is divided into square sample regions measuring ω_{max} pixels on a side.
2. Parallel viewing rays are cast into the voxel array from the four corner pixels of each region.
3. Each ray is devided into equal-size sample intervals and color and opacity is computed at the center of each interval by tri-linear interpolation of the colors and opacities of the nearest eight voxels.
4. The resampled colors and opacities are then composited to yield a color for the ray.
5. If the range of colors returned by the four rays in a sample region is less than some ϵ, no further processing is performed on the region, otherwise the region is divided into four subregions and more rays are cast.
6. Subdivision is continues until the range of colors falls below ϵ or the size of the region reaches some ω_{min}.
7. An image is formed by interpolating between the available colors (to ensure continuity): The image plane is devided again into square regions measuring ω_{max}. Pixels are interpolated at the midpoints of the four sides and at the center of each region. The region is then subdivided into four subregions and the process is repeated. Subdivision continues until the region contains only one pixel.
8. To refine the image all interpolated pixels are cleared, the level of detail is increased by decreasing ω_{min},ω_{max} and/or ϵ and repeating the the algorithm. To avoid that the same rays are cast several times a flag is maintained for each pixel.

Problems: High values of ω_{max} cause features to missed, high values of ϵ cause features to be ignored even if they are not missed and high values of ω_{min} cause features to be poorely resolved even if they are neither missed or ignored. Inapropriate values for these parameters cause suboptimal presentation of the data as well as unequal intervalls between successive frames in refinement sequenzes. Algorithms are needed that automatically select an optimum sequenz of values based on the characteristics of a particular dataset.

Results:



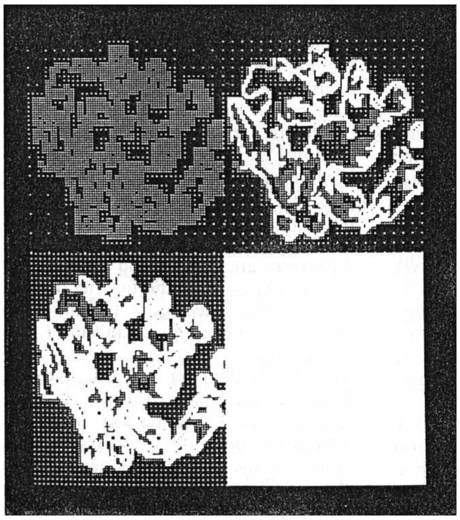


Table 1. Performance statistics for cytochrome

Frame	Image (Fig.)	Rays (Fig.)	Total rays	Additional rays	(%)	Total time (s)	Time interval (s)	ω_{max}	ω_{min}	r
1	3a	4a	3870	3870	20	4.6	4.6	16	2	32
2	3b	4b	9947	6077	51	12.1	7.5	8	1	32
3	3c	4c	14459	4512	74	17.1	5.0	4	1	16
4	3d	4d	19289	4830	100	25.7	8.6	1	1	

3 Object space coherence

The optimization is based on the observation that many datasets contain coherent regions of empty voxels ([Levoy '90a]).

Methodes for encoding object space coherence:

- octree hierarchical spatial enumeration ([Levoy '90a])
- polygonal representation of bounding surfaces
- octree representation of bounding surfaces

3.1 Fast traversal of empty areas [Levoy '90a]

Optimization by hierachical spatial enumeration

Given a voxel dataset like in 1.4 with $N = 2^M + 1$, $M \in \mathbb{N}$.

1. Data structure

   Voxels are treated as points and are the vertices of cells filling the space between voxels. Cells in this dataset are indexed by a a level number $m = 0, ..., M$ and a vektor $\boldsymbol{i} = (i, j, k)$, $i, j, k = 1, ..., N-1$. The value contained in a cell $\boldsymbol{i}$ on level m is denoted $V_m(\boldsymbol{i})$. The size of cells on level m is defined to be 2^m times the spacing between voxels.
   The pyramid is constructed as follows:

   - $V_0(\boldsymbol{i}) = 0$ if all its eight vovels lying on the vertices have opacity equal to zero.
   - $V_m(\boldsymbol{i}) = 0$, $m > 0$ if all eight cells on level $m-1$ that form its octants are zero.
   - otherwise its value is one.

2. Optimized algorithm

   For each ray

   - compute the cell $V_0(\boldsymbol{i})$ which is hit by the ray at top level 0
   - if $V_0(\boldsymbol{i}) = 0$ advance along the ray to the next cell on the same level
   - if the parent of the new cell differs from the parent of the old cell, move up to the parent of the new cell.
   - if the parent does contain zero advance on its level, otherwise go back to the childs level.
   - if one of the cells $V_0(\boldsymbol{i})$ is one, draw samples (???) at evenly spaced locations along that portion of the ray falling within the cell, resample the data at these sample locations and composite the resulting color and opacity into the color and opacity of the ray.

   This opimization allows to advance very fast cross empty areas.

3.2 Progressive refinement 1[Laur, Hanrahan '91]

Progressive refinement via a pyramidal volume representation. The octree is drawn using a set of splats each scaled to match the size of the projection of a cell. The splats are approximated with RGBA Gouraud-shaded polygons that can be easily (and correct!) scaled. (Big section in the article about "correct" splats)

3.2.1 The pyramidal representation

Here "pyramide" indicates a full resolution set, "octree" some subset of the pyramide.
Every node of the pyramide contains

- the average RGBA value of all its children
- the average deviation error e^l_j approximating the region of space associated with the node with a constant function equal to the node's RGBA value:

$$e^l_j = \sqrt{\sum_{i=1}^{n^l_j} s_i^2 + (\sum_{i=1}^{n^l_j} s_i)^2}$$

  e^l_j can be computed by a single traversal of the pyramid.

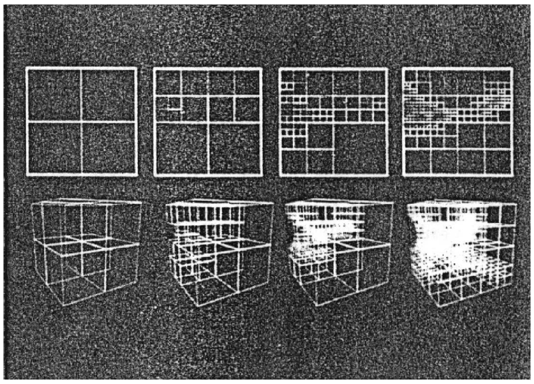
3.2.2 The progressive refinement

Let e be a given deviation error.

The pyramid will be traversed in the following way:

- At each level in the pyramid, e is compared to the average deviation e^l_j of the node.
- If $e^l_j < e$ the traversal is terminated, otherwise the traversal proceeds downwards.

The result is an octree, which is a subset of the pyramide and whose leaves are the nodes of the pyramide with deviation error less than e.
Thus, we have higher resolution in "complicated" areas, than in more homogeneous ones.



3.3 Progressive refinement 2 [Danskin, Hanrahan '92]

3.3.1 Pyramid data structures

- **The average pyramid:**
  Each node in the average pyramid stores the average of its eight children. A level 0 voxel in the average pyramid is the cubic region supported by 8 data values and contains their average value.

- **The $max_{27}(\alpha)$ and the $min_{27}(\alpha)$ pyramide:**
  Each leave of the $max_{27}(\alpha)$ pyramide stores the average of the maximum value of the corresponding data point and of its 26 neighbours.
  Each node on higher levels contains the maximum value of its child and its 26 neighbours.
  The $min_{27}(\alpha)$ pyramide works analogue.

- **The $range_{27}$ pyramid:**
  Each voxel contains the Manhattan distance

  $$NYNYdist(a,b) = \frac{|a.r - b.r| + |a.g - b.g| + |a.b - b.b| + |a.a - b.a|}{4}$$

  between the scaled RGBA value derived from sampling the $min_{27}(\alpha)$ and $max_{27}(\alpha)$ pyramide.

  Whenever this value is small the region is nearly homogeneous.

- **The $variance_{27}$ pyramid:** see 3.2.1.

The $range_{27}$ pyramide has a better worst case behaviour as the $variance_{27}$ pyramid: A large voxel high up in a pyramide may be mostly empty, containing only a very thin yet totally opaque wall and yet have low variance because of the samll amount of variant pixels. This could lead to dramatic image artifacts. The $range_{27}$ pyramid cannot make this kind of mistakes.

3.3.2 Applications

- **Presence Acceleration**
  Detect large (nearly) empty areas taversing a $max_{27}(\alpha)$ pyramid (determing when $\alpha <$ as a given value), fill this regions with average values from the average pyramid.
  problems: it ignores accumulated opacity and it moves up the pyramide too fast.

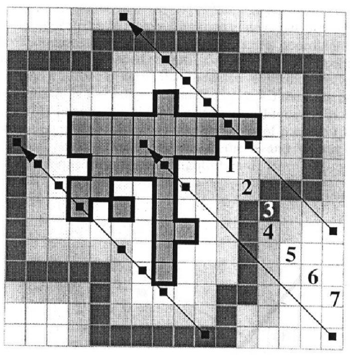
- **Homogeneity Acceleration**
  Like Presence Acceleration but with a $range_{27}$ pyramid.

3.4 Proximity clouds [Yagel, Shi '93]

This approach allows fast traversal of empty space, without building up an octree.

The algorithm

1. Compute in a preprocessing step for each voxel the distance to the closest occupied voxel.
2. If a ray casted into the volume hits a voxel carrying the value n it can make a leap over the next n voxels without missing an occupied one.



Advantage

"This approach saves the need to perform a tree search for the appropriate neighbour - an operation that is the most time consuming and the major disadvantage in the hierarchical data stucture."[Yagel '96]

3.5 Detection of areas with constant, linear and non-linear behaviour via Wavelt decomposition [Sánchez, Carvajal '98]

General propositions

Two ways of visualizing 3D data sets:

- extracting isosurfaces from the volumetric data by casting rays from the viewer's eye through the screen into the volume and finding the intersection of the surface with the dataset.
- direct visualization of a semi-transparent and self-iluminated volume.

Interpolation is necessary in both algorithms.

The algorithm

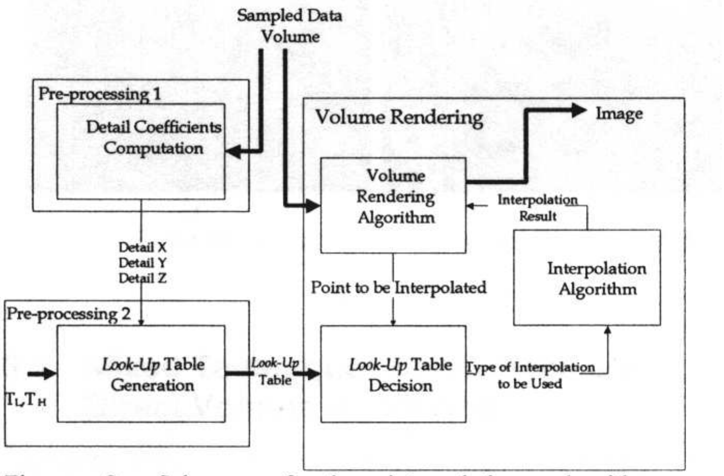


Figure 8: Scheme of the interpolation algorithm implementation.

Application

The trilinear interpolation of color and opacity of a sample within a voxel is replaced by adaptive interpolation: depending on the homogenity of the surrounding voxels, the interpolation will be constant, linear or non-linear. The different areas are detected via Wavelet decomposition: If the detail projection of the Wavelet decomposition of the analysed area with respect to a certain Basis is zero, the areas has constant, linear or non-linear behaviour depending on the selection of basis functions.

4 Coherence in stereoscopic visualization

4.1 General

Both halves of a stereoscopic stereoscopic pair of ray-traced volumetric images need not to be fully rendered. Exploiting the coherence between two views, the second half of the stereo pair can be generated in a fraction of the time of the first half.([Adelson, Hansen '94])

A fast stereo rendering algorithm can be devided in three steps ([He, Kaufman '96a]):

1. Select an appropriate viewpoint position (e.g. left-eye, right-eye or the middle-eye)

2. Apply a chosen standard rendering scheme for the selected viewpoint
3. Reproject the results or intermediate results of step 2 onto the right or left image or both according to certain coordinate transformations.

To measure the performance of a fast stereo rendering algorithm the three criterias are proposed:

1. Minimum total time for generating the left- and the right-side image.
2. High image quality
3. Balanced image quality between the two images. (It is better to generate both images in medium quality than generating one of them in high accuracy and the other one in low accuracy.)

4.2 Ray-traced volume rendering [Adelson, Hansen '94]

4.2.1 Stereoscopic reprojection

There are two view points (centers of projection of the respective left-/right eye image), seperated by a distance $\boldsymbol{e}$. One of the view points is positioned at the origin and the other at the point $(\boldsymbol{e}\cos(\frac{1}{2}\Phi), 0, \boldsymbol{e}\sin(\frac{1}{2}\Phi))^T$.[1] To accomplish that, the volumetric data is tranformed in the following way:

- Place the origin of the viewing coordinate system at the middel of the line between the two view points at $(\frac{\boldsymbol{e}}{2}\cos\frac{\Phi}{2}, 0, \frac{\boldsymbol{e}}{2}\sin\frac{\Phi}{2})^T$.
- Set the w-axis of the (u, v, w) viewing coordinate system to $(-\sin\frac{\Phi}{2}, 0, \cos\frac{\Phi}{2})$.

This is done by a rotation in the x-z-plane around the point $\boldsymbol{P} = (0, 0, \frac{\boldsymbol{e}}{2\sin\frac{\Phi}{2}})$.

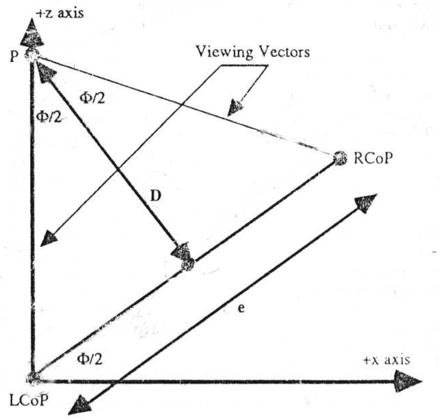


Figure 1: Displaced Parallel Projection Geometry. Rotating and translating the centers of projection - as opposed to the standard symmetrical placement about the Z axis - results in computational savings during projection.

The volumetric data has now the "view-transformed" coordinates (x_p, y_p, z_p).

Now the left-eye projection is simply a parallel projection based on the data position after the viewing transformation.

The right-eye projection involves

- a rotation of $-\frac{\Phi}{2}$ about the y-axis
- a translation of $(-\boldsymbol{e}, 0, 0)$, to place the right view point to the origin
- a second rotation about the y-axis to place $\boldsymbol{P}$ back on the z-axis.

The view-transformed point (x_p, y_p, z_p) projects onto (x_{sl}, y_{sl}) and (x_{sr}, y_{sr}) as follows:

- Left eye-image:

$$x_{sl} = x_p, \qquad y_{sl} = y_p$$

[1] Traditional the viewpoints are placed symetrically about the origin.

- Right eye-image:

$$x_{sr} = x_p \cos\Phi + z_p \sin\Phi - \boldsymbol{e} \cos\frac{\Phi}{2}, \qquad y_{sr} = y_p$$

or (approximating $\cos\Phi \approx 1$ and $\sin\Phi \approx \Phi$ for small Φ)

$$x_{sr} = x_{sl} + z_p\Phi - \boldsymbol{e}, \qquad y_{sr} = y_p$$

4.2.2 Ray traced volume rendering

Equidistant samples of a "left-eye" ray correspond to samples of equidistant "right-eye" rays: The x_{sr}-value will increase at each sample by Δz ($\Delta z \cos\Phi$), where Δz denotes the distance of two samples of the "left-eye" ray.

If the left-eye rays are processed sequenzially along scan-lines, the x_{sr}-projection values from two consecutive left-eye rays with distance Δx will equally differ by the constant Δx.

To accelerate the computation, left-eye samples are accumulated only until full opacity is reached. To avoid that right-eye rays are terminated too early, the accumulated opacities of the right-eye samples are re-examined. If the accumulated opacity of a ray is not 1.0 and whose last projection is not on the far side of the volume bounding box, sampling of this right-eye ray is continued.

Problem of this Algorithm[He, Kaufman '96a]: The sampling points of this algorithm are not reprojected onto the integer grids of the right image. Round the re-projected samples to their nearest integer value. As a result errors are introduced.

4.3 Fast stereo volume rendering [He, Kaufman '96a]

4.3.1 Segment composition

Usually the angle between the left and right rays Φ is small ($\Phi < 1.5$ degree is recommended). As a result series of sampling points along a single left-ray could be reprojected onto the same right-eye pixel.

Assuming that a sampling point (x_p, y_p, z_p) is projected to (x_{sl}, y_{sl}) of the left-image and (x_{sr}, y_{sr}) of the right image and the distance between the left and the right center of projection is e. If we assign

$$c = x_p \cos\Phi - e\cos\frac{\Phi}{2}$$

then for a left ray, cast from (x_{sl}, y_{sl}), c is a constant, and

$$x_{sr} = round(c + z_p \sin\Phi)$$

Assume that

$$c + z_p \sin\Phi = n \in \mathbb{N}$$

then for all sampling points on the ray in the interval

$$[(x_p, y_p, z_p - \frac{1}{2\sin\phi}), (x_p, y_p, z_p + \frac{1}{2\sin\phi})]$$

we have

$$x_{sr} = n.$$

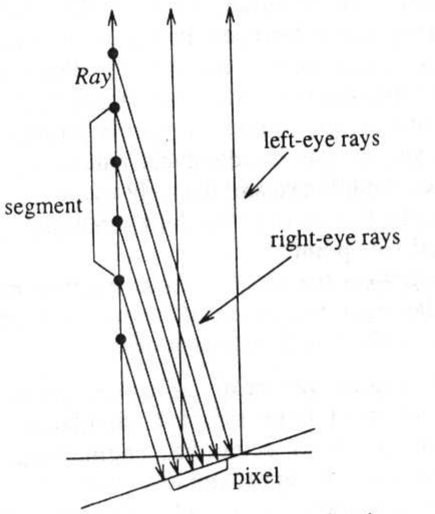


Figure 1: *Segment re-reprojection.*

Thus, instead of reprojecting the sampling points on the ray one by one segments of the ray with lenght $1/\sin\Phi$ can be reproject and composite the color and opacity of the segments onto a pixel on the right image:

Suppose that α is the opacity, $t = 1 - \alpha$ the translucency, the accumulated color and translucency C_{in} and t_{in} resp. at the entry point of a certain segment. (C_{out} and t_{out} resp. at the exit point). The translucency t_{seg} and the color C_{seg} of this segment can be calculated as

$$t_{seg} = \frac{t_{out}}{t_{in}}$$

$$C_{seg} = \frac{C_{out} - C_{in}}{t_{in}(1 - t_{seg})} = \frac{C_{out} - C_{in}}{\alpha_{out} - \alpha_{in}}$$

and the composition on the reprojected pixel is

$$C_{pixel} = C_{pixel} + C_{seg} t_{pixel} (1 - t_{seg})$$

$$t_{pixel} = t_{pixel} t_{seg}$$

Solution of the early ray termination problem:

The left ray is not terminated if a certain treshold is reached: the sampling within the current segment is terminated then we jump to the entry point of the next segment which corresponds to the next right eye pixel and reproject the sapmling points inside that segment. The same process is applied to all segments that correspond to non-opaque pixels in the right eye image. But the sampling points behind left ray termination are reprojected one by one.

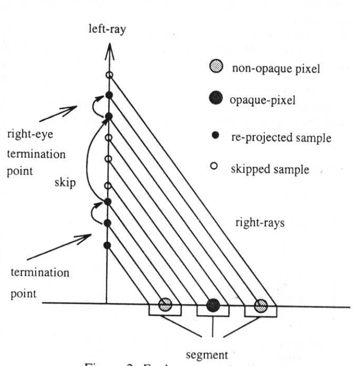


Figure 2: *Early ray termination.*

4.3.2 Linearly-interpolated re-projection

In section 4.3.1 constant interpolation is used for reprojection. For a better right-eye image, an algorithm for linear interpolation is given as follows:

For each n

1. process the nth sampling point on the most right left-eye ray. Save Color, opacity, the affected x-coordinate and the weight(???? arbitrary????) in the nth element of a 1D array RAY.

2. For each remaining left-eye ray from right to left do

   (a) For the nth sampling point on the ray generate the right-eye sampling values as

   $$RSample(c, \alpha) = Sample(c, \alpha)(1 - RAY[n].weight) + RAY[n].(c, \alpha)RAY[n].weight$$

   (b) Reproject the right-eye sampling points onto $RAY[n].pixel$.

   (c) Set

   $$RAY[n].weight = 1 - RAY[n].weight \quad (1)$$
   $$RAY[n].pixel = RAY[n].pixel - 1 \quad (2)$$
   $$RAY[n].(c, \alpha) = Sample(c, \alpha) \quad (3)$$

4.4 Results

- Generation of the right-eye image without segment composition:
  Time saving: 70-80%
- Generation of the right-eye image with segment composition:
  Time saving: around 97%

5 Frame-to-Frame Coherency

5.1 Fast perspective projection volume navigation [Brady et al. '97] [Brady et al. '98]

In this paper they are using Frame-to-Frame coherence and another rendering method. Based in a previous work that was just for parallel projection. It is useful for external views since the distance from the viewer to the data is relatively large but it is not useful to go inside the volume where the viewer is inside the data.

Previous statements It is supposed that the volume is isotropic. The sampling distance is σ. All the units in the paper are written in units of the voxel spacing σ.

The view frustum :

- Size of the view frustum $\ll$ volume.
- View frustum position and orientation is defined by position p and with an orientation $\vec{v}$.
- Depth of view is fixed d.
- An arbitrary path is traced through the volume and is defined by a discrete sequence of viewing sites $(p_i, \vec{v_i})$ where $i \in \aleph$ whose values are not known in advance.
- The path is sequence of incremental steps in a direction or rotations about an axis.

The coherence between nearby frames f is used.
The results from the computation of a given view are used to compute f successive frames, after which a refresh view must be produced. **Sampling** is obtaining the color and opacity of a point in the ray. (Interpolation, Transfer functions mapping, and shading or light calculation).
Composition is Volumetric composition equation in a front to back.
The algorithm is divided in 2 phases.

Phase 1 *(figure 1)*: Short ray segments are cast composite color computing the composite color and transparency for each segment.
Assumed that the sampling rate is σ for simplicity.

- The sample points are divided into L levels where $0 \leq l < L$ based on the distance from the viewpoint.
- D_l distance of the first sample of level l.
- Each level l consist of a set of ray segments $S_l(p, \vec{v})$ or S_l that treat the samples from D_l to $D_{l+1} - 1$ *($D_{l+1} - D_l$ samples)*.
- $S(l, \vec{v})$ or $S = \cup S_l(p, \vec{v}) \forall l$
- Result phase 1 L two-dimensional arrays.
- Each level can be sampled at different resolutions.
  $W_l \times H_l$ horizontal and vertical resolutions of the segments casts in level l.
  With that they can get *adaptive sampling* to avoid undersampling of distance points. Trying to maintain similar lateral sampling.
  Avoid oversampling in the regions near the viewpoint. The few levels near the view point cast very few rays /it Acceleration.

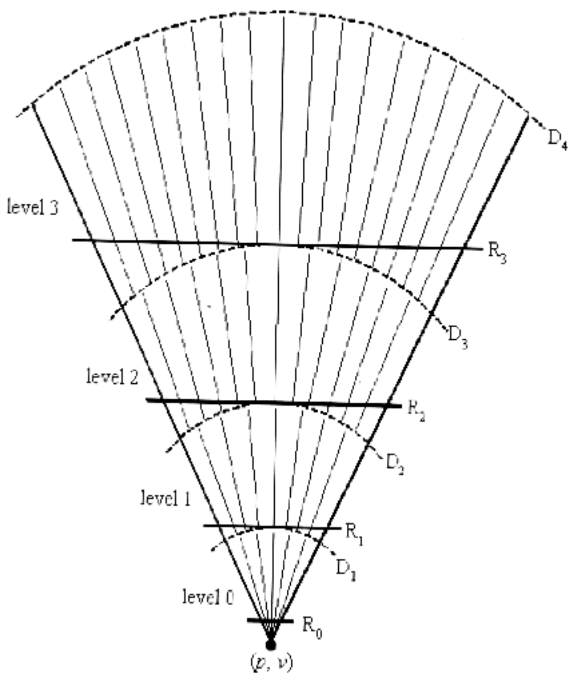


Figure 1: 2D illustration of the segments computed by the two-phase algorithm. The horizontal lines represent the rectangles into which the segments are to be mapped

Phase 2 Construct approximations of the full rays in the second phase.

- $\forall l$ do from $W_l \times H_l$ to Screen resolution.
- alpha blending

Done using 2D-texture-Mapping T_l *for each level* (acceleration hardware).

- Define a rectangle for any level R_l
- Drawn R_l with T_l mapped at distance D_l perpendicular two the view direction and from front to back and Alpha blending.
- Special treatment of R_0 Drawn between 0 and D_1.

or from $l = 0$ to $l = L - 2$ resampling level l from $W_l \times H_l$ to $W_{l+1} \times H_{l+1}$ and composing with level $l + 1$. *not used*

Performance

- Brute force raycasting is an special case of two-phase algorithm in which $W \times H$ are cast to Depth d in a single level.
- 10 Levels and 0.385 times as many sample steps as the brute force approach. See table 1 in *figure 3*.
- Composing operation is nonlinear reordering interpolation and the composite steps will incur some error. See *figure 2* to do a visual comparison between brute force and two phase ray casting.

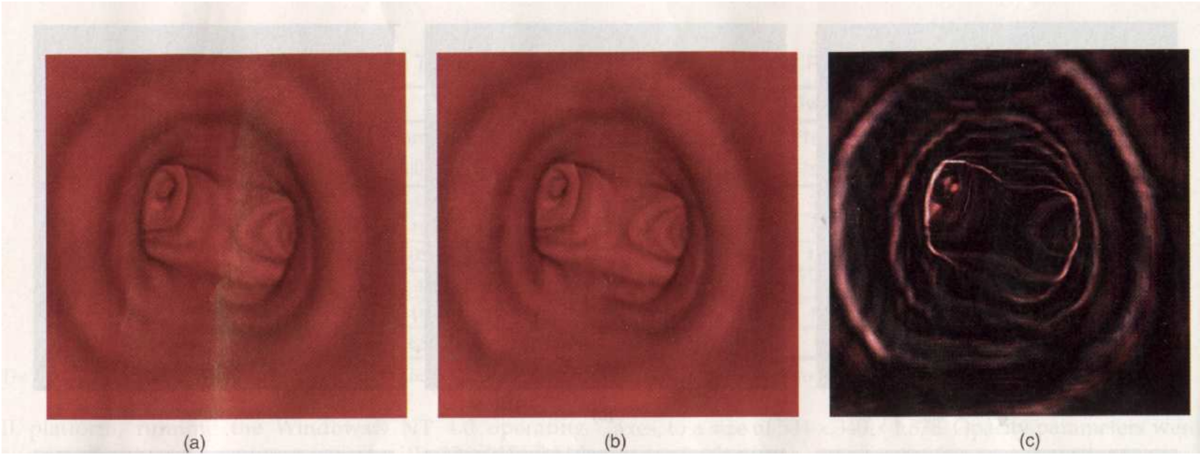


Figure 2: Rendering comparison: **a)** Brute force perspective raycasting. **b)** The two phase algorithm. **c)** Difference between the two multiplied by 10.

TABLE 1
COMPARISON OF RENDERING TIMES FOR TWO-PHASE VS. BRUTE-FORCE PERSPECTIVE RAYCASTING

			Brute-force		Two-phase (L=10)			
Data type	D	W, H	Samples (millions)	Time (sec.)	Samples (millions)	Phase 1 (sec.)	Phase 2 (sec.)	Total (sec.)
32-bit RGBL	160	160	4.1	5.54	1.8	2.43	0.04	2.47
		320	16.4	21.72	6.3	8.96	0.17	9.13
		480	65.5	48.49	25.2	19.67	0.31	19.98
8-bit gray	160	160	4.1	3.54	1.8	1.44	0.04	1.48
		320	16.4	13.30	6.3	5.75	0.17	5.92
		480	65.5	29.74	25.2	13.0	0.31	13.31

Figure 3: Tables of times and comparisons

Interactive Navigation : Phase 1 can be reused to approximate frames from viewpoint near the base position *(the position from where the phase 1 was calculated).*

- δ radius from the base position.
- θ angle of the original viewing direction.
- λ levels that need to be recalculated in every step. So reused $L - \lambda$.

To avoid jerky motion due to the periodic pause to fresh the samples when the view position is out of (δ, θ). New position of refresh can be estimated *(long translations in a direction or rotations about an axis)* and spread it during the frames calculation.

Viewpoint Translation. *(See figure 4).given* $S(p, \vec{v})$

- Δp is the position increment $f = \frac{2\delta}{\Delta p}$.
- $S(p + 2\delta, \vec{v})$ next predicted structure. Computing $\frac{1}{f}$ of it in any step.

- The first λ levels move with viewpoint.
- The depth of levels $\lambda - 1$ varies by the amount δ.

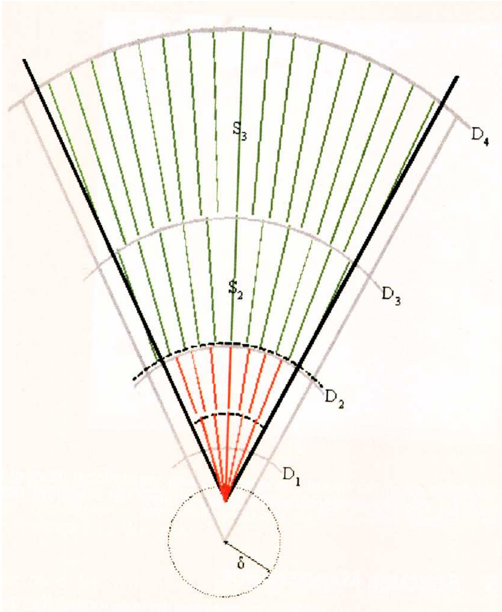


Figure 4: 2D description of the forward translation

- For each step one plane of samples should be rendered onto the back of the last level so that the total viewing depth remain the same.
- Calculate an slightly larger field of view for the positions behind $(p, \vec{v})$ the frustum.
- Fixed Δp. Higher f less quality.

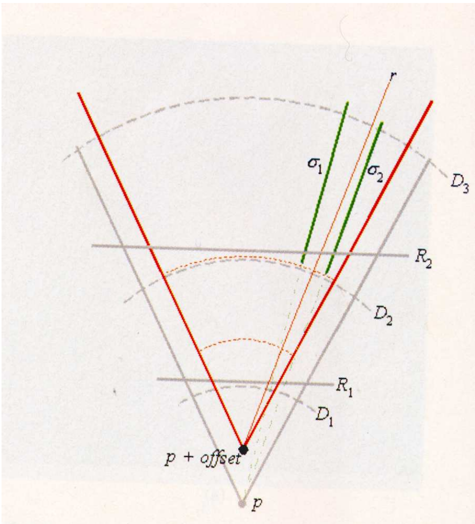


Figure 5: 2D illustration of the resampling of S during translation

PROBLEM It implies bigger δ and this implies the orientation of the S on the periphery of the view do not match.*(See figure 5)*
The new segment s is bilaterally interpolated by the 4 segments nearest to s which is not parallel to the 4. Which are the weights distance of the interpolation?
SOLUTIONS Average Distance or the resampling depends upon relative distance within the plane of the rectangle. Towards the edges the weighting is closer to the middle of the segments. In the middle distances are based on the position of the front.

Viewpoint Rotation *(figure 6)given* $S(p, \vec{v})$

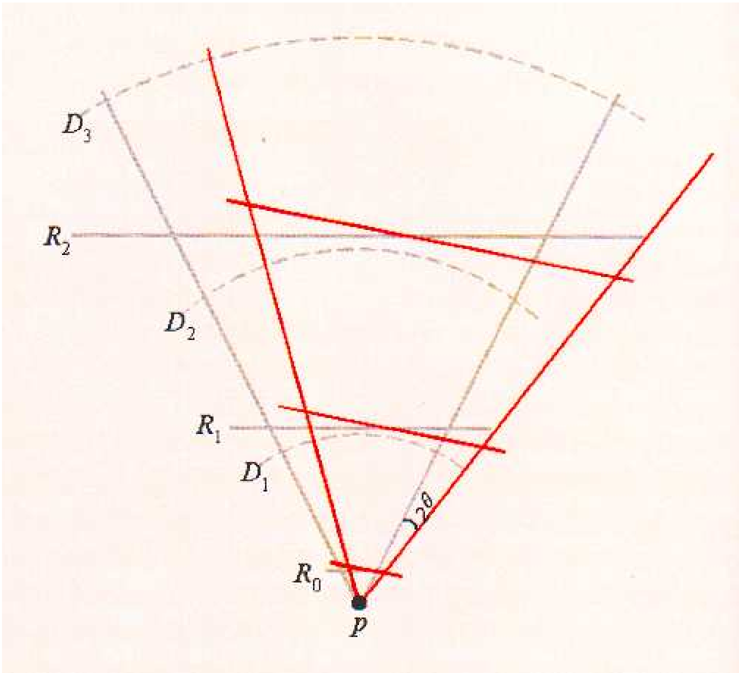


Figure 6: The initial segment data structure is depicted in gray and should be 2θ wider that the field of view

- Δv incremental in angle rotation in a axes.
- the size of the viewing frustum in the direction of rotation is $\frac{f\Delta v}{2}$ wider.
- $S(p, plus(\vec{v}, f\Delta v, axis))$ is the next predicted structure.
- levels from 0 to $\lambda - 1$ don't need to be recomputed at each step.
- More resampling accuracy than translation. No error in the orientation of the segments. Relative distance are well defined.

Times See table 2 in *figure 7)* in order to compare time savings.

TABLE 2.
COMPARISON OF RENDERING TIMES FOR COMPLETE TWO-PHASE PERSPECTIVE RAYCASTING WITH THE TIMES FOR INCREMENTAL FORWARD MOVEMENT AND INCREMENTAL ROTATION

$L = 10$, $\lambda = 2$, $\delta = 8$, $f = 16$, $\theta = 3°$			Complete two-phase raycasting		Two-phase incremental translation		Two-phase incremental rotation	
Data type	D	W, H	Samples (millions)	Time (sec.)	Samples (millions)	Time (sec.)	Samples (millions)	Time (sec.)
32-bit RGBL	160	160	1.8	2.47	0.12	0.26	0.097	0.27
		320	6.3	9.13	0.47	1.09	0.39	1.01
		480	25.2	19.98	1.88	2.15	1.56	2.17
8-bit gray	160	160	1.8	1.48	0.12	0.16	0.097	0.16
		320	6.3	5.92	0.47	0.63	0.39	0.61
		480	25.2	13.31	1.88	1.26	1.56	1.24

Figure 7: Tables of times and comparisons

5.2 Using Space-Leaping and Frame to Frame Coherence [Gudmundsson, Randen '90] [Yagel, Shi '93]

5.2.1 Fast RayCasting of volume data [Gudmundsson, Randen '90]

Direct volume rendering (visualize the volume data without any prior encoding like polygon extraction) but just the first hit. Surface based rendering.
If a surface point is visible in one projection then it is highly probable that it will remain visible in the next projection.
They can be computed incrementally avoiding full-fledged raycasting.

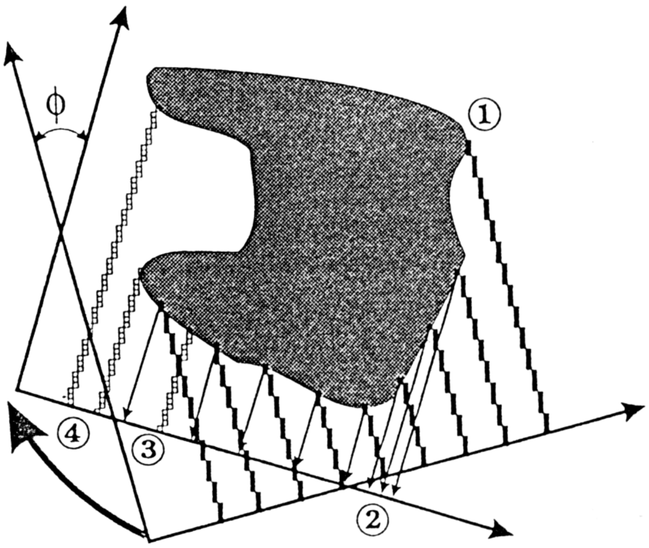


Figure 8: 2D illustration of an incremental rotation after a full raycasting. At**1** has become hidden. At **2** samples have become clustered. At **3** new rays are cast to fill in gaps. A left scan is done at **4** to find new visible surface.

Assume for simplicity:

- The rotation axis is perpendicular to the viewing direction (z axis).
- The rotation axis is parallel to y axes of the volume. Plane xz is projected to a line with constant y in the image plane.
- parallel projection.
- object rotate counterclockwise (Image plane fixed)

First full-fledged raycasting (ray per pixel).
Save in a list all detected surface points. One list for each y sorted in x from left to right.

1. Transform points in the list using a 2D geometric transformation.
2. Remove those points that have become hidden.
3. Detect new surface points by selective raycasting and insert in the list.
4. Resample all surface points.

The surface points became visible and invisible following two rules:

1. Surface points that disappear will project to the left of some new right silhouette edge.
2. Newly visible surface points will appear to the left of some old silhouette.

They don't keep track of the silhouette points.

Step 2 follow rule 1. Points are "sur-passed".
Step 3 and 4:

- After the initialization step the list contains uniformly sampled but the rotation transformation yields to non-uniform *(See figure 8).*
- So they resample rounded to the nearest pixel if more than one per pixel select the nearest. Samples slides too far apart new rays must be cast.
- step 3 and rule 2:
  All the pixels are initialized to B background. The criterion to cast rays in step 3 is : *A ray is cast for all contiguous pixels containing B immediately to the left of a resampled and shaded pixel.* Stop the scan when a ray does not detect a surface.

This fill the gaps and detecting newly visible points.
PROBLEM: When angle increment is large, background become visible between disjoint objects. The left scan will stop and some surface will go undetected.
SOLUTION: An additional right scan.

5.2.2 Accelerating Volume rendering by Space-Leaping [Yagel, Shi '93]

Similar approach that in [Gudmundsson, Randen '90]. There is a great deal of coherence between any pair of consecutive images (few rendering parameters change).
They save the coordinates of the first non-transparent voxel encountered by the ray emitted at each pixel. The save it in the "C-Buffer" *(See figure 9).*
The change in position, color, or intensity of a light source is accelerated directly for the data set. Rotation of the world space implies the C-Buffer$[i, j]$ coordinates are transform to C'-buffer$[i', j']$.

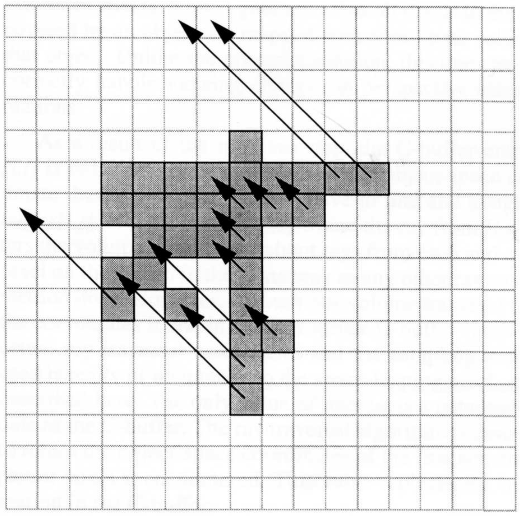


Figure 9: 2D illustration of the passage of rays through the grid when employing space-leaping

Then we have two cases:

1. Empty spaces just trace a ray again.
2. More than one sample per pixel.
    - Z-buffer doesn't work. This approach fails due to discretization artifacts. Some different objects can be mapped to the same coordinates.
    - Solution to the previous they use the approach of [Gudmundsson, Randen '90].
    - Solution to the previous splatting approach.
    - The rays that hit no object have background and no value in the C-Buffer. Bounding Boxes or their solution the back-faces are objects (They must be removed when they become a front face).

Difference between [Gudmundsson, Randen '90] and [Yagel, Shi '93]:

- [Gudmundsson, Randen '90] have image degradation can only support first opaque ray traversal.
- [Yagel, Shi '93] no image degradation and independent of the image composition.
- [Gudmundsson, Randen '90] just rotation.
- [Yagel, Shi '93] enables rotation and change in the shading parameters.

6 Inter Ray Coherency and Shear-Wrap

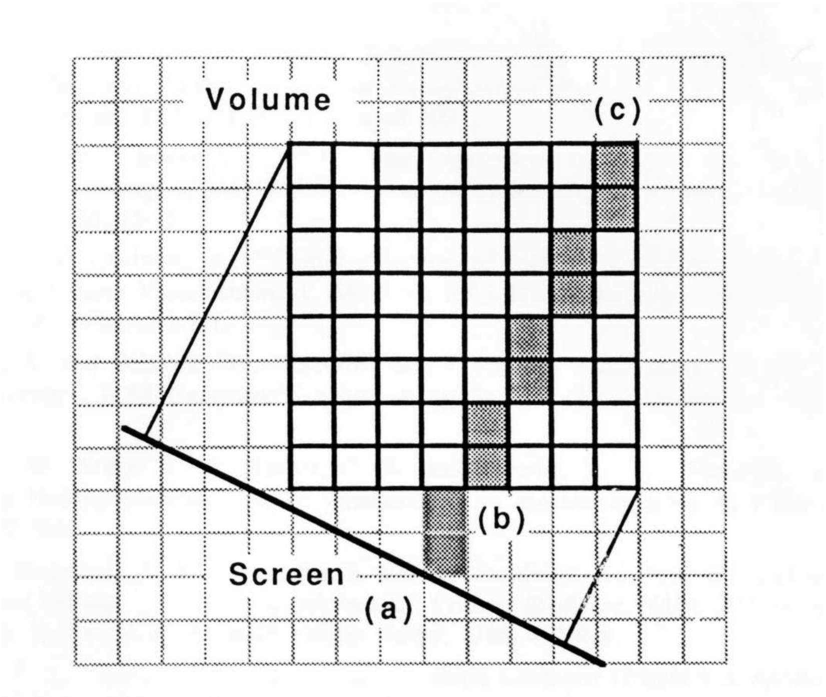


Figure 10: Principles of volumetric ray casting

6.1 Template-Based Volume Viewing [Yagel, Kaufman '92]

Exploiting coherence between rays in **parallel projection**.

- They use discrete Ray tracing. discrete line (constant interpolation) or continuos line (multiple samples per voxel and higher order interpolation).
  They use the first approach valid when you have binary data (synthetic objects voxelization with the second approach will give wrong results).
- Simple algorithm *(See figure 10).*
- 3D line-voxelization algorithm 6-line or 26-line.
- Desired line characteristics.
  We want the same form for all the rays independent of the length.(The discretization error is distributed along the line).
  Problem with round errors. Entry point to the volume calculated in floating point but the discrete line algorithm is activated from the initial integer point of the grid on.
- Template based approach solve the previous problems.
  If all the rays have the same form there is not need to reactivate the line algorithm for every ray. We use ray templates computed just once.

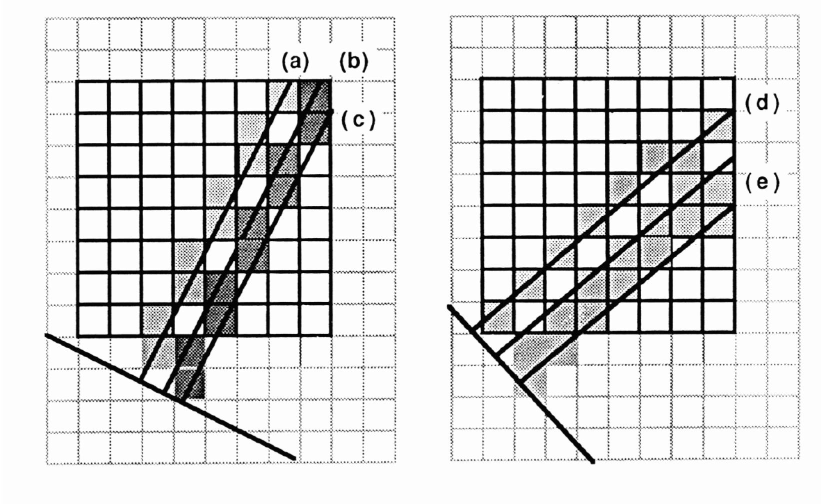


Figure 11: View dependent artifacts : **first image** some voxels between ray **a** and **b** are skipped while others are being visited by two of the rays **b c**. A whole ray of voxels is skipped between rays **d** and **c**.

PROBLEMS:
The rays differ in the exact initial position. The naive way can skip some voxels for 26-rays and multiple traversal for 6-rays *(See figure 11).*
SOLUTION:
Plane parallel to one of the volume faces (base-plane for the template placement). Guarantees a complete and uniform tessellation of the volume.

26-rays:

- Fewer voxels. 6-line (plus) twice as many voxels as a 26-line (max).
- 26-rays uniformly sampling the space.

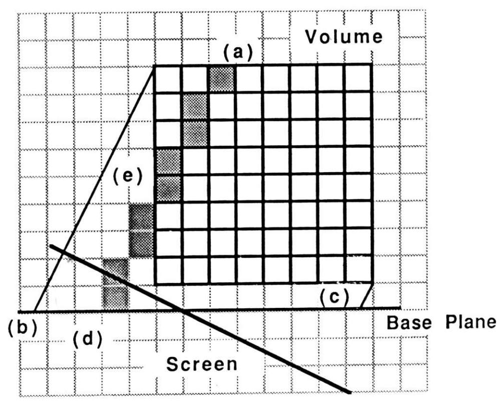


Figure 12: Principles of Template-based volume viewing

- Efficient parallel projection. *(See figure 12).*

    1. Initialization Determine the base plane.
       Computes the extend of the projected image. How big the projection plane must be.
       Computes parallel ray template (saving increments).
    2. Ray Casting (Trace the rays).
    3. 2D Mapping (From the base plane to the screen plane).

  In average 2 times faster than using a 26-line algorithm grows with the number of rays traced.

6.2 Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transform [Lacroute, Levoy '94]

Shear-warp optimized using object and image coherence for parallel projection. *(See figure 14).*

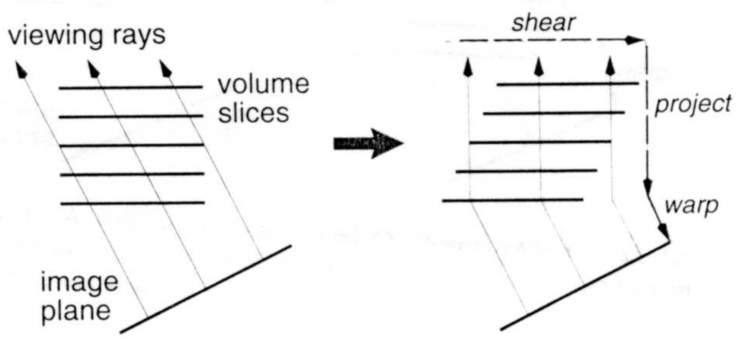


Figure 13: Parallel projection in sheared object space.

- Parallel projection of rendering algorithm. Scanline-based coherence.

- Make use of the property that sheared volume lines are aligned with pixels scanlines in the intermediate image. The volume and the image data both can be traversed in scanline order. *(See figure 14).*

- Scan-line based coherence data **run-length**: skipping runs of transparent voxels. Computed in a preprocessing step it is view independent.
  Two types of runs depending of a user defined threshold:

    1. transparent
    2. non-transparent voxels.

- To take advantage of coherence in the intermediate image, store scanlines for each opaque pixel (exceeds a user defined threshold) and offset to the next non-opaque pixel in the same scan-line. *(See figure 15).*
  Pixel array and offsets of the intermediate image form a run-length encode structure computed on the fly.

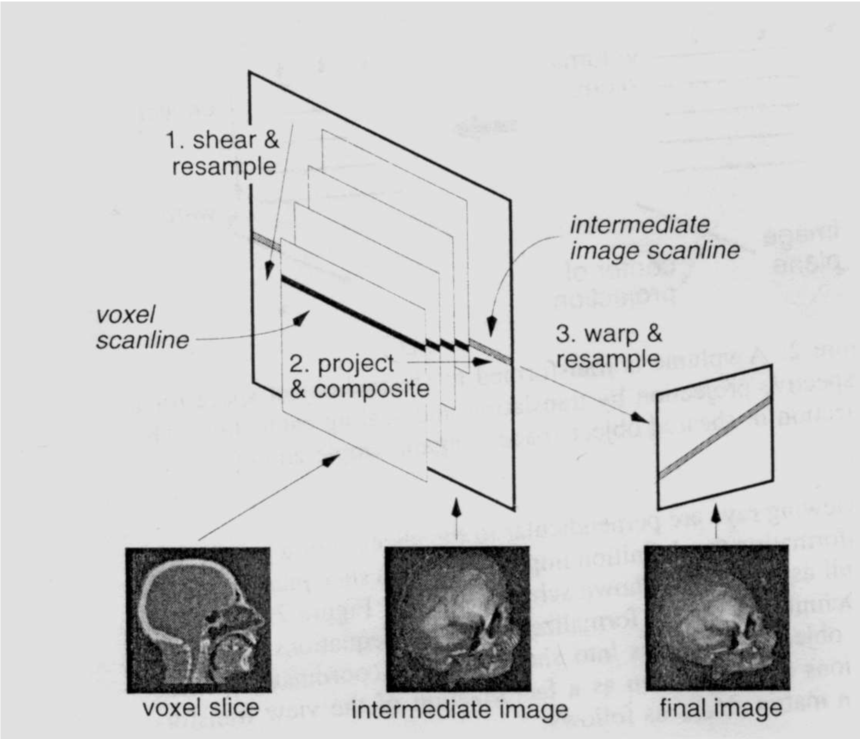


Figure 14: Scanline of pixels in the intermediate image are parallel to scanlines of voxels in the volume data. In parallel projection every voxel has the same scale factor and this can be choose so that for a given voxel scanline there is a one to one mapping between voxel and intermediate image pixel

- Offset in image pixels used to skip pixels which are occluded without examining every pixel.
- The previous structures leads to a fast scanline based rendering algorithm *(See figure 16).* Marching simultaneously through the volume and through the image in scanline order.
- The only voxels that are treated are this ones which are non transparent and visible.
- For voxels runs that are not skipped it is performed shading, resampling and composition.
  Since all the voxels in a voxel slice are scaled by the same factor. The resampling weights are the same for every voxel in a slice (bilinear interpolation).
  For shading look-up table based system.
  Look-up table to correct voxels opacity.

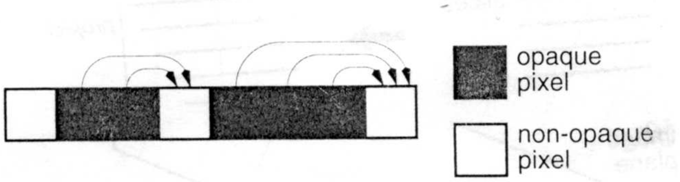


Figure 15: Offsets with opaque pixels in the intermediate image allow occluded voxels to be skipped efficiently

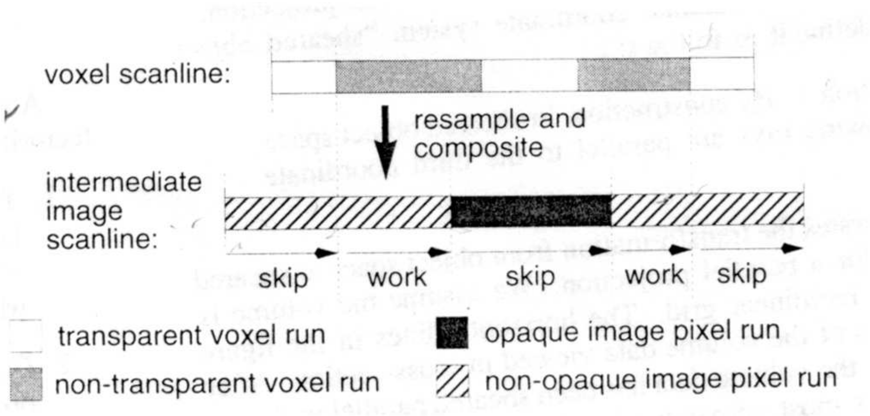


Figure 16: Resampling and composing by steaming intermediate image in scanline order, skipping over voxels which are transparent and pixels which are opaque

- Some properties.
  The opaque pixels runs usually are large so that many pixels can be skipped at once.

- Finally the image must be warp.
- Three run-length encoding of the volume are computed in a pre-processing step one for each possible principal viewing.

6.3 Fast Stereo Volume Rendering [He, Kaufman '96b] *(2nd Part)*

Segment composition scheme for stereo shear warp.

- Use the Shear-Warp Approach (Image quality decided by volume resolution)

- The left-rays (α_1, β_1) *(See figure 17)* and right-rays $(\alpha_2, \beta_2)$are perpendicular to the same volume slices in the sheared object space.
- Shear factor for the nth slice.

$$\begin{aligned} \vec{V_{left}} &= -n * \delta z \left(\frac{\cos\alpha_1 * \cos\beta_1}{\sin\beta_1}, \frac{\sin\alpha_1 * \cos\beta_1}{\sin\beta_1}, 1 \right) \\ \vec{V_{right}} &= -n * \delta z \left(\frac{\cos\alpha_2 * \cos\beta_2}{\sin\beta_2}, \frac{\sin\alpha_2 * \cos\beta_2}{\sin\beta_2}, 1 \right) \end{aligned} \tag{4}$$

- The distance between left and right eye ϕ.

$$\frac{\vec{V_{left}} \cdot \vec{V_{right}}}{\mid \vec{V_{left}} \mid\mid \vec{V_{right}} \mid} = \cos\phi \tag{5}$$

- Then

$$\begin{aligned} \cos\beta_1 \cos\beta_2 \cos(\alpha_1 - \alpha_2) + \sin\beta_1 \sin\beta_2 = \cos\phi \\ \alpha_1 - \alpha_2 \leq \phi \\ \beta_1 - \beta_2 < \phi \end{aligned} \tag{6}$$

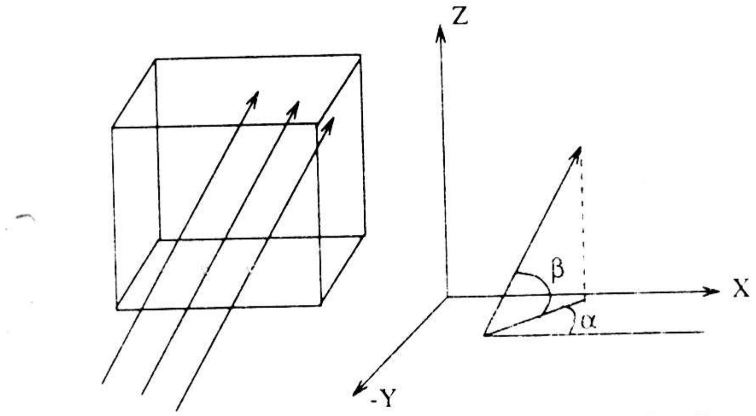


Figure 17: Relation between viewing rays and volume coordinate system

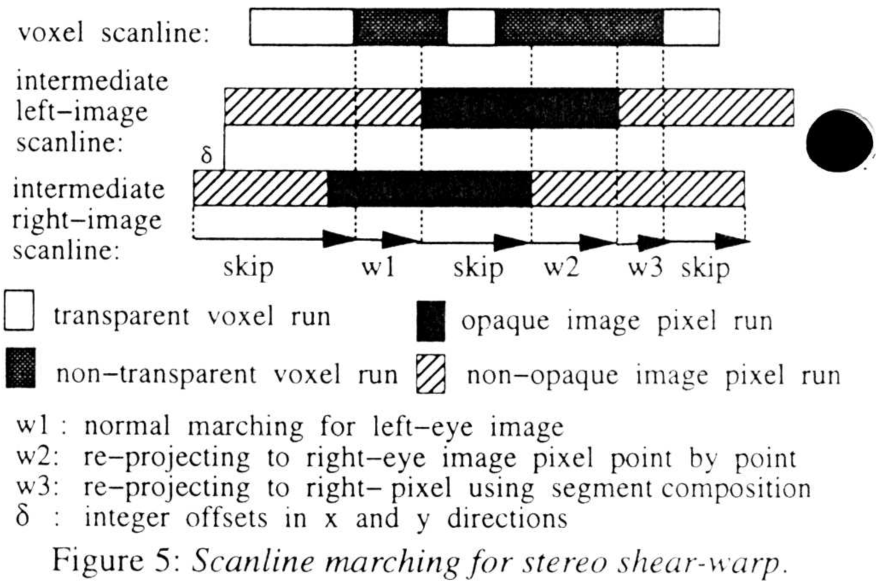


Figure 18: Scaling marching for stereo shear-warp

The offset between left and right eye projection is the same for all sampling points on the same slice.

$$\begin{aligned} C_x &= \frac{\cos\alpha_1 \cos\beta_1}{\sin\beta_1} - \frac{\cos\alpha_2 \cos\beta_2}{\sin\beta_2} \\ C_y &= \frac{\sin\alpha_1 \cos\beta_1}{\sin\beta_1} - \frac{\sin\alpha_2 \cos\beta_2}{\sin\beta_2} \end{aligned} \tag{7}$$

After some calculations *(See [He, Kaufman '96b])*. Assuming distance between eyes is 5 degrees (three times bigger than the recommended). They get to the conclusion that:

$$C_x \leq 0.1917 \text{ and } C_y \leq 0.1917 \tag{8}$$

This states that on the average the left eye-result of at least five slices can be re-projected to the right-image when the sampling rate is the same that the volume sampling distance.
So apply Shear-warp to the left image and re-project the composition result of several left eye slices to the right image.
To assure correct early ray termination. Intermediate image scan-line are marched simultaneous with the voxel scanline. If in both scanlines pixel is opaque it is skipped *(See figure 18)* .

Balanced images can be generated placing the point in the middle eye.

References

Stephen Adelson, Charles Hansen (1994). Fast Stereoscopic Images with Ray-Traced Volume Rendering. In **Arie Kaufman, Wolfgang Krueger** (Hrsg.) (1994), *1994 Symposium on Volume Visualization*, S. 3–10. ACM SIGGRAPH, Oktober 1994. ISBN 0-89791-741-3.

R. Avila, T. He, L. Hong, A. Kaufman, H. Pfister, C. Silva, L. Sobierajski, S. Wang (1994). VolVis: A Diversified Volume Visualization System. In **R. Daniel Bergeron, Arie E. Kaufman** (Hrsg.) (1994), *Proceedings of the Conference on Visualization*, S. 31–39, Los Alamitos, CA, USA, Oktober 1994. IEEE Computer Society Press.

L Bergman, H. Fuchs, E. Grant, S. Spach (1986). Image Rendering by Adaptive Refinement. *Computer Graphics (SIGGRAPH '86 Proceedings)*, 20(4):29–37.

Martin L. Brady, Kenneth Jung, HT Nguyen, Thinh Nguyen (1997). Two-Phase Perspective Ray Casting for Interactive Volume Navigation. In **Roni Yagel, Hans Hagen** (Hrsg.) (1997), *IEEE Visualization 9́7*, S. 183–190. IEEE, November 1997.

Martin L. Brady, Kenneth K. Jung, H. T. Nguyen, Thinh PQ Nguyen (1998). Interactive Volume Navigation. *IEEE Transactions on Visualization and Computer Graphics*, 4(3). ISSN 1077-2626.

Michael F. Cohen, James Painter, Mihir Mehta, Kwan-Liu Ma (1992). Volume Seedlings. In **Marc Levoy, Edwin E. Catmull** (Hrsg.) (1992), *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, S. 139–146, Cambridge, MA, mar – apr 1992. ACM Press.

D. Cohen-Or, S. Fleishman (1995). An Incremental Alignment Algorithm for Parallel Volume Rendering. *Computer Graphics Forum*, 14(3):C/123–C/133.

John Danskin, Pat Hanrahan (1992). Fast Algorithms for Volume Ray Tracing. In *Proceedings of the Workshop on Volume Visualization*, S. 91–98, New York, Oktober19–20 1992. Acm Press.

Eduard Groeller, Werner Purgathofer (1995). Coherence in Computer Graphics. Technischer Bericht TR-186-2-95-04, Institute of Computer Graphics 186-2 Technical University of Vienna, mar 1995.

Bjoern Gudmundsson, Michael Randen (1990). Incremental generation of projections of CT-volumes. In *Proceedings in The First Conference on Visualization in Biomedical Computing*, Atlanta, 1990.

Taosong He, Arie Kaufman (1996a). Fast Stereo Volume Rendering. In *IEEE Visualization '96*. IEEE, Oktober 1996. ISBN 0-89791-864-9.

Taosong He, Arie Kaufman (1996b). Fast Stereo Volume Rendering. In *IEEE Visualization '96*. IEEE, Oktober 1996. ISBN 0-89791-864-9.

P. Lacroute, M. Levoy (1994). Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation. *Computer Graphics*, 28(Annual Conference Series):451–458.

David Laur, Pat Hanrahan (1991). Hierarchical splatting: A progressive refinement algorithm for volume rendering. *Computer Graphics*, 25(4):285–288.

Marc Levoy (1990a). Efficient Ray Tracing of Volume Data. *ACM Transactions on Graphics*, 9(3):245–261.

Marc Levoy (1990b). Volume Rendering by Adaptive Refinement. *The Visual Computer*, 6(1):2–7.

Klaus Mueller, Roni Yagel (1996). Fast Perspective Volume Rendering with Splatting by Utilizing a Ray-Driven Approach. In *IEEE Visualization '96*. IEEE, Oktober 1996. ISBN 0-89791-864-9.

Ricardo Sánchez, Marcelo Carvajal (1998). Wavelet Based Adaptive Interpolation For Volume Rendering. In *IEEE Symposium on Volume Visualization*, S. 127–134. IEEE, ACM SIGGRAPH, 1998.

R. Yagel, A. Kaufman (1992). Template-based volume viewing. *Computer Graphics Forum*, 11(3):C153–C167.

R. Yagel, Z. Shi (1993). Accelerating Volume Animation by Space-Leaping. In **Gregory M. Nielson, Dan Bergeron** (Hrsg.) (1993), *Proceedings of the Visualization '93 Conference*, S. 62–69, San Jose, CA, Oktober 1993. IEEE Computer Society Press.

R. Yagel (1996). Towards Real Time Volume Rendering. In *Proceedings of GRAPHICON'96*, S. 230–241, Juli 1996.