A real-time shadow approach for an Augmented Reality application using shadow volumes

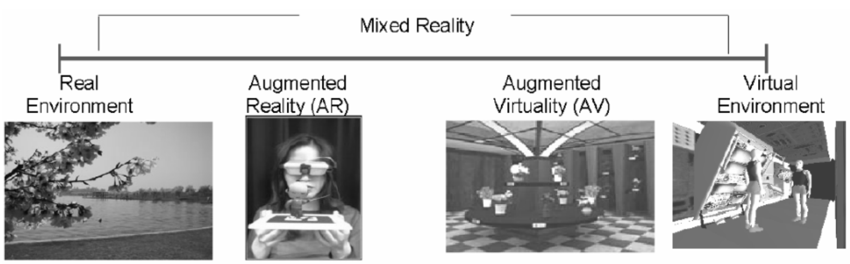
Overview

- Introduction of Concepts
- Standard Stenciled Shadow Volumes Method
- Proposed Approach in AR Application
- Experimental Results and Examples
- Evaluation and Improvement Analysis
- References

Augmented Reality

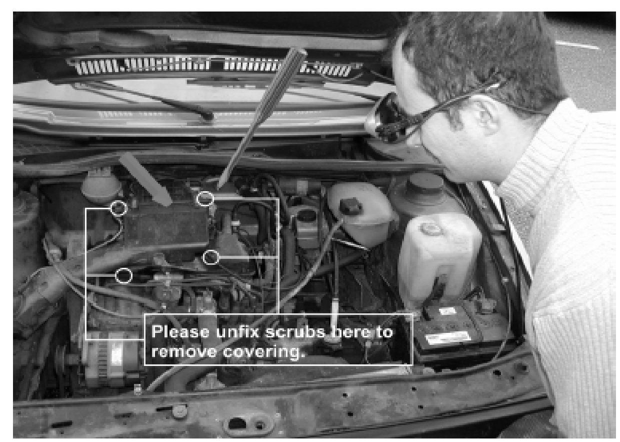
AR is defined by three desired characteristics:

1) Combines real and virtual
2) Interactive in real time
3) Registered in 3-D

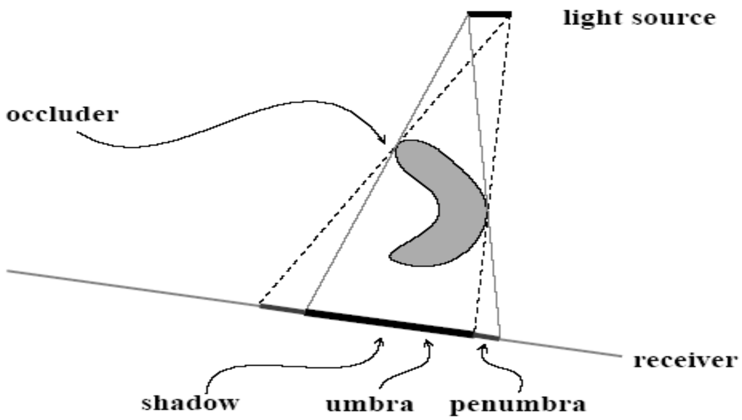


Reality - Virtuality (RV) Continuum [Milgram et al.

Augmented Reality



What's Shadow?



Shadow Volume

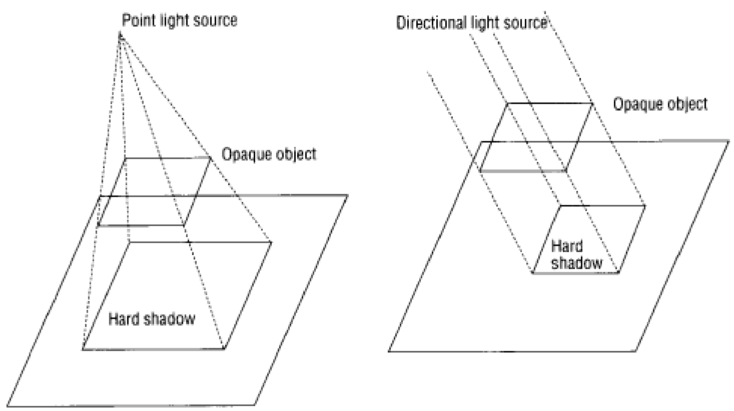
Shadow Categories

- Hard Shadow (Opaque *occluder* & point light)
- Soft Shadow (Opaque occluder & area light)
- Complex Shadow (Transparent occluder)

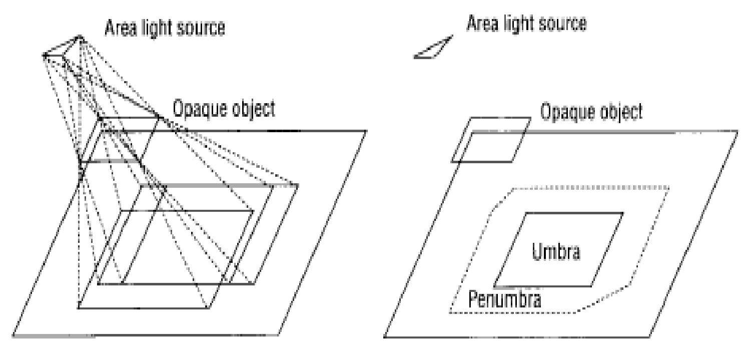
Shadow Volume

It defines a region of space that is in the shadow of a particular ***occluder*** given a particular ideal ***light source***.

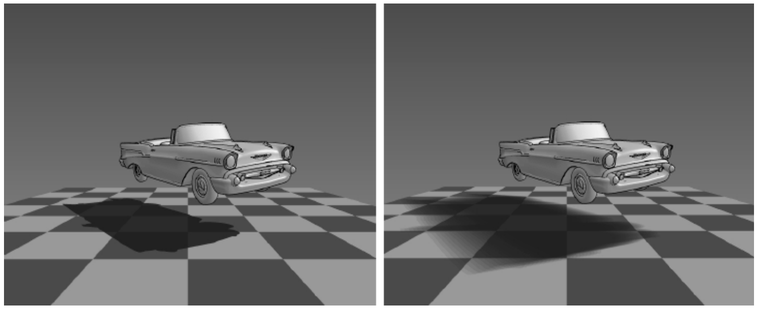
Hard Shadow



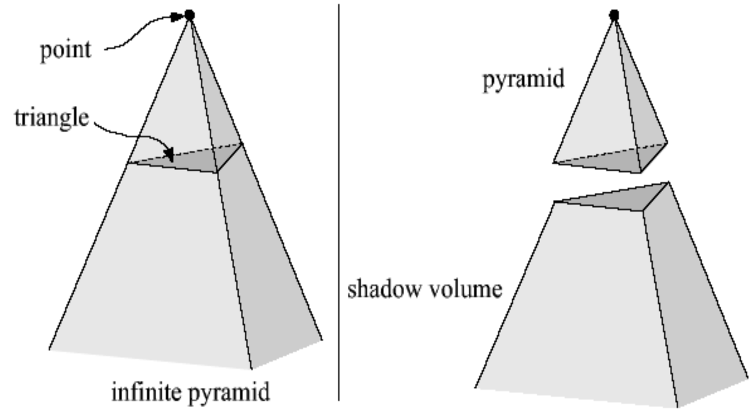
Soft Shadow



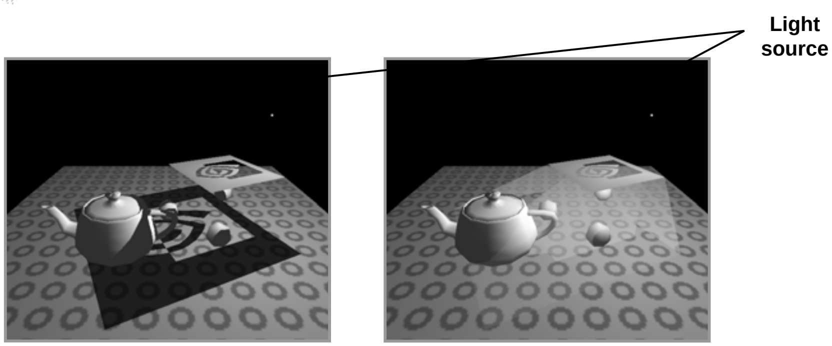
Hard Shadow vs. Soft Shadow



Shadow Volume



Shadow Volume Visualization



Scene with shadows **Visualization of the shadow volume**

OpenGL

OpenGL - 3D supported graphic software and provide
geometric *primitives* - points, lines, polygons, images, and bitmaps.
commands - allow the specification of geometric objects in two or three dimensions; control how objects are rendered into the *frame buffer*.

Buffer Mechanism

Z Buffer & *Stencil Buffer* & *Frame Buffer*

All buffers that store attributes for each pixel can be called as *Frame Buffer*, which can be divided into 4 categories below,

-- *Color Buffer*, caches color indexes or RGBA value.

-- *Depth Buffer (Z-Buffer),* caches depth information for each pixel.

-- *Stencil Buffer*, caches some portions of the whole image or scene for re-rendering quickly.

-- *Accumulation Buffer*, used to synthesize image.

ZPass & ZFail Test

The OpenGL depth test function is used to compare each incoming pixel z value with the z value present in the ***depth buffer***. The comparison is performed only if depth testing is enabled.

If the test procedure by the OpenGL function passes, we call this test ***ZPass***, otherwise ***ZFail***. We can image these two possibilities as the same in adverse test orientations:

-- one tests along the Z direction;

-- the other along the Z negative direction.

Homegeneous Coordinates

Suppose we have a point (x, y) in the Euclidean plane. To represent this point in the projective plane, we simply add a third coordinate of 1 at the end: (x, y,1).

Overall scaling is unimportant, so the point (x,y,1) is the same as the point , for any nonzero.

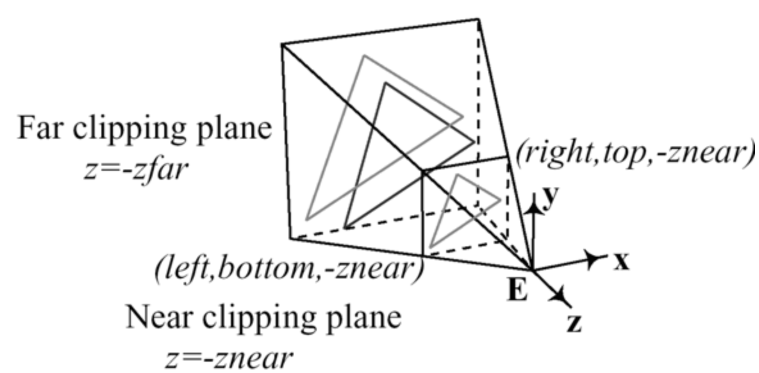
$$(X, Y, W) = (\alpha X, \alpha Y, \alpha W)$$

Because scaling is unimportant, the coordinates (X,Y,W) are called the homogeneous coordinates of the point.

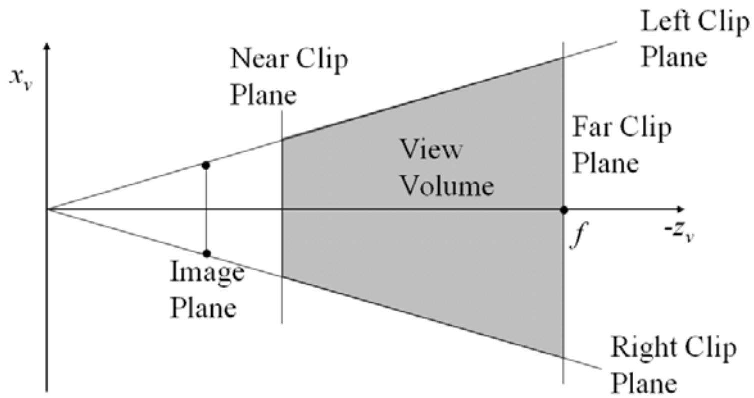
To denote a infinite point, let w equals 0 such as (x, y, 0).

Clip Planes

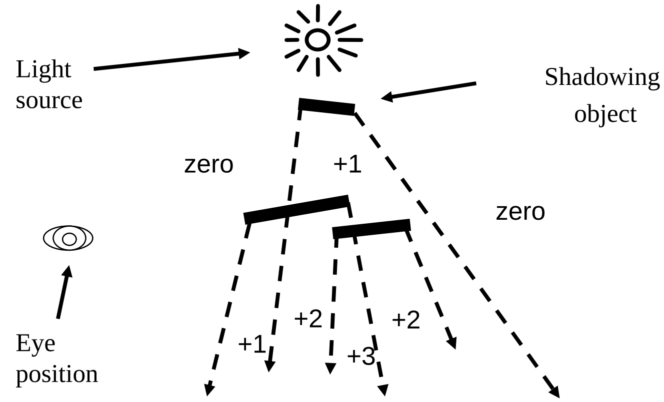
Clip Planes: Geometry is always clipped against the boundaries of a six-plane frustum in x, y, and z.



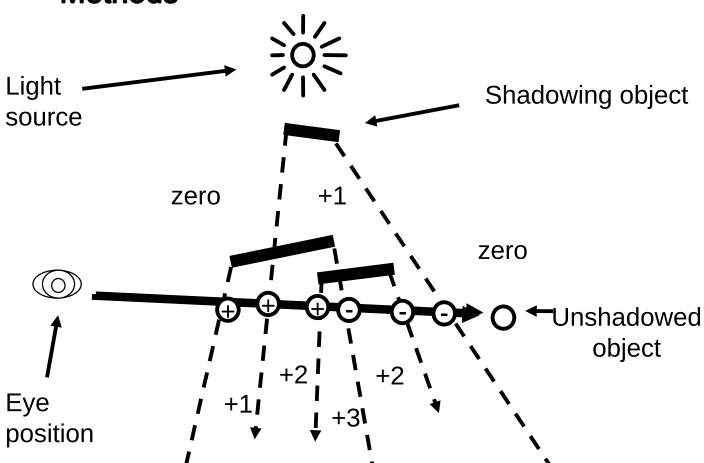
Clip Planes



Standard Stenciled Shadow Volumes Methods

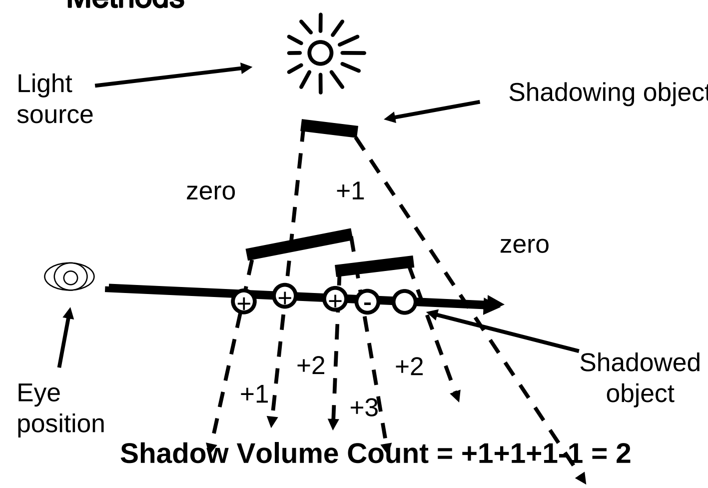


Standard Stenciled Shadow Volumes Methods



Shadow Volume Count = +1+1+1-1-1-1 = 0

Standard Stenciled Shadow Volumes Methods



Shadow Volume Count = +1+1+1-1 = 2

Standard Stenciled Shadow Volumes Methods

Generation of shadow volumes

Initialize the depth buffer and color buffer
Then twice using face culling and don't update depth or color

1st pass:

render *<u>front</u>* faces and *<u>increment</u>* when depth passes, viz. ZPass
(or *decrement* when depth fails, viz. ZFail)

2nd pass:

render *<u>back</u>* faces and *<u>decrement</u>* when depth passes, viz. ZPass
(or *increment* when depth fails, viz. ZFail)

Standard Stenciled Shadow Volumes Methods

Too simple? Maybe there are other unconsidered problems practically, so before we starting analysis it we can show you some photos from practical applications using OpenGL.

Standard Stenciled Shadow Volumes Methods

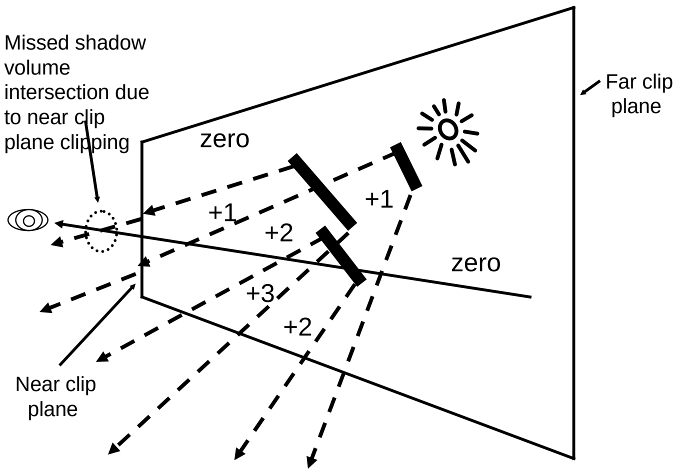
Correct shadows: Incorrect shadows:

Standard Stenciled Shadow Volumes Methods

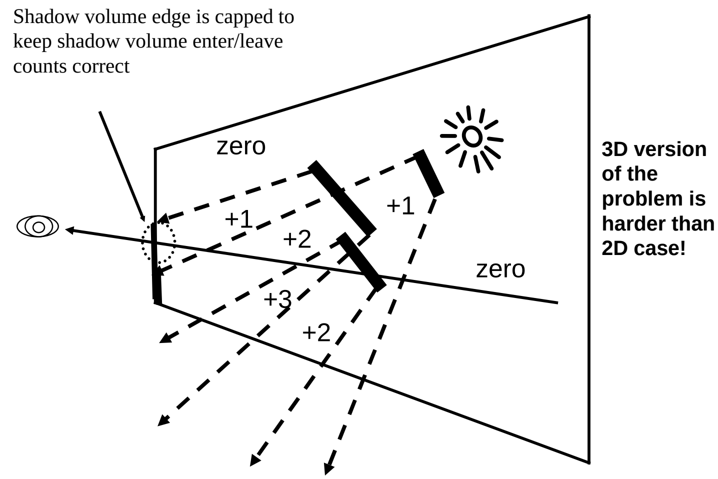
Why does it occur aliasing when we present the closer shadow effect?
Do you remember the <u>clip planes</u> talked just now?

Remark: so we can determine that the intrinsic cause leads to aliasing is <u>clip planes</u>. Since a real OpenGL space is NOT an ideal 3D model corresponding real world. so when we generate our shadow volumes, we can NOT assure that those shadow volumes are ***CLOSED*** *polygons. Furthermore, this error will lead error count for shadow volumes number of pixel.*

Problems with Near Plane Clipping (*zpass approach*)



Capping Shadow Volumes at the Near Clip Plane



Standard Stenciled Shadow Volumes Methods

We illustrate the problems caused by *near clipping plane* when we using *zpass* method; Similarly we will face to *far clipping plane* problems when we in *zfail* way.

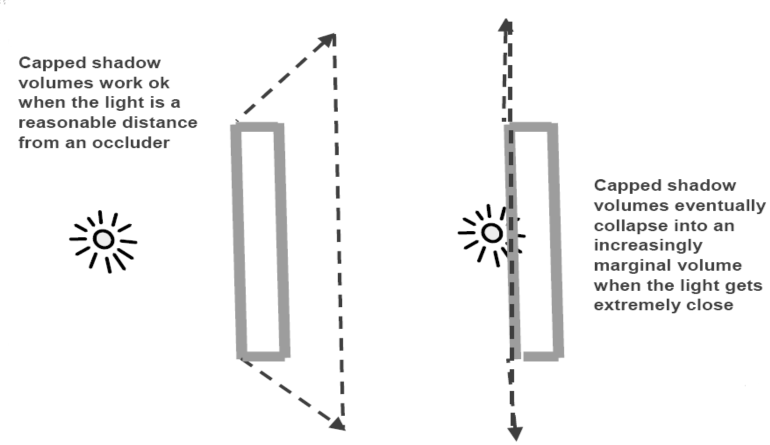
Z-Pass Algorithm

We count the number of shadow volumes for some pixel from the eye point to the first resterized visible pixel. So aliasing is due to *near plane clipping*.

Z-Fail Algorithm

We count the number of shadow volumes for some pixel from the infinite to the pixel. So aliasing is due to *far plane clipping*.

Problems Caused by Near Clip Plane



ZPass Analysis

According to the diagram, we can conclude that even when done carefully, shadow volume *near plane capping* is treacherous because of the fragile nature of required ray-plane intersections and the inability to guarantee identical and bit-exact computations .

So can we avoid the *near plane clipping*? Sure! We can use the *zfail* method.

We must mention that shifts the problem to the far plane clipping shadow volumes.

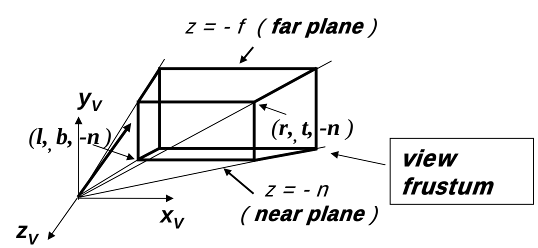
Alternative Approach: Zfail

- Render scene to initialize depth buffer
  - Depth values indicate the closest visible fragments
- Use a stencil enter/leave counting approach
  - Draw shadow volume twice using face culling
    - 1st pass: render ***back*** faces and ***increment*** when depth test **fails**
    - 2nd pass: render ***front*** faces and ***decrement*** when depth test **fails**
  - Don't update depth or color
- Then, pixel's stencil is non-zero if pixel in shadow, and zero if illuminated

Zfail versus Zpass Comparison

- When stencil increment/decrements occur
  - *Zpass:* on depth test pass
  - *Zfail:* on depth test fail
- Increment on
  - *Zpass:* front faces
  - *Zfail:* back faces
- Decrement on
  - *Zpass:* back faces
  - *Zfail:* front faces

Perspective Projection



Projective Matrix

$$\mathbf{P} = \begin{bmatrix} \frac{2\times Near}{Right-Left} & 0 & \frac{Right+Left}{Right-Left} & 0 \\ 0 & \frac{2\times Near}{Top-Bottom} & \frac{Top+Bottom}{Top-Bottom} & 0 \\ 0 & 0 & -\frac{Far+Near}{Far-Near} & -\frac{2\times Far\times Near}{Far-Near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Only third row depends on *Far* and *Near*

Standrad Stenciled Shadow Volumes Methods

P is used to transform eye-space positions to clip-space positions

$$\begin{bmatrix} x_c & y_c & z_c & w_c \end{bmatrix}^T = \mathbf{P} \begin{bmatrix} x_e & y_e & z_e & w_e \end{bmatrix}^T$$

We consider the limit of P as the far clip plane distance is driven to infinity

$$\lim_{Far\to\infty} \mathbf{P} = \mathbf{P}_{\mathbf{inf}} = \begin{bmatrix} \frac{2\times Near}{Right-Left} & 0 & \frac{Right+Left}{Right-Left} & 0 \\ 0 & \frac{2\times Near}{Top-Bottom} & \frac{Top+Bottom}{Top-Bottom} & 0 \\ 0 & 0 & -1 & -2\times Near \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Verifying Pinf will not clip infinitely far away vertices

- Transform eye-space (0,0,-D,0) to clip-space

$$\begin{bmatrix} x_c \\ y_c \\ -D \\ -D \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \begin{bmatrix} \frac{2\times Near}{Right-Left} & 0 & \frac{Right+Left}{Right-Left} & 0 \\ 0 & \frac{2\times Near}{Top-Bottom} & \frac{Top+Bottom}{Top-Bottom} & 0 \\ 0 & 0 & -1 & -2\times Near \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -D \\ 0 \end{bmatrix}$$

- Then, assuming glDepthRange(0,1), transform clip-space position to window-space position

$$z_w = 0.5 \times \frac{z_c}{w_c} + 0.5 = 0.5 \times \frac{-D}{-D} + 0.5 = 1$$

- So ∞ *in front of eye* transforms to the maximum window-space Z value, but is still within the valid depth range (i.e., not clipped)

Rendering closed, but infinite, shadow volumes

To be robust, the shadow volume geometry must be closed, even at infinity. To close a shadow volume completely, we must combine three sets of polygons:

-- all of the possible silhouette polygon edges extruded to infinity away from the light
-- all of the occluder's back-facing triangles, with respect to L, projected away from the light to infinity
-- all of the occluder's front-facing triangles with respect to L

Summary for Rendering Procedure

1. Placing the conventional far clip plane “at infinity”
2. Rasterizing infinite (but fully closed) shadow volume polygons via homogeneous coordinates
3. Adopting the ***zfail*** stencil-testing scheme

The following diagram show the implementation steps.

Standrad Stenciled Shadow Volumes Methods

	Color buffer	Z-buffer	Stencil buffer
Draw objects into color buffer and Z-buffer	O	O	
Draw shadow volumes into the stencil buffer	O	O	S
Draw shadows onto the objects	O_s	O	S

Legend:
O Object
S Shadow mask
O_s Shadowed object

Proposed Approach in AR Application

Differences between VR and AR
All objects in VR are virtual, but In AR some are real ones. So AR is a hybrid world with real and virtual objects, then we should face to the following problems that don’t occur in VR scenes.

Q1. How to metric the geometry characteristics of real objects?
Q2. Which order we should render the shadow in?

Shading and Shadows in AR

For Augmented Reality applications:

- consistency of geometry;
- consistency of time (smooth interaction);
- consistency of illumination.

A real-time estimation of the position and the direction of the ***real light source*** is required to calculate the right shading and shadows of the virtual objects.
(real-time shadows of *real/virtual* objects onto *real/virtual* objects)

Remodelling and Relighting Real Scenes

Two sets of photographs to estimate the ***geometry*** and the ***lighting model*** of the real scene:

- ✓ The first set consists of images of the scene taken from different viewpoints;
- ✓ The second set consists of images of the scene taken from one viewpoint but with *a real light source* at different positions.

The system permits interactive modification of the ***scene geometry***, including removing objects and adding, changing or removing ***light sources***.

Proposed Approach in AR Application

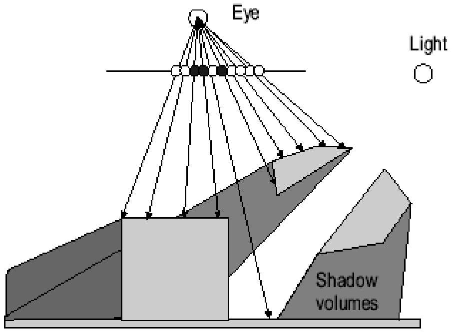
Real objects are represented by virtual objects that are invisible for the user, called <u>*phantom*</u> objects.

It is NOT necessary to have the exact *phantom* model for each real object in the world.

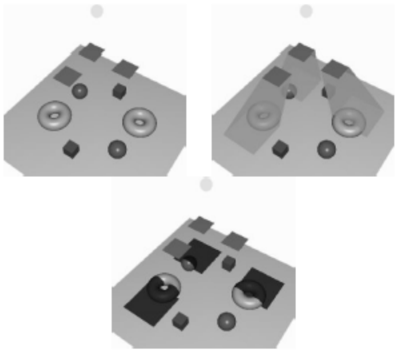
We can start to calculate the silhouette of both kinds of objects (virtual and real) and extrude the shadow volume.

The implementation of the silhouette calculation and the extrusion of the shadow volume can be easily implemented with the same method as in the *standard stenciled shadow volume* algorithm.

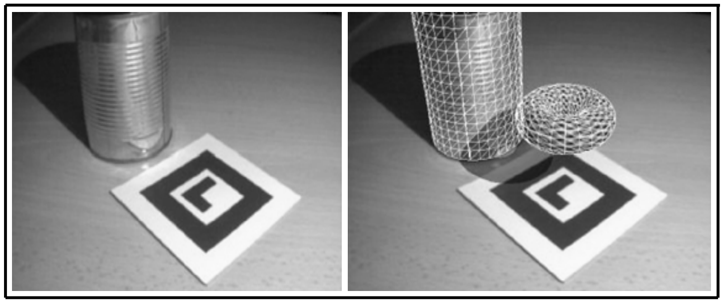
Proposed Approach in AR Application



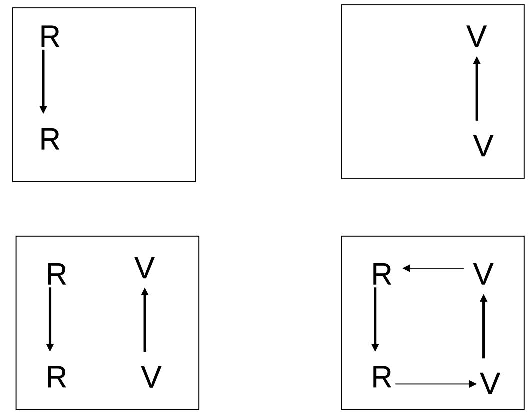
Proposed Approach in AR Application



Proposed Approach in AR Application



Four possibilities of shadow casts



Proposed Approach in AR Application

Rendering steps

As the algorithm has to work for all cases, for real/virtual objects that cast shadows onto real/virtual objects, the four different cases are rendered in the following three steps:

Step 1: *Real* objects cast their shadows onto *real* objects
Step 2: *Virtual* objects cast their shadows onto *real* objects
Step 3: *Real* and *virtual* objects cast their shadows onto *virtual* objects

Proposed Approach in AR Application

1. *R->R* works with a normal webcam.

2. *V->R* use the *phantom (virtual 3D models of real objects)* and shadow volumes of virtual objects to get this effect, using the method of *standard stenciled shadow volumes algorithm.*

Since missing the material information from the real word, we use black and transparent shadow polygons that are blended over the parts of the real scene (real image).

Proposed Approach in AR Application

	Color buffer	Z-buffer	Stencil buffer
STEP 1			
Copy framegrabber image of the webcam into the color buffer	R_R		
STEP 2			
Draw real objects into the Z-buffer	R_R	R	
Draw virtual shadow volumes into the stencil buffer	R_R	R	S_V
Draw virtual shadows onto real objects	R_{RV}	R	S_V

Proposed Approach in AR Application

3. *R->V &V->V* render the whole scenes which include *phantom (virtual 3D models of real objects)* with diffuse and specular components into the stencil buffer.

(*to generate shadow volumes for both virtual and real objects and draw the shadow volume polygons into the stencil buffer*).

Proposed Approach in AR Application

STEP 3			
Draw virtual objects into the color buffer and Z-buffer.	$R_R V$	R V	S_V
Draw the shadow volumes into the stencil buffer	$R_{RV} V$	R V	S_{RV}
Draw virtual shadows onto the virtual objects	$R_{RV} V_{RV}$	R V	S_{RV}

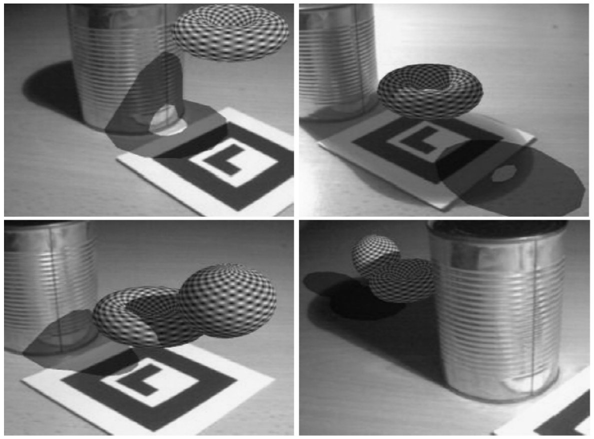
Legend:
R Real object
V Virtual object
R_R Real object with real shadow
R_{RV} Real object with real and virtual shadow
R_{RV} Virtual object with real and virtual shadow
S_V Mask of real shadow
S_{RV} Mask of real and virtual shadow

Experiments

- **The real scene is represented by virtual *phantom* models;**
- **The *virtual light* position and its properties are independent of the real light conditions;**
- **The position and orientation tracking of the real object is accomplished by using *ARToolKit* as marker detection system;**
- **No any animated objects;**

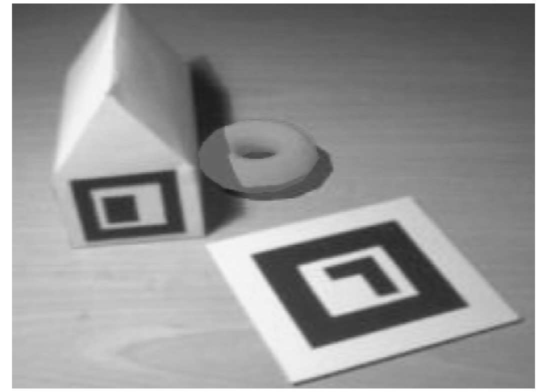
The augmented shadows enhance the real scene a great deal and offer the user a more intuitive and very photorealistic world.

Experimental Results and Examples



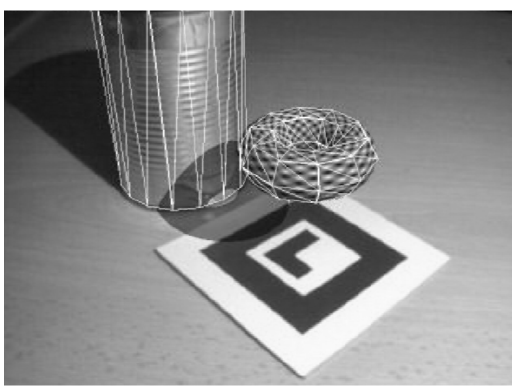
53

Experimental Results and Examples



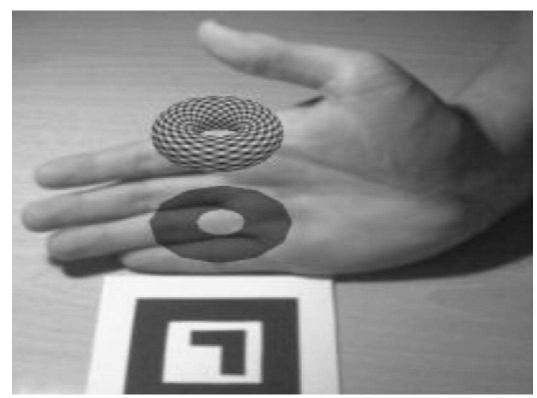
54

Experimental Results and Examples



55

Experimental Results and Examples



56

Conclusion

Proposed a *Shadow Volume* based shadow rendering algorithm for AR applications.

- A modified real-time Shadow Volume algorithm
- Multi-steps rendering procedure based on relations betweens real and virtual objects
- Volume algorithm improvement
  Volume algorithm can be improved by using Portals, BSP and some other techniques that avoid the rendering of unnecessary shadow volumes.
- Using nVIDIA's shading language Cg
- Improvement of the shadows (e.g. soft shadows)

Evaluation and Improvement Analysis

As showed in examples, the orientation of light source is virtual and specified manually. Naturally, ***can we give an estimation model for real light in real scenes?***

Then we can reconstruct the shadow of augmented reality under the real light source. Maybe very difficult, but very interesting.

reference: "what is the set of images of an object under all possible lighting conditions? "

References

[1] Michael Haller, Stephan Drab, and Werner Hartmann. "**A real-time shadow approach for an Augmented Reality application using shadow volumes**". *VRST'03*, Oct. 1-3, 2003, Osaka JAPAN.

[2] Cass Eveeritt and Mark J. Kilgard. "**Practical and robust stenciled shadow volumes for hardware-accelerated rendering**". NVIDIA Corporation, March 2002.

[3] Franklin C. Crow. "Shadow algorithms for computer graphics". *Proceedings of SIGGRAPH*, 1977, pp. 242-248.

References

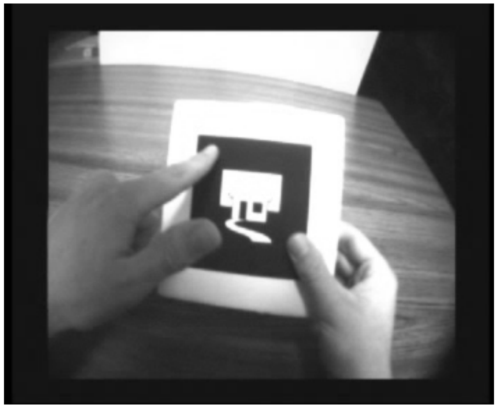
[4] Andrew Woo, Pierre Poulin, and Alain Fournier. "A survey of shadow algorithms". *IEEE Computer Graphics and Applications*, Nov. 1990, pp. 13-32.

[5] Eric Haines, Tomas Möller. "Real-time shadows", in Real-time Rendering by Tomas Möller and Eric Haines, A.K. Peters, ISBN 1568811012.

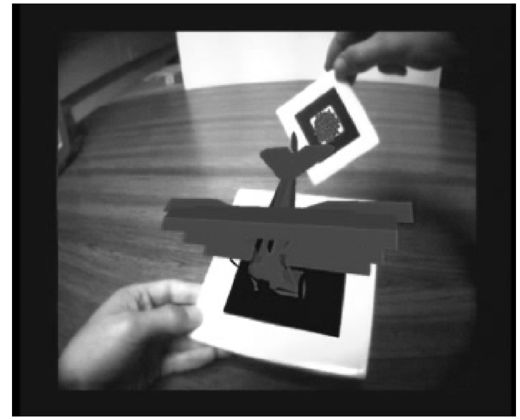
[6] A. Katayama, Y. Sakagawa, and H.Tamura. "A method of shading and shadowing in image-based rendering". *Int. Conference on Image Process*, vol. 3, pages 26-30. *IEEE*, 1998.

[7] M. D. McCool. "Shadow volume reconstruction from depth maps". *ACM Transaction on Graphics*, 2000

Augmented Reality -- AR Demo Movie 1



Augmented Reality -- AR Demo Movie 2



Q & A Session

Thank you!

Q & A

? ? ? ? ? ?