GEARS: A General and Efficient Algorithm for Rendering Shadows

Voicu Popescu

Reference

- GEARS.pdf
- GEARS.mov

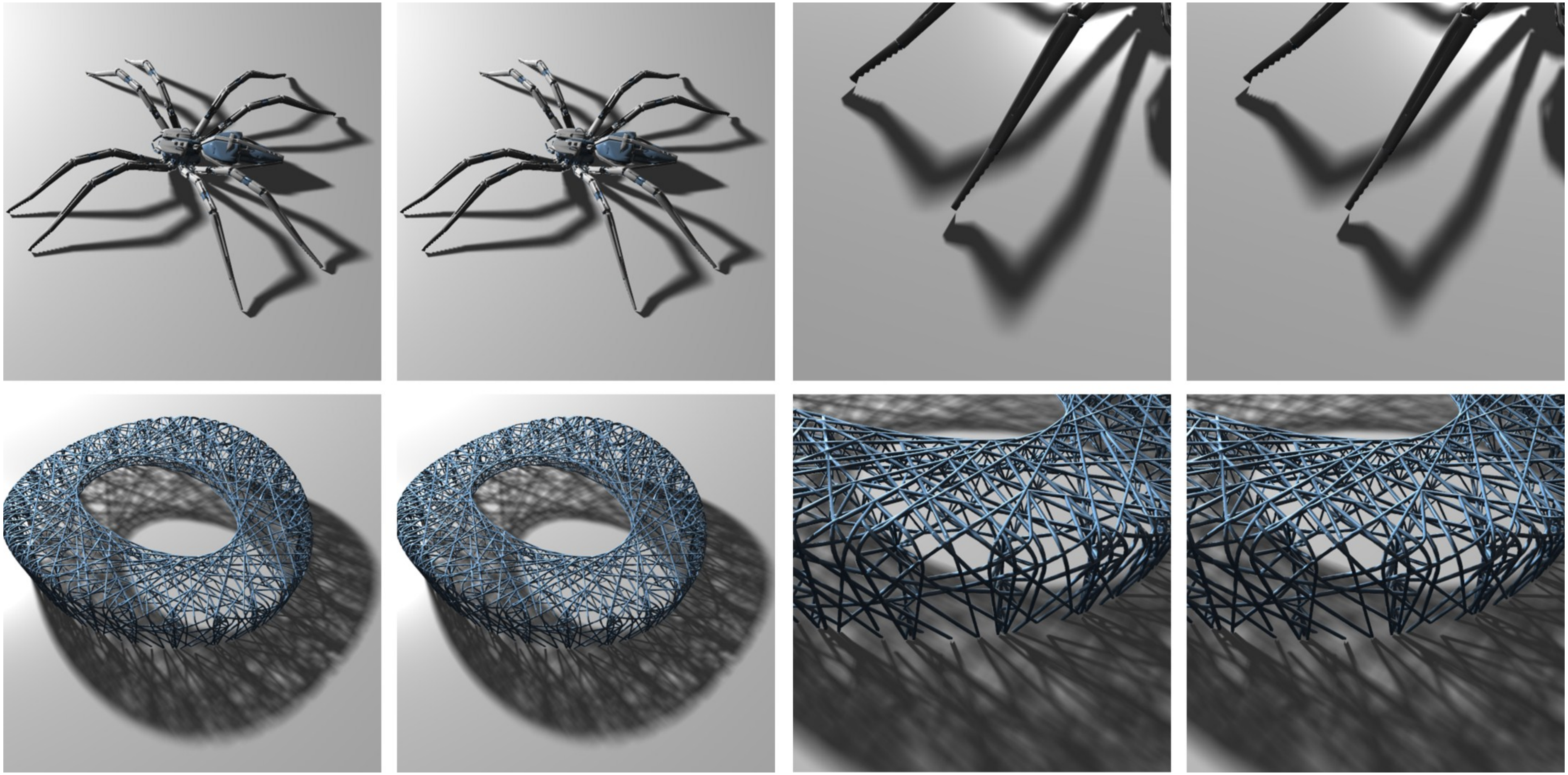


Fig. 1: Soft shadows rendered with our method (left for each pair) and with ray tracing (right for each pair), for comparison. The average frame rate for our method vs. ray tracing is 15.7fps vs. 0.5fps for the Bird Nest scene and 32.7fps vs. 2.7fps for the Spider scene.

Shadows

- Important effect in graphics
- Difficult to render
  - Require estimating visibility from light source

Hard shadows

- Point light sources
- Usually rendered using shadow mapping
  – Z-buffer rendered from light point
  – Output sample unprojected to 3-D then projected to shadow map
  – Inaccurate when shadow map texel is magnified in output image

Pixel-accurate hard shadows

- Render scene from output view, w/o shadows
- Unproject/re-project output image samples to shadow plane
- Define grid on shadow plane
- Assign output image samples to grid cells
- Render scene from light; For all triangles *t*
    - Project *t* onto grid to *t'*
        - For all grid cells *c* touched by *t'*
            - » For all output samples *s* in *c*
                - Mark *s* as in shadow if covered by *t'*

Soft shadows

- Area light source
- Umbra (full shadow), penumbra (in between), and light regions
- Computationally expensive
  - Estimating visibility from light source requires many rays

Soft shadows approaches

- Trace k x k rays for each pixel
  - Ray tracing is expensive, HW unfriendly, acceleration difficult, especially for dynamic scenes
- Construct k x k conventional shadow maps
  - Expensive to render scene k x k times
  - Approximation errors of conventional shadow maps
- Render scene at k x k resolution for each output image sample
  - Approach taken by our technique
- k should be at least 16

Algorithm overview

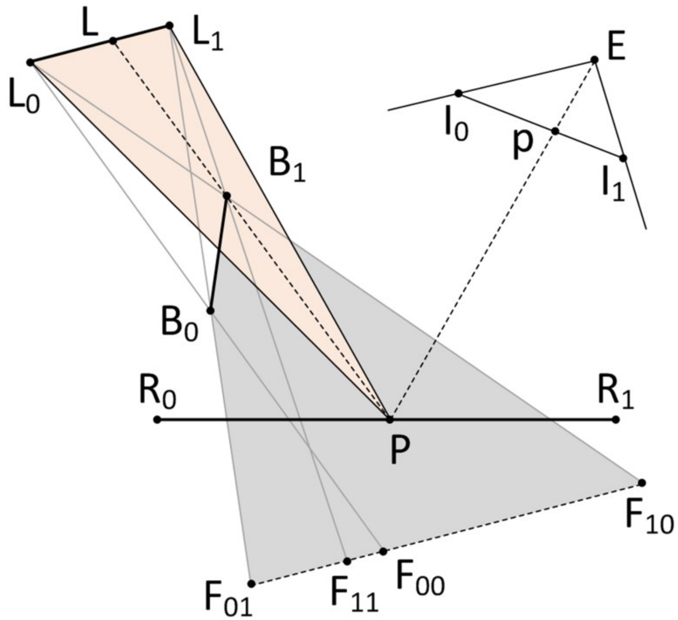


Fig. 2: Soft shadow computation overview for light L_0L_1, output image I_0I_1 with viewpoint E, blocker B_0B_1, and receiver R_0R_1.

Given a 3-D scene S modeled with triangles, an area light source modeled with a rectangle (L_0L_1 in Fig. 2), and a desired view with eye E and image plane I_0I_1, our algorithm renders soft shadows with the following approach:

1. Compute the output image without the soft shadows.
   a. Render S from E to obtain image I_0I_1.
2. Unproject each pixel p in I_0I_1 to pixel sample P.
3. Assign potentially blocking tris. to pixel samples P.
4. For each P, compute the frac. visibility v_p of L_0L_1.
   a. Construct camera PL_0L_1 with eye P and image plane L_0L_1 (orange frustum in Fig. 2).
   b. Render with PL_0L_1 all blocking triangles assigned to P on visibility bit mask M_P.
   c. Compute v_p as the percentage of unoccluded light samples in M_P. In Fig. 2, $v_p = LL_1 / L_1L_0$.
5. Add the contribution of light L_0L_1 to each pixel p of I_0I_1 using the computed fractional visibility v_p.

Acceleration scheme

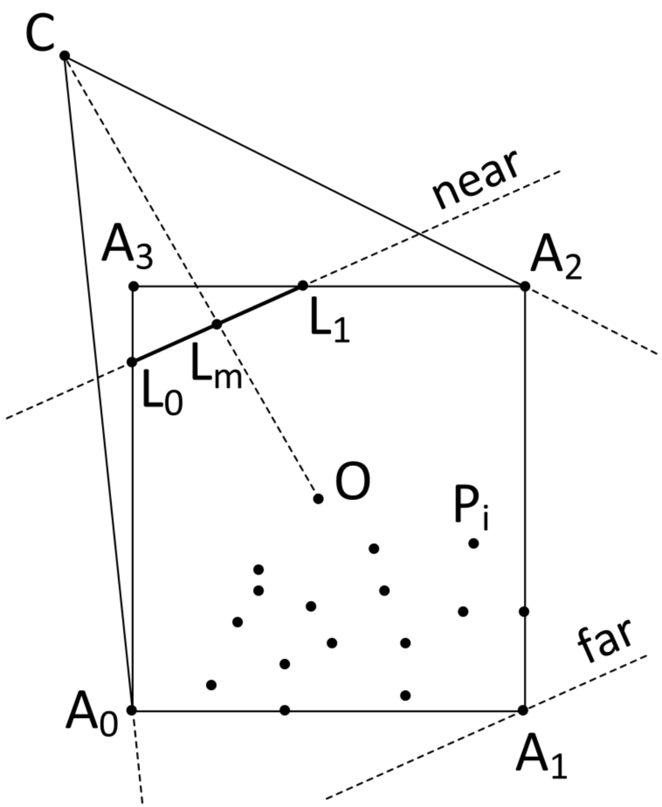


Fig. 3: Camera used to define and populate grid.

Given a scene S, a rectangular area light source $L_0L_1L_2L_3$, and the pixel samples P of the output image, the acceleration scheme computes a regular 2-D grid G that stores at each cell (u,v) a set of pixel samples P_{uv} and a set of potentially blocking triangles T_{uv} for the pixel samples in P_{uv}. The acceleration scheme proceeds as follows:

A.1. Construct camera C with grid G as image plane.

A.2. For each pixel sample P

a. Project P with C to P'.

b. $P_{uv} = P_{uv} \cup P$, where $P' \in G(u,v)$.

A.3. For each triangle T in S

a. For each vertex B_i of T and each light vertex L_j

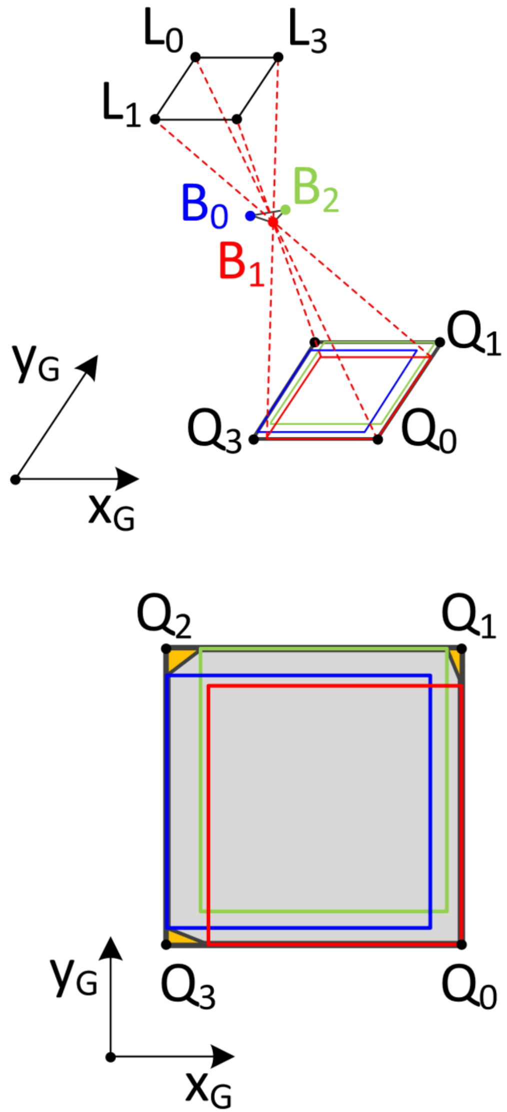
i. Compute 3-D points $F_{ij} = B_i + \| B_i - L_j \| d_{ij}$.

b. Project vertices B_i and points F_{ij} with C and compute the 2-D AABB of the projections.

c. For each $G(u,v)$ touched by the AABB

i. $T_{uv} = T_{uv} \cup T$.

Acceleration scheme



Given a scene S, a rectangular area light source $L_0L_1L_2L_3$, and the pixel samples P of the output image, the acceleration scheme computes a regular 2-D grid G that stores at each cell (u,v) a set of pixel samples P_{uv} and a set of potentially blocking triangles T_{uv} for the pixel samples in P_{uv}. The acceleration scheme proceeds as follows:

A.1. Construct camera C with grid G as image plane.

A.2. For each pixel sample P

a. Project P with C to P'.

b. $P_{uv} = P_{uv} \cup P$, where $P' \in G(u,v)$.

A.3. For each triangle T in S

a. For each vertex B_i of T and each light vertex L_j

i. Compute 3-D points $F_{ij} = B_i + \|B_i - L_j\| d_{ij}$.

b. Project vertices B_i and points F_{ij} with C and compute the 2-D AABB of the projections.

c. For each $G(u,v)$ touched by the AABB

i. $T_{uv} = T_{uv} \cup T$.

Fig. 4: Top: projection of shadow volume of triangle $B_0B_1B_2$ onto 2-D grid with axes x_G and y_G. Bottom: 2-D AABB $Q_0Q_1Q_2Q_3$ is a tight approximation of the shadow volume projection.

Acceleration scheme

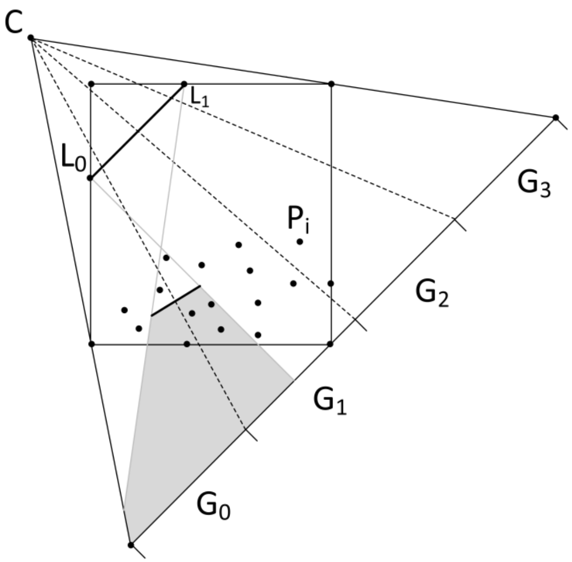


Fig.5: Pixel sample and triangle assignment to grid.

Given a scene S, a rectangular area light source $L_0L_1L_2L_3$, and the pixel samples P of the output image, the acceleration scheme computes a regular 2-D grid G that stores at each cell (u,v) a set of pixel samples P_{uv} and a set of potentially blocking triangles T_{uv} for the pixel samples in P_{uv}. The acceleration scheme proceeds as follows:

A.1. Construct camera C with grid G as image plane.

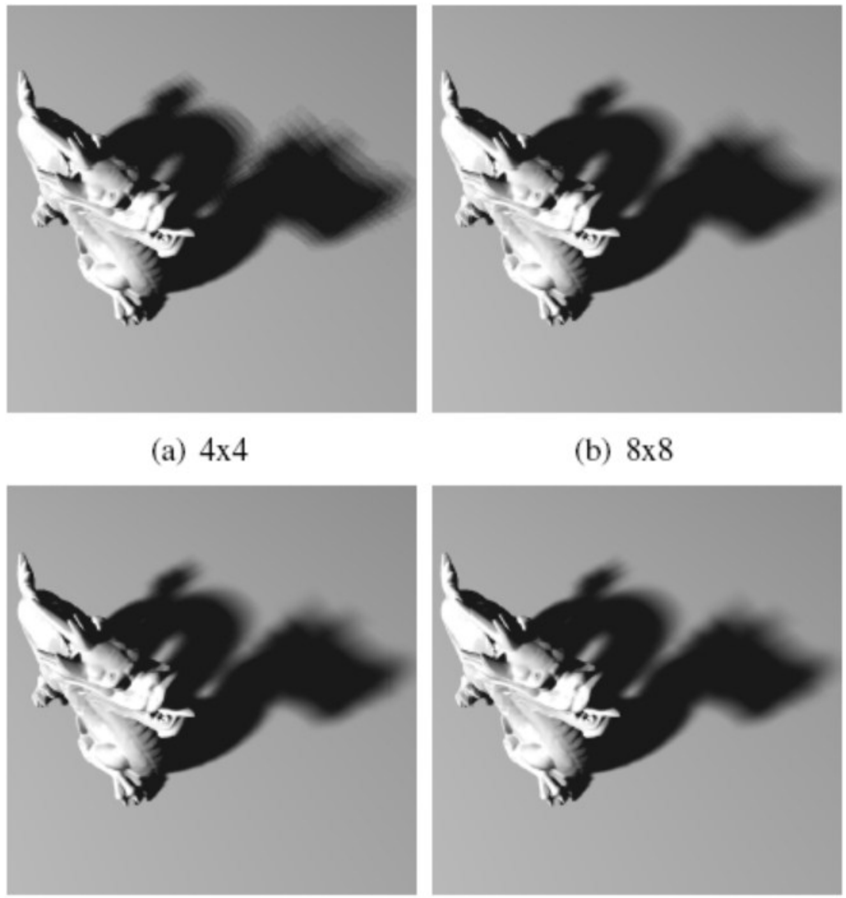
A.2. For each pixel sample P

- a. Project P with C to P'.
- b. $P_{uv} = P_{uv} \cup P$, where $P' \in G(u,v)$.

A.3. For each triangle T in S

- a. For each vertex B_i of T and each light vertex L_j
  - i. Compute 3-D points $F_{ij} = B_i + \| B_i - L_j \| d_{ij}$.
- b. Project vertices B_i and points F_{ij} with C and compute the 2-D AABB of the projections.
- c. For each $G(u,v)$ touched by the AABB
  - i. $T_{uv} = T_{uv} \cup T$.

Results: quality



(a) 4x4 (b) 8x8 (c) 16x16 (d) 32x32

Fig. 6: Quality dependence on resolution of visibility masks.

Results: quality

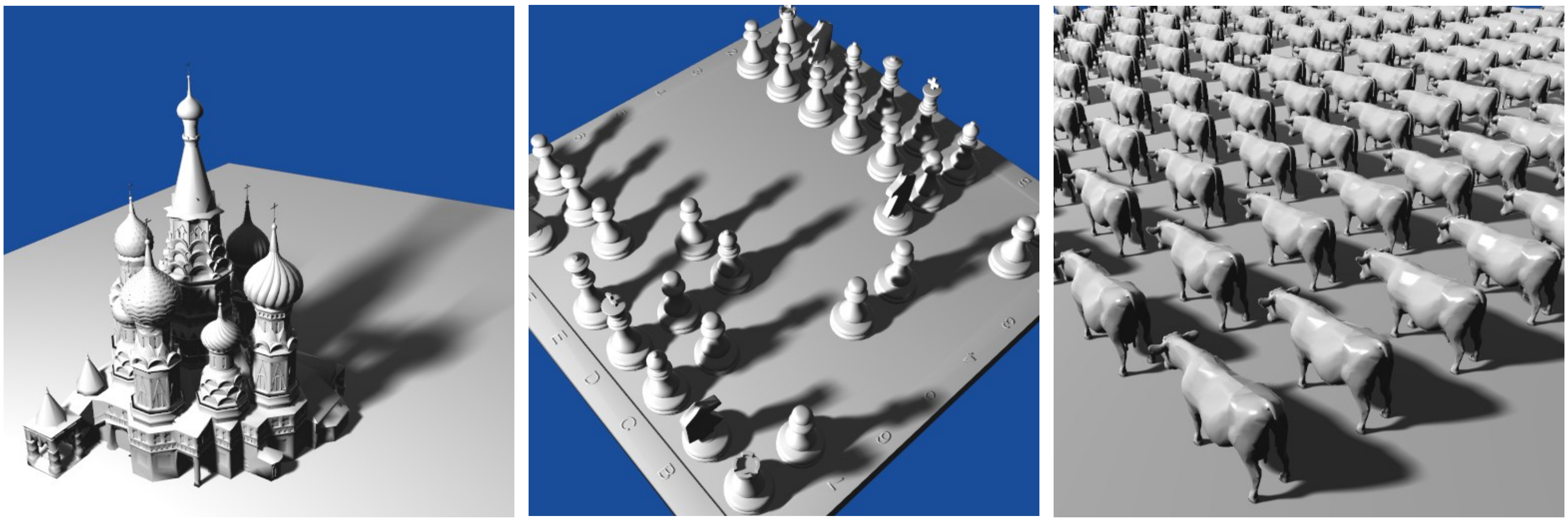


Fig. 7: Additional scenes used to test out method.

Results: performance

TABLE 1: Frame rate [fps] for various output resolutions.

Output res.	256 x256	512 x512	1024 x1024	1280 x1280
Cow Matrix	33.5	24.0	12.5	6.60
Chess	29.1	19.7	10.3	6.10
Church	32.2	20.1	12.0	6.50
Dragon	51.6	24.3	13.5	7.43
Bird Nest	34.9	15.7	4.6	3.16
Spider	60.8	32.7	14.4	11.0

TABLE 2: Frame rate [fps] for various visibility mask resolutions.

Bit mask res.	4x4	8x8	16x16	32x32
Cow Matrix	27.6	26.2	24	20.1
Chess	31.1	23.7	19.7	16.2
Church	34.5	26.3	20.1	15.7
Dragon	36.9	32.7	24.3	16.6
Bird Nest	23.4	18.9	15.7	8.9
Spider	45.8	40.3	32.7	18.6

Results: performance

TABLE 3: Frame rate [fps] for various light source sizes.

Light diagonal	1	2	3	4	5
Cow Matrix	27.2	24	21.4	19.1	17.2
Chess	24.3	19.7	15.3	12.4	7.8
Church	31.2	20.1	15.9	10.1	7.6
Dragon	35.7	24.3	18.4	12.8	9.3
Bird Nest	20.1	15.7	9.8	7.5	5.2
Spider	48.7	32.7	24.3	20	16.8

TABLE 4: Frame rate [fps] for various grid resolutions.

Grid res.	256x256	128x128	64x64	32x32
Cow Matrix	21.1	24	19.7	12.4
Chess	18.2	19.7	17.2	15.3
Church	18.6	20.1	19.1	15.2
Dragon	21.4	24.3	22.1	17.5
Bird Nest	13.1	15.7	12.2	8.7
Spider	29.6	32.7	28.3	23.8

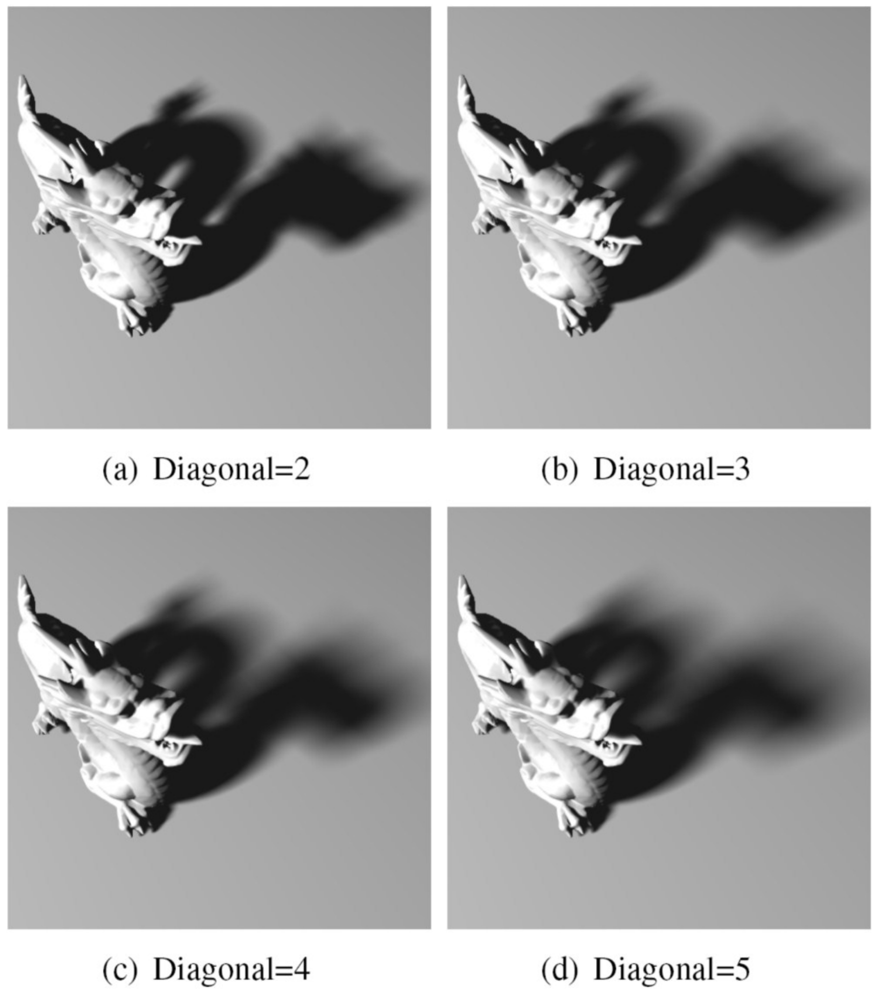


Fig. 8: Soft shadows with various light source sizes.

Results: performance

TABLE 5: Number of triangles and of pixel samples per grid cell.

Scene	Triangles		Pixel Samples	
	Max	Average	Max	Average
Cow Matrix	539	70	278	16
Chess	1333	54	103	11
Church	2121	54	146	9
Dragon	2844	60	283	16
Bird Nest	501	25	243	16
Spider	506	9	244	16

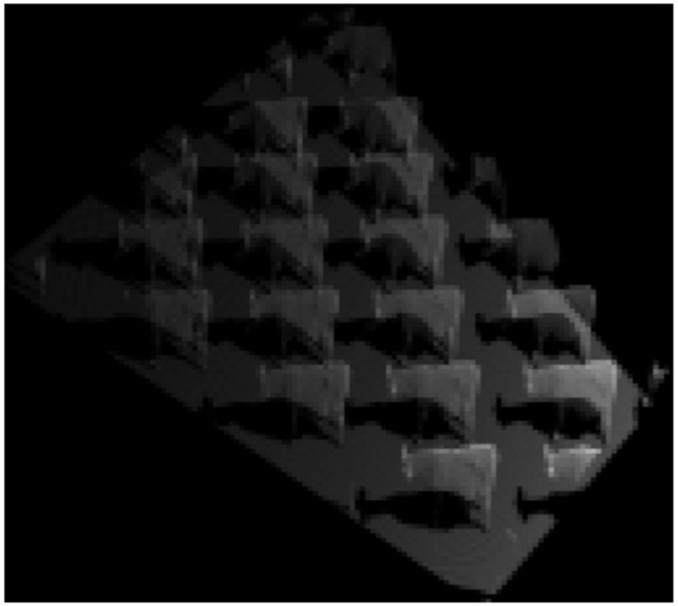


Fig. 9: *Visualization of pixel sample distribution over grid.*

Results: performance

TABLE 6: Triangle to pixel sample assignments [x1,000].

Scene	Necessary	Total	Percentage
Cow Matrix	2,523	22,435	11%
Chess	6,249	44,636	14%
Church	11,554	109,748	11%
Dragon	19,139	91,007	21%
Bird Nest	9,562	35,313	27%
Spider	1,673	9,260	18%

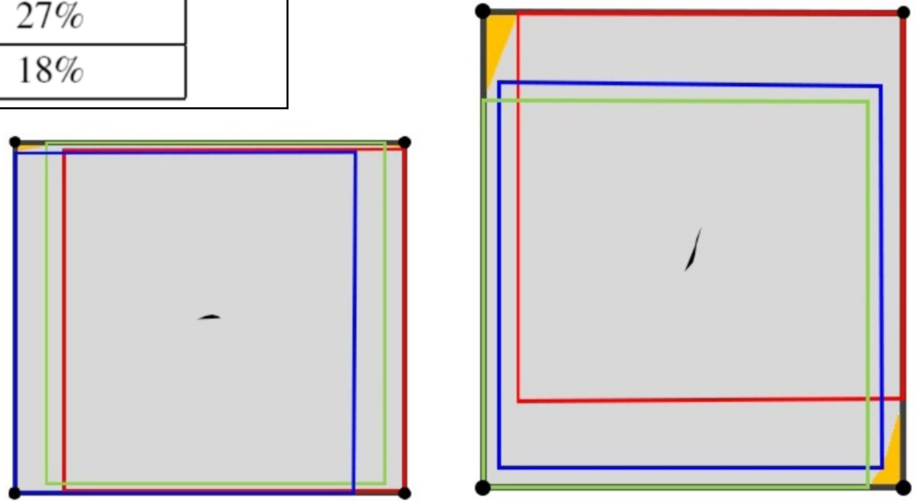


Fig. 10: Visualizations of actual projections of triangle shadow volumes onto grid plane. The difference (orange) between the convex hull (grey) and the AABB are small, as predicted by Fig. 4.

Results: performance

TABLE 6: Triangle to pixel sample assignments [x1,000].

Scene	Necessary	Total	Percentage
Cow Matrix	2,523	22,435	11%
Chess	6,249	44,636	14%
Church	11,554	109,748	11%
Dragon	19,139	91,007	21%
Bird Nest	9,562	35,313	27%
Spider	1,673	9,260	18%

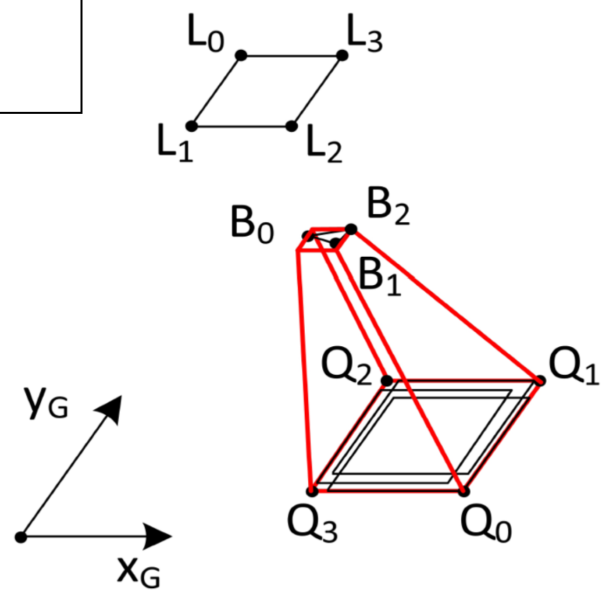


Fig. 11: *Triangle shadow volume approximated conservatively with a frustum (red).*

Performance: comparison to ray tr'ng

TABLE 7: Performance comparison between our method and ray tracing for the same light sampling resolution.

Scene	GEARS [fps] (A)	RT static [fps] (B)	Speedup (A)/(B)	RT dyn. [fps] (C)	Speedup (A)/(C)
Cow Matrix	24.0	1.70	14	1.29	18
Chess	19.7	1.79	11	1.46	14
Church	20.1	2.10	10	1.75	12
Dragon	24.3	2.56	10	1.90	12
Bird Nest	15.7	0.50	31	0.46	34
Spider	32.7	2.70	12	1.80	18

Performance: comparison to ray tr’ng

TABLE 8: Light sampling resolution comparison between our method and ray tracing for the same frame rate.

Scene	Frame rate [fps]	GEARS Bitmask res.	Ray tracing N. of light rays
Cow Matrix	8.3	32x32 = 1,024	48
Chess	10.5	32x32 = 1,024	50
Church	11.0	32x32 = 1,024	90
Dragon	10.0	32x32 = 1,024	72
Bird Nest	15.7	16x16 = 256	12
Spider	32.7	16x16 = 256	18

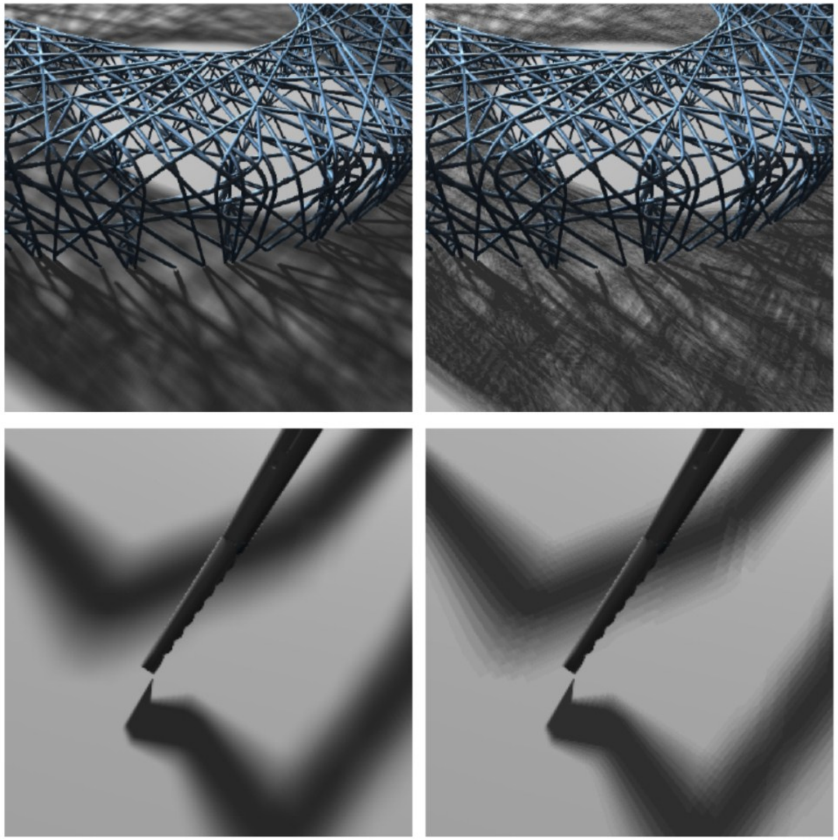


Fig. 12: Quality comparison between GEARS (left) and ray tracing (right) for equal performance.