

Shadow Map Optimization

Mark Colbert

June 17, 2007

The following article formulates the optimization problem for shadow maps without providing any good answer to the problem. For our purpose, a shadow map is a rasterized image of depth information for a scene from a light's perspective. The map is used in rendering to test if a given pixel is visible with respect to a light source. An overview of the method is provided in Algorithm 1.

Algorithm 1 Shadow Map Algorithm

```
1: Bind shadow map framebuffer
2: Set light projection matrix  $\mathbf{P}_l$ 
3: Rasterize scene to shadow map
4: Bind window framebuffer
5: Set camera projection matrix  $\mathbf{P}_c$ 
6: for all p rasterized pixels do
7:   Project p into light space
8:   if depth < shadow map value for p then
9:     light
10:  else
11:    do not light
12:  end if
13: end for
```

Here, the camera projection matrix \mathbf{P}_c can be modeled by a standard projection camera model,

$$\mathbf{P}_c = \mathbf{K}_c[\mathbf{R}_c | -\mathbf{R}_c\mathbf{C}], \quad (1)$$

where, \mathbf{R}_c is the rotation of the camera in world space, \mathbf{C} is the camera's center in world space, and \mathbf{K}_c is the matrix containing the intrinsic parameters associated with the camera. In graphics, the intrinsic parameters typically assume 0 skew, square pixels¹, and a principle point in the center of the image, thus \mathbf{K}_c is of the form,

$$\mathbf{K}_c = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad (2)$$

¹Square pixels does not imply square aspect ratio

where, f is the focal length of the camera.² The light projection matrix has a similar form where the rotation, center and focal length are defined *a priori* by the user,

$$\mathbf{P}_l = \mathbf{K}_l[\mathbf{R}_l | -\mathbf{R}_l L]. \quad (3)$$

The key observation is that the shadow map only stores depth values with respect to the light’s position. Therefore, we can project the scene to any plane within the user-defined frustum and still obtain the same depth values. This is equivalent to using some user-defined projection matrix \mathbf{P}_l , and mapping it to another plane within the frustum via a 3x3 homography \mathbf{H} with eight degrees of freedom,

$$\mathbf{P}'_l = \mathbf{H}\mathbf{P}_l \quad (4)$$

By projecting to any plane within the frustum, we are able to distort the projection of the shadow map to a form where there exists minimum discretization error due to rasterization when reading the shadow map for rendering. In other words, the closer we can get the shadow map to represent the shape of the shadows in screen space, the less aliasing artifacts we will obtain. Here is how we can describe this constraint formally.

Assume we have some arbitrary four-dimensional point (three-dimensions for the position and one-dimension for scale) in both the camera’s frustum and the light’s frustum, \mathbf{X} , then the projection of the point onto the camera’s image/near plane, \mathbf{x}_c can be described as,

$$\mathbf{x}_c \cong \mathbf{P}_c \mathbf{X}, \quad (5)$$

where, the operation is equivalent up to a scale. Similarly, we can define the projection of \mathbf{X} onto the light’s projection plane,

$$\mathbf{x}_l \cong \mathbf{H}\mathbf{P}_l \mathbf{X}. \quad (6)$$

If we operate in Euclidean space, instead of projective, we can remove the scale ambiguity associated with the projective transform by dividing the vectors by their third components.³ We can then represent the points \mathbf{x}_c and \mathbf{x}_l as the following vectors,

$$\mathbf{x}_c = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \mathbf{x}_l = \begin{bmatrix} s \\ t \\ 1 \end{bmatrix}. \quad (7)$$

Obviously, this division removes the linearity of the problem, which inherently makes it more difficult to solve. However, the removal of the linearity also makes it easier to represent the optimization constraint of the homograph \mathbf{H} . Within the (x,y) and (s,t) spaces, we can define the change in area from one mapping to another for a given point in space \mathbf{X} as the determinate of the Jacobian $\mathbf{J}(\mathbf{X})$,

$$\mathbf{J}(\mathbf{X}) = \begin{bmatrix} \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} \end{bmatrix}. \quad (8)$$

²We ignore the actual projection model used within the OpenGL/DirectX API for simplicity. In fact, the projection matrix is more involved than just removing one of the rows of \mathbf{P} , since the depth value must be associated with a given vertex for rasterization.

³This assumes the point is not at infinity.

However, this defines the ratio of the areas between the spaces for only one point \mathbf{X} . In fact, we wish to optimize the problem over the set of all surface points within the frustum of the camera and the frustum of the light. Formally, we can describe this set as $S_{\mathbf{X}}$ such that,

$$S_{\mathbf{X}} = \{\mathbf{X}_i | \mathbf{X}_i \in \text{Frustum of } \mathbf{P}_c \wedge \mathbf{X}_i \in \text{Frustum of } \mathbf{P}_l\}. \quad (9)$$

For our constraint, the goal is to obtain the a mapping that is as close as possible to the shape of the shadows on the screen. This corresponds to the notion that the ratio of areas between (x, y) to (s, t) should be close as possible to the ratio of areas for (s, t) to (x, y) . These are represented respectively as the Jacobian $\mathbf{J}(\mathbf{X})$ and the inverse Jacobian $\mathbf{J}^{-1}(\mathbf{X})$. Therefore, we can formally pose the problem as minimizing the distance between the two ratio of areas for all surface points in set $S_{\mathbf{X}}$,

$$\arg \min_{\mathbf{H}} \forall \mathbf{X}_i \in S_{\mathbf{X}} (\det \mathbf{J}(\mathbf{X}_i) - \det \mathbf{J}^{-1}(\mathbf{X}_i))^2. \quad (10)$$

The immediate problem with using $S_{\mathbf{X}}$ for an optimization problem is the computational complexity. $S_{\mathbf{X}}$ does not represent only the vertices of the geometry within the two frustums, but instead represents every discretized surface point. Thus, we must approximate the set for any reasonable real-time optimization. One approach is to assume the surface points follow a known distribution, such as the normal or poisson distribution. By using either of these distributions, the parameters of the distributions can be approximated from sparse sampling, which may be taken directly from the vertices. To formulate the notion of using a distribution instead of $S_{\mathbf{X}}$, the optimization is performed with respect to the expected value of the distribution,

$$\arg \min_{\mathbf{H}} \mathbb{E} [(\det \mathbf{J}(\mathbf{X}) - \det \mathbf{J}^{-1}(\mathbf{X}))^2]. \quad (11)$$

However, this is still a difficult minimization that most likely needs to be solved numerically and fitted to a function *a priori* to execution.

As another lingering problem, the notion of equal ratio of areas only provides one constraint. Thus, we must still set up the system in such a manner where \mathbf{H} has only one degree of freedom. Typically, this would be done by some observed parameterization of \mathbf{H} that fits well to many cases. However, the problem is this is identical to the work done by many other graphics researchers. While their work does not present the general framework proposed here, most authors are still performing a similar task, where they try to find a homography, referred to as warping, that best adjusts the shadow map to align with the shadows in screen space. One nice attribute about using the general framework is that we can try a variety of parameterizations and even adapt existing parameterizations to adjust and lower distortion with respect to the distribution of surface points.

Now, all we have to do is find a good parameterization and see how well it behaves under a variety of conditions. This is obviously easier said than done. As a first attempt, Kevin proposed using a homography with respect to the camera and lighting direction. Specifically, for a given surface point, \mathbf{X} , he looked at the camera and light position, \mathbf{C} and \mathbf{L} , and defined the normal for the projected plane of the shadow map as an affine

linear combination,

$$\mathbf{n} = \frac{\alpha \cdot (\mathbf{C} - \mathbf{X}) + (1 - \alpha) \cdot (\mathbf{L} - \mathbf{X})}{\|\alpha \cdot (\mathbf{C} - \mathbf{X}) + (1 - \alpha) \cdot (\mathbf{L} - \mathbf{X})\|^2}. \quad (12)$$

The last parameter of the plane was defined by using the point on the viewing frustum that was the farthest from \mathbf{L} . This homography was subsequently multiplied by another homography mapping the virtual plane to the near plane used by the camera's projection matrix, \mathbf{P}_c . Unfortunately, this parametrization, while very effective for some situations, contained many singularities in which the mapping completely breaks down.

Therefore, we are left with two ways to look at the problem. One, we continue to search for an optimal one-dimensional parameterization of the problem for which we can optimize using the presented constraint. Otherwise, we look at finding other constraints to the system, so we can minimize the number of degrees of freedom. Logically, these two approaches are identical, however, this author believes the problem statements represent two different ways of finding a solution.