我们提出了一种新的实时软阴影算法，每个光源使用单个阴影贴图。 因此，我们的算法非常适合于渲染复杂和动态场景，并且可以处理所有可光栅化的几何形状。 该方法的关键思想是将阴影贴图用作场景的简单且统一的离散表示，从而使我们在大多数情况下可以生成逼真的柔和阴影。 特别是，它自然可以处理封堵器融合。 而且，我们的算法可以高精度处理矩形光源和纹理光源，并且可以很好地映射到可编程图形硬件。

介绍

实时渲染逼真的柔和阴影是计算机图形学中的一个基本问题。 除了提高渲染图像的真实感外，它们还简化了对象之间空间关系的识别。 从实际的角度来看，点光源会产生所谓的硬阴影，其中在光和本影之间会出现清晰的过渡。 但是，由于大多数光源是扩展的（区域或体积），所以光的强度从无阴影到完整阴影会平滑变化，从而生成具有半影区域的柔和阴影。 渲染硬阴影仅需要计算两个点（阴影点和灯光）之间的可见性，而软阴影则需要对从阴影点可见的光源多少进行复杂的评估，通常以可见度的百分比表示。

针对实时非专用软阴影渲染应用程序，一种非常合适的算法应具有以下功能:

实时处理动态和复杂场景,

不依赖于接收者和遮挡物的几何形状（例如网格，点云，图像...），

产生尽可能逼真的阴影.

**相关工作**：基于对象空间轮廓检测的最新优化技术为合理复杂的场景（例如半影楔[AAM03，ADMAM03]）实时提供近似的软阴影。 但是，除了一些缺点（例如，错误的咬合器融合）外，它们仍然局限于流形网格，并且其复杂性随场景的复杂性而增加，这使得第一和第二标准都很难满足。 另一方面，阴影贴图是基于图像的技术，可支持任何类型的可光栅化几何体：网格，点云或具有alpha纹理的模型（通常用于表示叶子或金属丝网）。 它们对场景的复杂性也不太敏感，由于这些原因，我们将重点放在第二类技术上（请注意，最近的关于实时软阴影的调查可以在[HLHS03]中找到）。

阴影贴图算法[Wil78]首先从光源渲染场景的深度图。使用称为阴影图的深度图，可以进行简单的深度比较，确定最终图像的哪些像素点亮或不点亮。这种廉价的过程允许在每一帧实时地从头开始生成阴影。不幸的是，阴影贴图也表现出一些缺点。一种是采样分辨率不足时硬阴影的混叠边界。可以通过增加有效的阴影贴图分辨率[FFBG01，SD02，MT04，WSP04]来解决此问题，也可以使用百分比接近滤波技术[RSC87]将其替换为模糊。另一个基本问题是对硬阴影的限制，因此，已经在阴影映射算法的基础上建立了几种最新的实时软阴影方法。有些需要为每个光源渲染多个阴影贴图[ARHM00，HBS00]，将其仅用于静态场景。其他处理动态场景的人会从单个光样本中渲染柔和阴影，即使这会产生众所周知的伪影，因为只有从光样本中可见的对象部分才被视为遮挡物。但是，除了这个缺点之外，这种现有技术还受到其他几个重要限制。例如，有些仅限于平面接收器[SS98]，而另一些则不适当地考虑了封堵器的形状以及封堵器融合[BS02]，并且当原始隐藏的阴影出现时也会产生弹出效果[AHT04]。一些基于阴影贴图和轮廓提取的混合方法[CD03，WH03]只能计算外部半影部分的粗略近似值（相对于硬阴影边界）。基于图像的算法仍然必须加以改进，以提供具有更逼真的柔和阴影的动态场景的实时渲染。

在最近的一项工作中[AHL \* 06]，Atty提出了一种软阴影算法，该算法的阴影可见度计算采用一定百分比的可见性计算，就像我们的方法一样，是基于阴影图样本的反投影。 但是，与我们的算法不同，他们的方法依赖于一个非常严格的假设：阻塞器和接收器是不相交的集合。 此外，它们仅限于非常小的阴影贴图分辨率（它们报告为200×200像素），并且由于所有可见性计算都是在离散的阴影贴图空间中进行的，因此阴影更易混叠。

**贡献**：我们提出了一种新的软阴影算法。我们将阴影图视为场景的离散化表示，而不是将阴影图的使用限制为简单的深度查询，每个样本都是一个小的潜在遮挡物。 关键思想是使用这种简化的场景表示来计算场景点和扩展光源之间的可见性百分比。为了提供实时性能，我们的算法为每个光源使用单个阴影图。尽管有这种近似值，但在大多数情况下，我们的算法仍可以通过几乎正确的封堵器融合来提供逼真的柔和阴影。此外，它可以处理任何类型的可光栅化几何图形，可以处理矩形和纹理光源，并且易于集成到现有应用程序中。通过使用模块化精度可以保证实时性能，并且最终不需要任何预先计算。

2 我们的软阴影算法

即使我们的算法自然地处理了多个矩形光源和矩形阴影贴图像素，为了简化说明，我们还是在宽度为的单个正方形光源和正方形像素上介绍了主要过程。

我们的算法为每个光源计算一个软可见性缓冲区(V-buffer),该缓冲区在最终渲染过程中调制漫反射和镜面照明.可见性缓冲区为屏幕的每个像素存储可见性因子,从而提供单个像素看到的光的百分比();如果,则全亮;如果,则在全阴中,在半影时,.

我们的方法要解决的关键问题是如何对V-buffer进行快速而准确的计算.该算法分为两个步骤：首先，它从光线中计算阴影图，然后，对于V缓冲区的每个像素，如果其对应点p在半影中，则通过计算光线来评估其可见度 阴影贴图样本的子集遮挡的区域.阴影图样本的这个子集称为搜索区域,用A表示.在第2.1节中详细介绍了可见性计算,而在第2.2节中介绍了优化（即半影分类和搜索区域A的选择）.

2.1 可见性计算

阴影图采集

可见性缓冲区计算的第一步是获取存储线性光空间深度值的阴影图。此步骤需要定义一个投影台（如图2a所示），其原点位于光源中心.近平面及其边界与光源平行.它与原点的距离为，宽度为.可以任意选择其他视锥参数,或者更好地动态选择视锥参数以优化有效阴影贴图分辨率.最后,阴影图分辨率应为2的幂,以简化分层版本的构造(第2.2节).

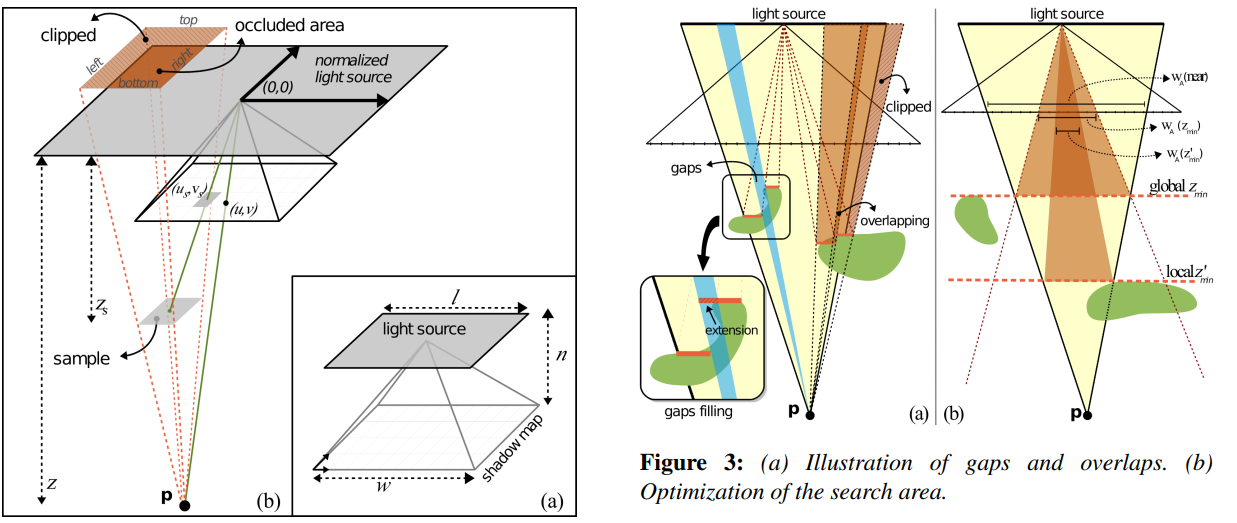
可见性过程

给定阴影图,可见性遍的目标是计算场景中每个可见点的可见性因子.因此,给定着色点,令和分别为其投影到阴影图上的坐标(以像素为单位)和其光空间深度值(图2b).计算其可见性因子的基本算法很简单.我们首先假定该点是完全可见的(),然后从中删除阴影图中存储的每个可能的遮挡样本遮挡的面积.

坐标和深度样本所占的面积计算如下.它的对象空间表示,即与光源和阴影图平行的正方形,从当前点投射到光源平面上(图2b).此投影是一个平行于光源边界的矩形.令为归一化光源空间（即二维光空间）的边界，该二维光空间按比例缩放以使光的大小为1(图2b).因此,由下式给出:

其中,是样本在物体空间中的大小,是样本平面和光源平面之间的比例.最后,通过将边界B限制在[-1/2,1/2]来给出光源与样本之间的交点,并从中减去归一化的遮挡区域.

请注意，到目前为止，我们将深度值小于z的阴影贴图的每个样本视为潜在的遮挡物集合.我们在2.2节中展示了如何大幅减少这组样本(即搜索区域A).



间隙填充

由于阴影图的不连续表示，样本之间可能会出现小的重叠和间隙（如图3a所示）.在重叠区域中，某些亮点被去除了两次，因此产生了稍暗的半影。另一方面，在间隙区域中，遮挡的光点不会被移除，这可能会在本影区域产生不必要的光（请参见图7d）。尽管重叠误差是完全可以接受的，但必须消除间隙瑕疵。为了填补这些空白，我们假设它们出现在阴影贴图的相邻样本之间，然后扩展样本的左边界和底边界，以使所有遮挡样本都加入二维光空间中（图3a）。尽管先前的假设在1D中是正确的，但在2D中却并非总是如此，因为样本在两个方向上移动。这解释了为什么我们仅扩展样本以填补空白，为什么我们也不能修剪样本以消除重叠。因此，此过程稍微增加了重叠误差，但是在有效地填补空白的同时，仍然很少察觉到。我们还指出，只要重叠遮挡物存在两个遮挡物，其中一个遮盖物靠近光源，而另一个遮挡物靠近当前阴影点，重叠误差就会变大。幸运的是，这种极端情况在实践中很少发生。总结一下,对于给定的遮挡样本：

1. 使用公式1计算其边界,
2. 通过调整等式1,计算其领域的边界.
3. 取和之间的最小值作为的最终值.
4. 上述相同的流程用于.

显然,当且仅当该邻居也是潜在的遮挡者,即,如果其深度小于当前接收器深度,则样本才向其邻居之一扩展.

纹理光源

上面介绍的方法使用每个样本所占面积的快速分析计算.该方法可以很容易地以与[AAM03]中类似的方式适用于处理纹理化和动画化的光源（例如火）.实际上，定义样本遮挡的光区域的四个标量值B可以直接用于索引存储该区域的归一化光源颜色的4D纹理。 这样，我们可以处理任何类型的面光源。 有关此方法的所有详细信息，请参见[AAM03]。 4D纹理的使用会产生离散化伪像，并且纹理存储起来很昂贵。 由于我们的遮挡区域是与光线平行的矩形四边形，因此，使用汇总面积表（SAT）[Cro84]可以大大减轻照明效果和提高准确性.纹理光源如图4所示.

2.2 优化

通过减少潜在的遮挡样本数量以及仅对潜在在半影中的像素执行昂贵的半影计算，可以显着提高性能。 实际上，第二个优化需要第一个。

为了实现这些优化，我们首先必须构建一个层次阴影图（HSM），该阴影图为每个像素存储遮挡物样本的最小和最大深度值（级别0只是阴影图本身）.与mipmap一样，通过将分辨率降低两倍（而不是平均值）来迭代构建其他级别，每个低级像素都存储最小和最大覆盖深度值。 在实践中，我们的HSM有效地存储在硬件映射纹理中，从而使这些还原步骤可以由GPU有效执行：在每个步骤中，使用执行片段最小和最大操作的琐碎片段着色器将最大级别渲染到较低级别.

搜索区域缩小

为了减少潜在的遮挡样本的数量,我们准确地评估了搜索区域A.由于遮挡样本位于由光四边形和当前点形成的矩形金字塔内部，因此，通过取之间的交点可以获得第一粗略近似 阴影贴图和金字塔的近平面（图3b）.那么子集A是一个以（p在阴影图上的投影）为中心的宽度为的正方形(如果光源是矩形,则是矩形).可以通过两个步骤来改善这种近似.首先,HSM的顶层为我们提供了存储在阴影贴图中的最小深度值.因此,金字塔可以被深度为的平面夹紧（图3b），新搜索区域的宽度变为:

然后，通过访问HSM中的适当级别,快速找到新搜索区域的局部最小深度值(图3b)(请注意,需要进行四个纹理提取).使用等式2和而不是计算的新值.当搜索区域显着减小时,可以重复此步骤,但是实际上,我们发现一个步骤就足够了.

半影分类

现在，我们希望对潜在的半影像素进行快速分类,以减少可见度因子的昂贵计算.对于当前点,一旦评估了准确的局部深度边界值和(在搜索区域缩小期间),我们将比较其深度值:

如果，则像素完全点亮,并且,

如果，则该像素被视为遮挡且,

否则,像素可能处于半影中,必须精确计算.

自适应精度

最后，我们算法的速度主要取决于搜索区域的像素大小，这对于很大的半影很重要。就视觉质量而言，大半影比细影需要更少的细节.因此,当实时性能需要更快的计算时,我们通过为每个点p动态选择HSM中的级别来降低大半影的精度,以使搜索区域不超过给定的大小阈值(图5).因此,对于大半影,我们不对阴影图的许多样本(HSM的0级)进行反投影,而是从较粗略的HSM中投影较少但较大的样本。由于我们无法为HSM的每个像素存储任何覆盖范围信息（覆盖值取决于接收器深度），因此我们仅将所选粗略级别中的最小深度值用作采样深度zs。当场景由大的半影区域组成时，这种近似会导致较小的视觉质量下降，这将在4.1节中进行介绍和讨论。

3 实现

我们的软阴影映射算法可以通过几种方式实现，具体取决于应用程序和硬件。 由于没有通用的最佳解决方案，因此我们介绍实现并讨论其变体。

延迟着色策略

为了独立于场景的复杂性，我们使用了类似延迟着色的策略：在每一帧处，场景都是从视点首次渲染而没有任何阴影计算，但对于阴影贴图，是渲染到单个分量浮点中 存储线性深度值的缓冲区。 在我们的实现中，此深度缓冲区仅由可见性遍历用于计算V缓冲区（每个光源一个），因此，将为最终渲染遍第二遍渲染整个几何体（此遍可利用早期z- 剔除）。 我们选择这种方法，而不是采用完全延迟的阴影方法，因为它更灵活，并且提供了更好的硬件抗锯齿支持。

动态分支与多次渲染

计算可见性缓冲区的最简单方法是对一个片段着色器使用一次通过，执行以下三个步骤：