第一部分

光线追踪基础

*Chris Wyman*

**5**

第4章

一个天文馆圆顶主摄像机

John E. Stone

伊利诺伊大学厄本那－香槟分校附属

贝克曼高级科学技术研究所

摘要

本章介绍了一种实现了使用方位角等距投影对天文馆圆顶主图像进行高质量交互式光线跟踪的摄像机。光线跟踪非常适合实现各种特殊的全景和立体投影而不牺牲图像质量，该相机实现支持抗锯齿，景深焦点模糊和圆形立体投影，而使用传统光栅化和图像变形难以高质量地产生所有效果。

（译者注：天文馆圆顶是指天文台观测用的房子，它的屋顶与众不同，不是方的、长的或斜的，而是圆的。这有它的特殊用途，天文台远远看上去，只不过是半个圆球，可是走近一看，圆球上却有一条宽宽的裂缝，原来是一个巨大的天窗，庞大的天文望远镜就是通过这个天窗指向辽阔的天空。感兴趣的读者可以自行了解一下天文台的构造，这里不再赘述。作者的意思是，这篇文章所描述的渲染用相机，就像是天文台里面的这个天文望远镜一样。）

4.1 介绍

天文馆圆顶主图像在一个黑色正方形内对180°的半球形视野进行编码，把包含整个视野的内切圆形图像投射到天文馆圆顶上。圆顶主图像使用所谓的方位角等距投影产生，并与真实世界180°等距鱼眼镜头的输出紧密匹配，但没有真正的镜头的瑕疵和光学像差。有许多使用光栅化和图像变形技术的方法可以创建圆顶主投影，但直接光线跟踪比其他替代方案具有特别的优势：最终圆顶主图像具有均匀的采样密度（在过采样区域没有像翘曲立方体投影或许多平面透视投影那样有浪费掉的采样[3]），支持立体渲染，并支持本征曲面焦平面上的景深。通过在科学可视化软件中集成圆顶主图像的交互式渐进光线跟踪，更广泛的科学可视化材质可以在公共球幕投影场地[1,5,7]中使用。

4.2 方法

使用方位角等距投影形成圆顶主图像，如图4.1所示。圆顶主图像通常在正方形视口内计算，以180°视野渲染以填充正方形视口中的内切圆，内切圆刚好碰到视口的边缘，而其它的地方就用黑色背景填充。从上方或下方看，圆顶主投影可能看起来大致类似于圆顶半球的正投影，

**39**

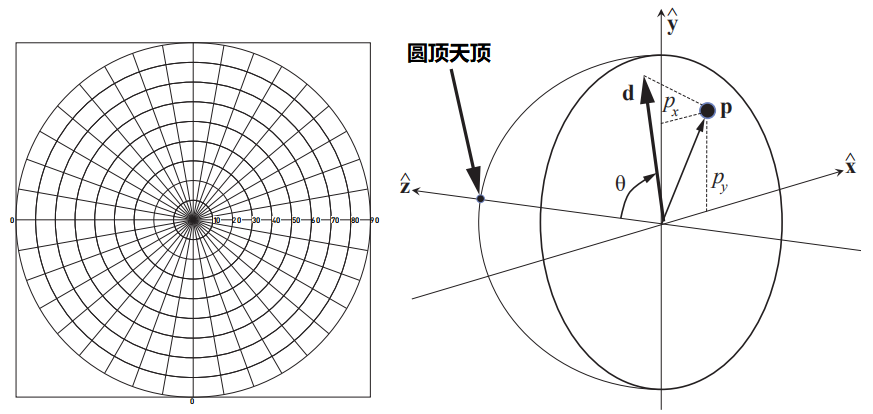


图4.1：圆顶主图像使用方位角等距投影，看起来类似于180°鱼眼镜头的照片。左：圆顶主图像有在10°处绘制的纬度（圆圈）和经度（线）的明显均匀间距投影180°视野的间隔。像素到视口的距离中心与投影中心的真实角度成正比。对：圆顶主图像平面中的矢量p，方位角方向分量px和py，光线方向db，光线方向db之间的角度θ和圆顶天顶，以及相机正交基矢量xb，yb和bz。

但是具有的关键差异是圆顶主图像纬度环之间的间距是均匀的，这种均匀的间距方便地允许光线追踪相机在图像平面中采用均匀采样。图4.2显示了（一个点）在图像平面中的位置与其在圆顶半球上产生的光线方向之间的关系，图4.3显示了使用此处描述的相机模型生成的光线追踪圆顶主图像的示例序列。

4.2.1 从视口坐标计算光线方向

圆顶主摄像机在几个关键步骤中计算主要光线方向，从中心开始的最大视野角度按照整个视野的一半计算，例如，对于典型的180°视野，是90°或弧度。对于方位角等距投影，每个像素到视口中心的距离与用弧度表示的相对于投影中心的真实角度是成比例的。圆顶主图像通常是正方形的，因此对于一张具有180°视野的4096×4096的圆顶图像，我们将具有一个在两个维度上都是的弧度/像素缩放因子。对于图像平面中的每个像素，计算像素I和视口的中点M之间的距离，然后乘以视野的弧度/像素缩放因子，产生一个以弧度为单位的二维矢量，。然后从两者距离视口中心的距离分量和计算出P的长度，产生以弧度为单位的相对圆顶的天顶（顶部那个点）的真实角度θ，那么计算θ的关键步骤就是

(4.1)

和

(4.2)

**40**

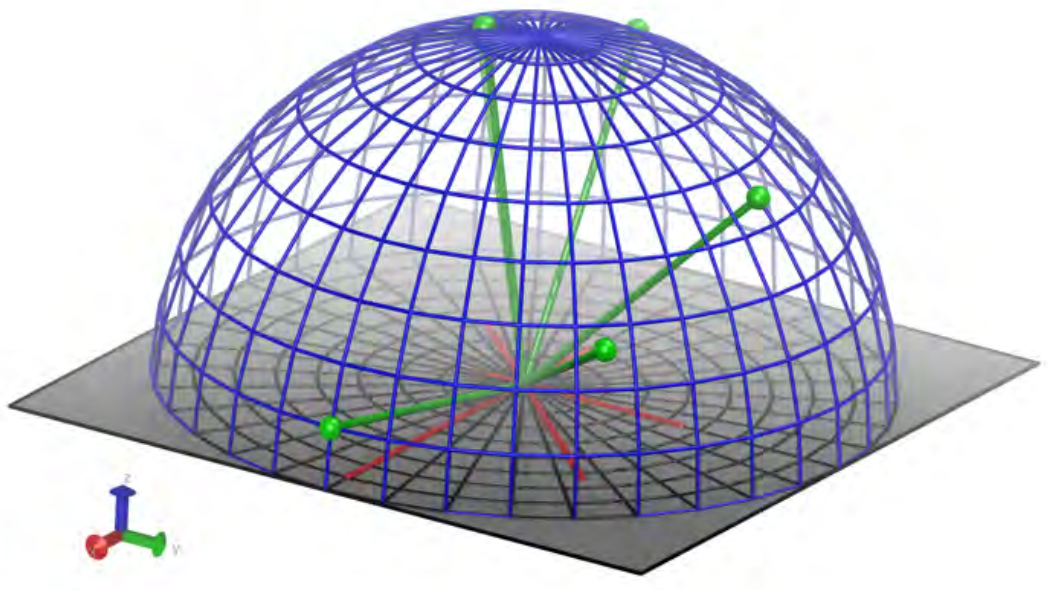


图4.2：关于图像平面（灰色正方形与内切纬度/经度线）、圆顶半球（蓝色）、图像平面中的示例向量p（红色）和圆顶表面上的对应光线的方向（绿色）的视觉描绘。

对于具有180°视场的圆顶主，角度θ与通过p计算得到的光线仰角是互补的。

重要的是要注意θ既用作距离（从视口中心开始，按弧度/像素缩放）也用作角度（从圆顶的天顶）。为了计算光线的方位角方向分量，我们通过用p除以θ计算出，此处用作长度。对于θ= 0，主要光线指向圆顶的天顶，这种情况下方位角是不确定的，所以我们要防止在这种情况下出现除零。如果θ大于，则像素在圆顶的视野外故涂成黑色。对于零和之间的θ值，圆顶坐标中的归一化光线方向是

(4.3)

如果每条光线需要（用到）正交向上（）和向右（）的方向向量，例如景深，可以使用现有的中间值廉价（译者注：不消耗太多的计算资源）地确定它们。可以通过将光线方向作为θ的函数的导数取反来计算向上的方向向量，产生一个指向圆顶的天顶并与经度线垂直的线对齐（译者注：此处根据它的数学定义应该描述为垂直于光线方向即切线并且是朝向圆顶的天顶方向，原文意思有点令人混淆）的单位向量

(4.4)

可以完全根据方位角的方向分量和确定向右的方向，产生与水平纬度线对齐（译者注：此处根据它的数学定义应该描述为与水平纬度线的切线对齐即平行，原文意思有点令人混淆）的单位向量，

(4.5)

**41**

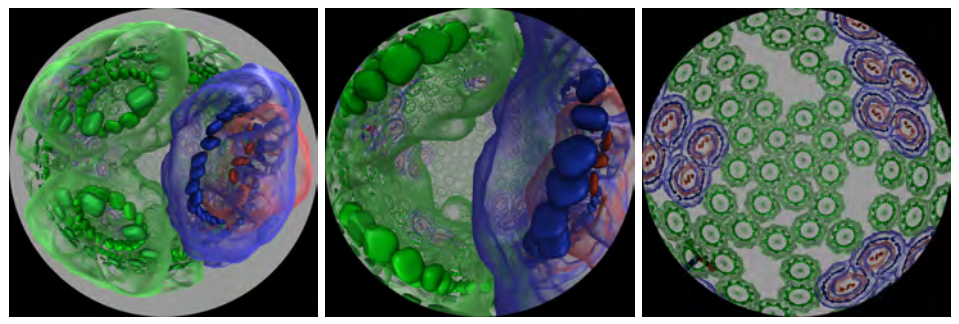


图4.3：在通过OptiX加速的Visual Molecular Dynamics（VMD）[4,7]中交互式渲染的一系列圆顶主图像。序列显示摄像机飞入紫色细菌中的光合小泡。由于它的结构主要是球形的，当相机到达囊泡中心时，即最右边的图像，这时的圆顶投影完全变成平的了。

有关在圆顶坐标系中计算光线、向上和向右的方向向量的简约示例，请参见清单4.1。最后，将光线方向从圆顶坐标系转换到世界坐标系，我们将其组件投影到相机的正交基原点向量，和上

(4.6)

如果需要，同样的坐标系转换操作也必须用在向上和向右的向量上。

1. **static** \_\_device\_\_ \_\_inline\_\_
2. **int** dome\_ray (**float** fov,  // 以弧度为单位的视野FOV
3. float2 vp\_sz,  // 视口大小尺寸
4. float2 i,  // 图像平面上的像素／点
5. float3 & raydir,  // 返回的光线方向
6. float3 & updir,  // 返回的向上方向，与经度线对齐
7. float3 & rightdir) { // 返回的向右方向，与纬度线对齐
8. **float** thetamax = 0.5 f \* fov;  // 以弧度为单位的视野FOV的一半
9. float2 radperpix = fov / vp\_sz;  // 通过X/Y计算弧度／像素
10. float2 m = vp\_sz \* 0.5 f;  // 计算视口中心／中点
11. float2 p = (i - m) \* radperpix ;  // 计算方位角，分量
12. **float** theta = hypotf(p.x , p .y); // hypotf ()确保最佳精度
13. **if** (theta < thetamax) {
14. **if** (theta == 0) {
15. // 在圆顶中心，方位角是不确定的而我们必须避免，
16. // 除零，所以我们将光线方向设置成指向圆顶的天顶
17. raydir = make\_float3(0, 0, 1);
18. updir = make\_float3(0, 1, 0);
19. rightdir = make\_float3(1, 0, 0);
20. } **else** {
21. // 正常情况下：计算＋混合方位角和仰角分量
22. **float** sintheta, costheta;
23. sincosf(theta, &sintheta, &costheta);
24. raydir = make\_float3(sintheta \* p.x / theta,
25. sintheta \* p.y / theta,
26. costheta );
27. updir = make\_float3(-costheta \* p.x / theta,
28. -costheta \* p .y / theta,

**42**

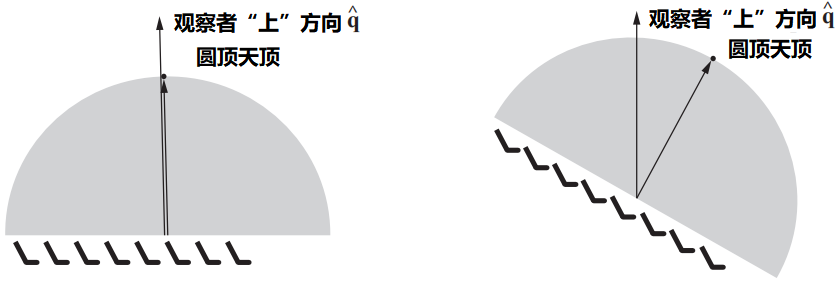


图4.4：圆顶的天顶与观众“向上”方向在传统的平面天文馆圆顶（左）和一个更现代化的有着30◦倾斜的体育场式座位的圆顶剧院（右）中的关系。

1. sintheta);
2. rightdir = make\_float3(p.y / theta, -p.x / theta, 0);
3. }
4. **return** 1; // 图像平面中的点是在视野内的
5. }
6. raydir = make\_float3(0, 0, 0); // 在视野外
7. updir = rightdir = raydir;
8. **return** 0; // 图像平面中的点在视野外
9. }

代码清单4.1：这个简短的示例函数演示了从图像平面中的一个点计算出从圆顶半球的地板射出的光线方向所需的关键算法，给定由用户指定的视野角度（通常为180°）和视口尺寸。相对于投影中心的圆顶角度与从视口中心到图像平面中的指定点的距离是成比例的。此功能是为天顶为圆顶正z方向的圆顶半球编写的，此函数返回的光线方向必须由调用此函数的代码自行投影到相机基础向量上。

4.2.2 圆形立体投影

圆顶投影焦平面的非平面全景性质呈现出了立体渲染的特殊挑战，非立体圆顶主图像可以通过多级渲染、变形和过滤来合成许多传统的透视投影，而高质量的立体输出基本上需要对每一次采样进行单独的立体摄像机计算（因此在光线追踪时是对于每一条光线）。这在使用现有的光栅化API时导致了显著的性能开销和图像质量的权衡取舍，但这确实非常适合交互式光线追踪。数学自然延伸了上一节中概述的光线计算，并且相对于渲染一对单视场图像而言，引入了不显著的性能成本。

使用圆顶主摄像机的立体圆形投影[2,6,8]，每个射线的原点向左或向右移动一半的双眼距离。这个偏移是沿着垂直于两者射线方向（）和观众自己的圆顶的天顶或“向上”方向（）的立体双眼轴向。

**43**

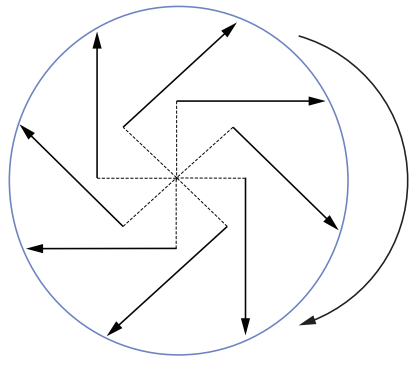


图4.5：圆形立体投影技术和将每条光线的原点根据光线方向进行相当于一半的双眼间距的偏移后的效果的示意图，该图显示了用于左眼投影的眼睛移位偏移（虚线所示）。

这解释了包括图4.4中所示的各种倾斜的圆顶配置，通过O = O + e（×）计算光线移位后的原点坐标，其中e（）是如图4.5所示的这样应用移位方向和缩放因子来把眼球正确地移动到世界空间的眼睛位置的眼睛移位函数。我们通过对每条光线独立计算眼睛的立体位移获得圆形立体投影。

虽然圆形立体投影不完全没有失真，但它们“在你观察的地方总是正确的”[6]，圆形立体投影在观众看向立体投影的地平线时是最为正确，但是当观看天顶（）附近时却不是这样。当立体极轴后面的区域可见时，观看者可以看到反立体（译者注：反立体又称反立体效应，即立体观察时，获得的光学立体模型与实地物体的凹凸变化，以及景物的远近正好相反的现象）图像。为了有助于缓解这个问题，分离的立体眼睛可以调制为相对于观众立体赤道或地平线（圆顶的底面）的仰角的函数。通过将眼睛的间距在观众的天顶上调制为零（从而降级为单视场投影），反立体的观察倾向可以在很大程度上得以消除。请参见清单4.2的简单而具有代表性的实例。

1. **static** \_\_host\_\_ \_\_device\_\_ \_\_inline\_\_
2. float3 eyeshift(float3 ray\_origin, // 原始的非立体眼睛原点
3. **float** eyesep,      // 世界坐标系下的两眼间距
4. **int** whicheye,      // 左／右眼标记
5. float3 DcrossQ) {  // 光线方向 x 观察者的向上方向
6. **float** shift = 0.0;
7. **switch**(whicheye) {
8. **case** LEFTEYE:
9. shift = -0.5 f \* eyesep; // 移位光线的原点的左边
10. **break**;
12. **case** RIGHTEYE:
13. shift = 0.5 f \* eyesep;  // 移位光线的原点的右边
14. **break**;
16. **case** NOSTEREO:
17. **default**:
18. shift = 0.0; // 单视场投影

**44**

1. **break**;
2. }
4. **return** ray\_origin + shift \* DcrossQ;
5. }

代码清单4.2：最精简的处理立体和单视场投影的眼睛位移函数实现。

通过把两个立体子图像按照上/下布局输出到相同的输出缓冲区中的方式进行渲染，在两倍高度的帧缓冲的上半部分和下半部分分别存放左、右眼的子图，实现在单个批次（pass）中计算出立体圆顶主图像。图4.6显示了垂直上/下堆叠方式布局的立体帧缓冲区，这种方法集合了每帧中光线追踪作业的最大并行数据量，从而减少API开销并提高硬件调度效率。现有的硬件加速光线追踪框架缺乏在相机和输出缓冲区列表上执行渐进式光线追踪的有效机制，所以打包的立体相机实现可以更容易地采用渐进式渲染来实现交互式立体圆顶可视化。这在使用视频流（译者注：流媒体）技术远程查看、云端的渲染引擎的实时结果时特别有益。垂直堆叠布局的立体子图像的关键优点是任何图像后处理或显示软件都可以用简单的指针偏移算法轻易地访问这两个彼此独立的立体子图像，因为它们在内存中是连续的。通过圆形立体投影制作的圆顶主图像和电影通常可以直接进入直接导入传统的图像和视频编辑软件中，可以使用和用于传统的透视投影相同的工具来执行大部分的基础编辑和后处理。

4.2.3 景深

圆顶主投影的景深焦点模糊可以通过计算景深的弥散圆的基向量来实现，并随后使用基向量计算光线的原点偏移抖动，最后更新光线方向。弥散圆基向量和最好是和光线方向一起计算，因为它们都依赖于相同的中间值。公式4.4和4.5分别描述了和的计算，一旦使用和计算抖动的景深光线原点，光线方向必须要更新。通过减去光线与焦平面（在这种情况下为球体）相交的点作为原点的新光线来计算和更新光线的方向并规一化（计算）结果。请参见清单4.3的简单实例。

1. // 计算一条新光线的原点和光线方向的CUDA设备函数，
2. // 给定弥散圆的半径，
3. // 每条光线的正交向上和向右基向量，
4. // 焦平面／球体距离，随机／快速随机种子／状态向量。
5. **static** \_\_device\_\_ \_\_inline\_\_
6. **void** dof\_ray(**const** float3 & ray\_org\_orig, float3 & ray\_org,
7. **const** float3 & ray\_dir\_orig, float3 & ray\_dir,
8. **const** float3 & up, **const** float3 & right,
9. unsigned **int** & randseed) {
10. float3 focuspoint = ray\_org\_orig +

**45**

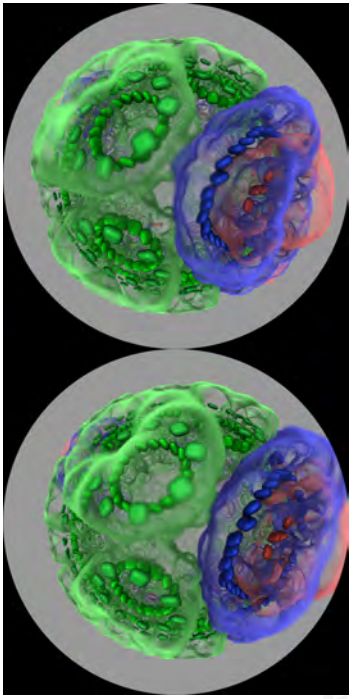


图4.6：垂直堆叠的立体对圆顶主图像以a。渲染最后单次通过，在球面焦平面上施加景深。

1. (ray\_dir\_orig \* cam\_dof\_focal\_dist);
2. float2 dofjxy;
3. jitter\_disc2f(randseed, dofjxy, cam\_dof\_aperture\_rad);
4. ray\_org = ray\_org\_orig + dofjxy.x \* right + dofjxy.y \* up;
5. ray\_dir = normalize(focuspoint - ray\_org);
6. }

代码清单4.3：这个简短的示例函数说明了计算所需的关键算法使用景深时的新光线原点和方向。

4.2.4 抗锯齿

圆顶主图像的抗锯齿无需考虑任何异常因素，通过对连续的几次采样进行视口坐标系的抖动就能很容易地实现。对于交互式光线追踪，对样本进行简单的（盒型，译者注：原文使用box，具体是个什么形状不太好描述，但是不影响理解）平均值过滤很节省（计算资源）并且容易实现。由于视野外采样到的是黑色，抗锯齿采样还用于从圆顶主图像中产生的圆形图像提供平滑的边缘。

4.3 Planetarium Dome Master Projection示例代码

本章提供的示例源代码是为NVIDIA OptiX编写的API，是使用CUDA GPU编程语言编写的。

**46**

尽管示例代码为了简单起见而被删减，使用一个很小的辅助函数来显示关键的全局摄像机和场景参数，例如在磁盘上生成统一的随机样本用来计算景深，以及类似的任务。提供这些让读者可以更容易地根据自己的需要去解释和调整示例的实现。

圆顶主摄像机以摄像机函数模板的形式实现，在OptiX光线追踪框架的几个主要光线生成“程序”中实例化。该函数接受STEREO ON和DOF ON这两个模板参数以分别启用或禁用立体圆顶主图像和景深焦点模糊。通过单独创建相机函数的实例，与禁用掉的功能相关的算法运算被去除掉（译者注：其实就是通过预编译宏来生成不同的shader变体而已），这对复杂场景的高分辨率交互式光线追踪特别有利。

感谢

这项工作的研究支持者有国家卫生研究院的P41-GM104601; NCSA高级可视化实验室;而CADENS项目的研究支持者是NSF奖项ACI-1445176。

参考

[1] Borkiewicz, K., Christensen, A. J., and Stone, J. E. Communicating

Science Through Visualization in an Age of Alternative Facts. In ACM SIG

GRAPH Courses (2017), pp. 8:1–8:204.

[2] Bourke, P. Synthetic Stereoscopic Panoramic Images. In Interactive Tech

nologies and Sociotechnical Systems, H. Zha, Z. Pan, H. Thwaites, A. Addison,

and M. Forte, Eds., vol. 4270 of Lecture Notes in Computer Science. Springer,

2006, pp. 147–155.

[3] Greene, N., and Heckbert, P. S. Creating Raster Omnimax Images from

Multiple Perspective Views Using the Elliptical Weighted Average Filter. IEEE

Computer Graphics and Applications 6, 6 (June 1986), 21–27.

[4] Humphrey, W., Dalke, A., and Schulten, K. VMD—Visual Molecular

Dynamics. Journal of Molecular Graphics 14, 1 (1996), 33–38.

[5] Sener, M., Stone, J. E., Barragan, A., Singharoy, A., Teo, I., Van

divort, K. L., Isralewitz, B., Liu, B., Goh, B. C., Phillips, J. C.,

Kourkoutis, L. F., Hunter, C. N., and Schulten, K. Visualization of

Energy Conversion Processes in a Light Harvesting Organelle at Atomic De

tail. In International Conference on High Performance Computing, Networking,

Storage and Analysis (2014).

[6] Simon, A., Smith, R. C., and Pawlicki, R. R. Omnistereo for Panoramic

Virtual Environment Display Systems. In IEEE Virtual Reality (March 2004),

pp. 67–73.

**47**

[7] Stone, J. E., Sener, M., Vandivort, K. L., Barragan, A., Singharoy,

A., Teo, I., Ribeiro, J. V., Isralewitz, B., Liu, B., Goh, B. C., Phillips,

J. C., MacGregor-Chatwin, C., Johnson, M. P., Kourkoutis, L. F.,

Hunter, C. N., and Schulten, K. Atomic Detail Visualization of Photosyn

thetic Membranes with GPU-Accelerated Ray Tracing. Parallel Computing 55

(2016), 17–27.

[8] Stone, J. E., Sherman, W. R., and Schulten, K. Immersive Molecular

Visualization with Omnidirectional Stereoscopic Ray Tracing and Remote Ren

dering. In IEEE International Parallel and Distributed Processing Symposium

Workshop (2016), pp. 1048–1057.

**48**