动态裸眼 3D 数字全息显示

参赛者：黄隆钤，刘新梓，孔志彬

（浙江大学理学院，杭州 310058）

指导老师：陈星

摘要

全息，即全部的信息。全息照相技术所使用的光学干板，可以记录光的振幅和相位，它记录了光的全部信息，因而可以对物体进行再现。

从麦克斯韦方程组可以导出的亥姆霍兹方程，求解可得角谱衍射积分，对其进行傍轴近似，就得到了菲涅尔衍射积分。利用菲涅尔衍射积分，我们可以描述光场的傍轴传播过程。

我们利用计算机，模拟全息照相的过程。对一个目标三维物体，可以利用菲涅尔衍射积分的数值算法计算其透射光传播到空间光调制器上的波面情况，提取相位生成相位全息图。加载到空间光调制器上，用参考平行光进行照射，即可在观察平面看到三维物体。

对运动的三维物体，顺序播放每一帧的相息图，可以实现动态裸眼 3D 数字全息显示。

我们对动态裸眼 3D 数字全息的显示进行了深入研究，实现了效果良好的 3D 全息视频显示。根据物体的三维旋转矩阵，可以实现 3D 物体经自由旋转后的相息图。我们编写了基于 MATLAB 的 3D 全息 App “HelloHolo”，可以让使用者可视化地进行 3D 物体任意角度相息图的制作，极大地提高了 3D 及 3D 视频相息图的制作效率。

关键字 全息照相；菲涅尔衍射积分；动态 3D 全息显示；3D 全息 App；相息图的空间旋转；、

---

1 研究背景

1.1 现有的 3D 全息研究

三维物体空间结构复杂，故难以用具体数学函数描述其物光波的分布。而且三维信息量大、全息计算繁杂，因此如何压缩三维全息计算的数据量、实现大视角全视差的显示已成为计算全息的难点。

就目前而言，三维物体相息图的获得主要包含三种方法：物点散射法、体视全息法、层析法。现对各方法简要介绍如下：

- 物点散射法：将三维物体中每个物点看作一个点光源，计算机模拟点光源在有限方向上的光学传播以获得该点在全息面上的复振幅分布，叠加所有物点的复振幅即可得到物体的波面在全息面上的信息分布，通过编码即可获得相息图。
- 体视全息法：用计算机或相机获得三维物体的多个视角的二维投影像，并对它们进行编码处理可得三维物体的计算全息图。我们了解到目前有两种实现方式，一种是对每张投影像加入倾斜因子并进行积分计算，将其积分制作为该视角投影图中对应点的像素值，组合一起得到全息图;另一种是将二维投影像分别进行傅里叶变换成二维频谱矩阵，将所有二维矩阵组成一个三维矩阵，通过分析光场分布函数，能够从中提取一个能表示物体三维特征的二维矩阵，对其进行编码即可得三维物体的相息图。
- 层析法：将 3D 物场沿深度方向分层，将各层在全息面的菲涅尔复振幅叠加后制作成相息图。

1.2 3D 全息的应用

3D 光显示能够更真实地重现或模拟与客观世界相同的场景，增强表达图像的深度感、层次感和真实性。我们研究的 3D 全息技术，拥有非常广阔的应用前景：

(1) 3D 影院。如今，佩戴 3D 眼镜的 3D 影院已经非常普及，3D 全息技术则会为裸眼 3D 观影等技术打开广阔市场。我们制作了简单的 3D 全息视频，我们相信这将是未来电影的 3D 裸眼电影的一个缩影。

(2) 医学教学，全方位地展示手术过程，医生可以观摩手术操作，学习手术技术 还有应用于医疗过程中，将人体信息变得三维可视化 利于确定治疗方案

(3) 体育教学。通过建立运动模型来全方位示范动作要领和产生的动作效果 或者用于体育训练 运动员分析动作特点 纠正自身动作

(4) 布局展示。项目经过完善和拓展，还可以应用于体育比赛的战术布局、楼市楼盘的立体显示等。三维全息影像将为教练提供一个动态、立体的战术布局板，为楼市楼盘提供一个长期无损耗、节省资金的新的展示方式。

2 3D 动态全息显示的原理

2.1 全息照相

物体发出的物光波与参考光波在全息干板上发生干涉，干涉后的强度分布为：

$$\begin{aligned} I(x,y) &= \left|\widetilde{E}_o(x,y)+\widetilde{E}_r(x,y)\right|^2 \\ &= \left|\widetilde{E}_o\right|^2+\left|\widetilde{E}_r\right|^2+\widetilde{E}_r(x,y)\widetilde{E}_o^*(x,y)+\widetilde{E}_o(x,y)\widetilde{E}_r^*(x,y) \end{aligned}$$

全息干板/全息图相当于一个振幅投射系数 $t(x,y)=k_0+k_1I(x,y)$ 的衍射屏，参考光再次照射时，透过全息图的光波的复振幅分布中，第一、二项为零级衍射光，第三项为再现物光波，其信息与原物光波完全相同，为 +1 级衍射光、发散波，是虚像；最后一项为 -1 级衍射光，是会聚波、原物光波的共轭波。

由于第三项为原物光波的缘故，参考光再次照射后，可以看到复现的物体。

2.2 菲涅尔衍射积分

光场满足麦克斯韦方程，进而满足波动方程

$$\nabla^2\vec{E}-\frac{1}{c^2}\frac{\partial^2\vec{E}}{\partial t^2}=0$$

假设解为一般形式为 $U(\vec{r},t)=U(\vec{r})e^{-2i\pi\nu t}$ ，带入可以得到三类解，第一类为角谱衍射积分，第二、三类为瑞丽-索末菲积分，我们不进行研究。对角谱衍射积分中的根号进行一阶近似（傍轴近似），就可以得到菲涅尔衍射积分：

$$\begin{aligned} U(x,y,z_0) &= \frac{\exp(ikz_0)}{i\lambda z_0}\iint\widetilde{E}(x_0,y_0)\exp\left[\frac{ik}{2z_0}[(x-x_0)^2+(y-y_0)^2]\right]\mathrm{d}x_0\mathrm{d}y_0 \\ &= \frac{\exp(ikz_0)}{i\lambda z_0}\exp\left[\frac{i\pi}{\lambda z_0}(x^2+y^2)\right]\times\iint U(x_0,y_0,0)\exp\left[\frac{i\pi}{\lambda z_0}(x_0^2+y_0^2)\right]\exp\left[-i2\pi\left(\frac{x}{\lambda z_0}x_0+\frac{y}{\lambda z_0}y_0\right)\right]\mathrm{d}x_0\mathrm{d}y_0 \\ &= \frac{\exp(ikz_0)}{i\lambda z_0}\exp\left[\frac{i\pi}{\lambda z_0}(x^2+y^2)\right]\times\mathrm{FFT}\left\{U(x_0,y_0,0)\exp\left[\frac{i\pi}{\lambda z_0}(x_0^2+y_0^2)\right]\right\} \end{aligned}$$

菲涅尔衍射原理告诉我们，从光从物面 $U_0(x_0,y_{0,}0)$ 传播到像面 $U(x,y,z_0)$ ，像面上每一点都是物面所有点透射的光的相干叠加。这个积分的最后一步就是数值算法，在程序中为 s_fft.m 函数。

2.3 层析法

我们采用层析法，对三维点云物体按照深度进行分层切片，分别计算每一层的衍射积分，再进行叠加。例如，对一个三维兔子进行 10 层的切片，前九张的切片效果如下图所示：

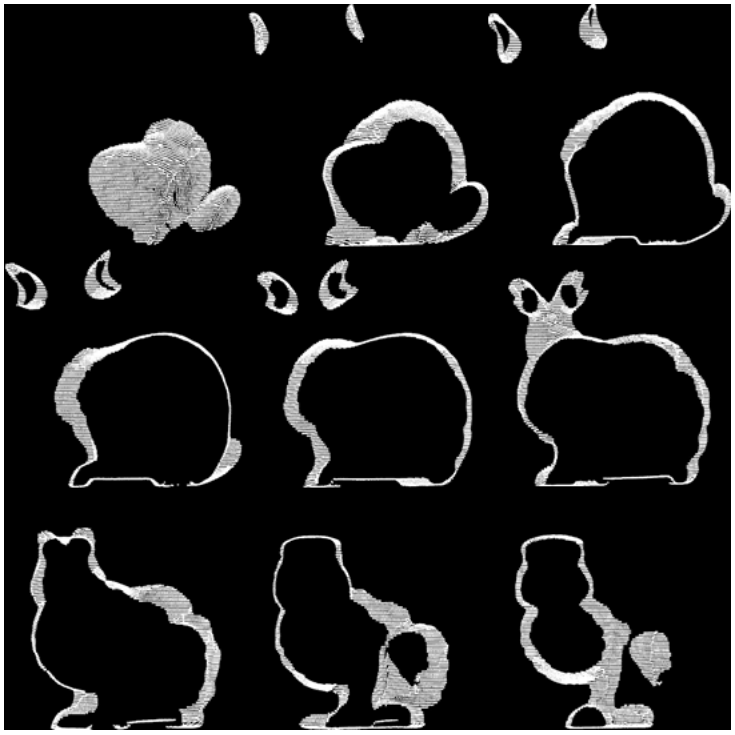


图 1 切片效果

2.4 全息照相的数值模拟

我们利用菲涅尔衍射积分模拟目标物体发出的光到全息干板的传播，采用层析法对三维物体的每一个点进行衍射积分，利用 S-FFT 算法（请见附录）得到物光传播一定距离的复振幅分布，叠加后提取相位得到相息图。

相息图对应着全息干板，加载到空间光调制器上，参考平行光再次照射，将在设定的观察平面呈现出三维物体的全息像。

2.4 随机相位与 GS 迭代算法

随机相位方法是在提取相位之前，在像平面上叠加随机噪声相位，来模拟毛玻璃的漫反射，令所有空间频率的光波分量均匀地传播到 SLM 平面上。若原图未添加随机相位，衍射积分后抛弃振幅只取出相位，会造成图像信息的丢失，无法复现原图。预先叠加随机相位，SLM 平面上的振幅相对均匀，如漫射物体的光波场，此时相位是主要信息，只提取相位便能够较好地还原图像。

GS 迭代算法最早由 Gerchberg[14]等在 1972 年提出，开创了相位恢复技术应用的基础。随机相位结合 GS 迭代，可以把振幅信息通过迭代编码进相位中，大幅度提高相息图还原质量，流程如下：

(1) 获取物平面的信息 A_i（目标图像振幅信息），对物平面乘以随机相位 $\exp(\mathrm{j}\varphi_i)$；

(2) 利用菲涅尔衍射积分，计算到达 SLM 平面的衍射场分布；

(3) 对(2)的结果取 $A_p = 1$，采用菲涅尔衍射积分逆变换计算到达物平面的复振幅分布，表示为 $A_i' \exp(\mathrm{j}\varphi_i')$ 若物点 A 与像点的 A' 的振幅相近，则输出灰度级的相位全息图。否则，令 $\varphi_i = \varphi_i'$ 进行新一轮的迭代计算。重复上述过程，直到获得的光波场的复振幅达到期望值，输出相应的结果。

我们在程序中设定了 GS 迭代的次数参数，使用者可以自由设定迭代次数。

2.5 3D 相息图的空间旋转

对三维物体的坐标矩阵乘以旋转矩阵，就可以实现三维物体的旋转。

对三维物体的坐标矩阵 M，记

$$R_x(\alpha)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\alpha & -\sin\alpha\\ 0 & \sin\alpha & \cos\alpha\end{bmatrix}$$

$$R_y(\beta)=\begin{bmatrix}\cos\beta & 0 & \sin\beta\\ 0 & 1 & 0\\ -\sin\beta & 0 & \cos\beta\end{bmatrix}$$

$$R_z(\gamma)=\begin{bmatrix}\cos\gamma & -\sin\gamma & 0\\ \sin\gamma & \cos\gamma & 0\\ 0 & 0 & 1\end{bmatrix}$$

若将模型绕 x 轴旋转角度 α，绕 y 轴旋转 β，绕 z 轴旋转 γ 度，则旋转后的模型为（公式中·为矩阵乘法）

$$M' = M \cdot R_x(\alpha) \cdot R_y(\beta) \cdot R_z(\gamma).$$

对模型进行三维旋转，再进行相息图计算，就可以得到再现像为旋转后的物体的相息图。编写为程序中分别为 rotx.m、roty.m、rotz.m。

2.5 3D 全息显示光路

下图为实验所用光路图，为了提升显示效果，我们加入了一个反射镜以再有限空间内增大光程。

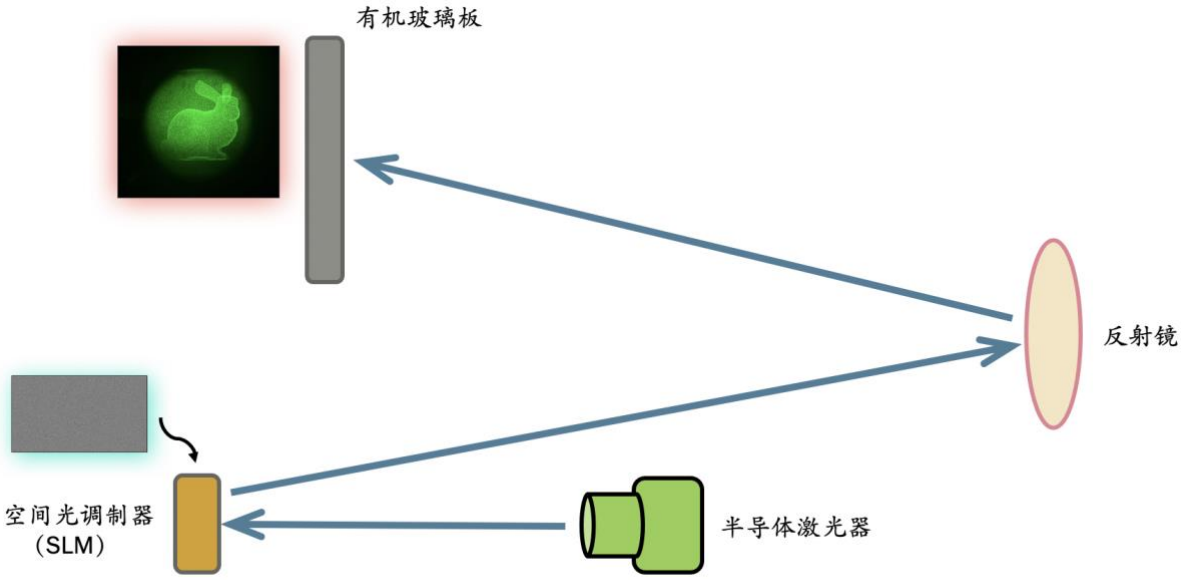


图 2 3D 全息显示光路示意图

2.6 3D 物体的动态显示

SLM 支持 10 帧每秒的动态响应，因此可用来实现 3D 视频的全息投影。即 3D 物体的动态显示。

3 实验操作与技术分析

3.1 HelloHolo App 的使用

从 Matlab 打开 app，可以看到下图的主界面。接下来可以从文件导入三维点云数据，支持 .xyz,.txt,.ply 格式。导入后，可以在左下角窗口看到所导入的三维物体。

App 主界面如下图所示：

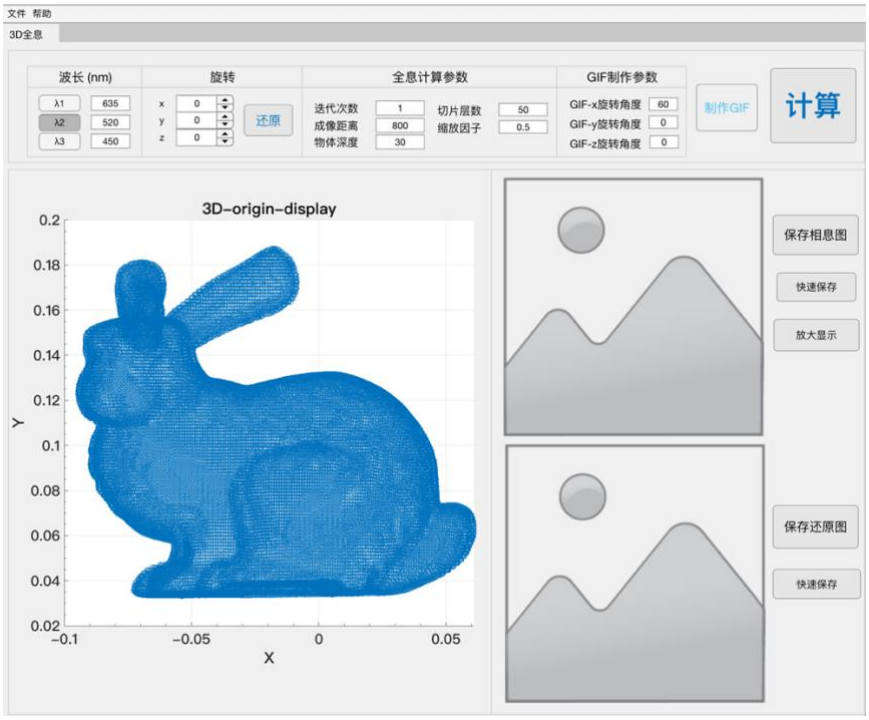


图 3 HelloHolo App 主界面

界面参数介绍如下：

旋转用于调整原模型的角度，如 x：45 表示物体绕本体坐标系 x 轴旋转 45 度。旋转后点击计算，将计算模型旋转后的相息图，即实现了“相息图的空间旋转。

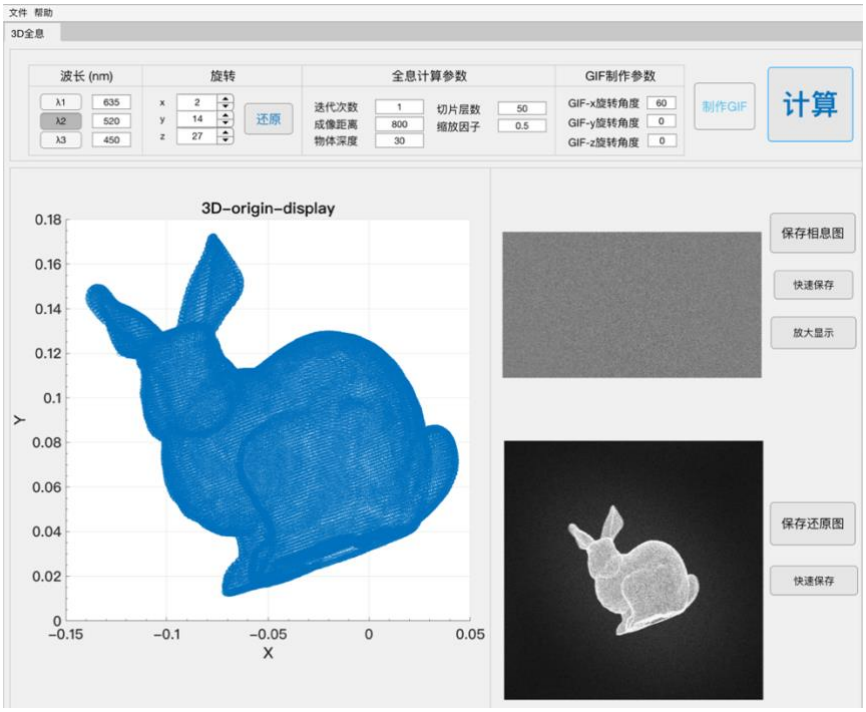


图 4 旋转效果

3D-origin-display：显示原三维物体及旋转后的效果。该界面用于辅助相息图的旋转，它和相息图的模拟还原图相对应。使用者设置旋转参数后，该界面会实时变化，可以根据设想的三维物体角度进行可视化地设置，方便了相息图的生成。

波长为光源的波长：我们提供了默认的 635nm、520nm、450nm（红光、绿光、蓝光）的波长，使用者可以自行修改。

全息计算参数中的**迭代次数**为对三维物体切片处理过程中所用 G-S 算法[14] 的迭代次数。**切片层数**控制对三维物体的切片数量。**成像距离**为光路中的衍射距离，**物体深度**为实际物体的横向深度，**缩放因子**控制相息图还原后的大小。

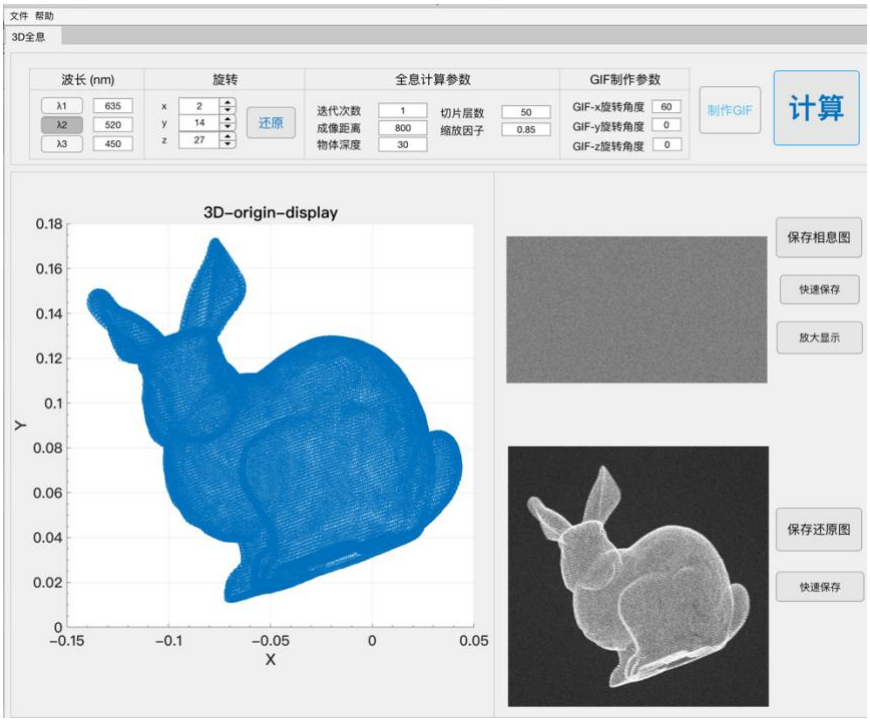


图 5 缩放效果

使用方法简介如下，使用者也可以查看帮助来学习操作步骤。

1. 单击文件，找到导入选项，选择物体的三维数据文件（默认导入了 bunny. txt 文件）。

2. 设置光源波长、旋转角度等参数，点击计算。

3. 等待相息图制作，完成后能看到相息图和模拟还原的效果图。

4. 单击保存相息图，自定义保存目录；也可以单击保存至当前文件夹，默认保存在当前程序目录。

3.2 3D 全息光路搭建

我们按照下图所示进行了光路的搭建，为了提升显示效果，我们放置了反光镜以增大光程，由于衍射角的存在，随着光程的增加，再现像会变大。

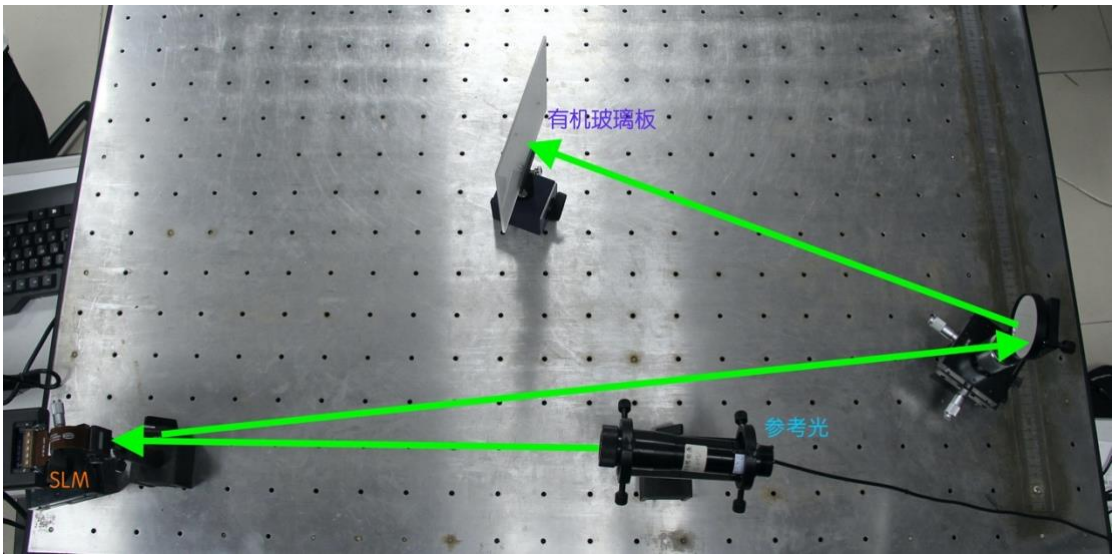


图 6 实验光路图

3.3 3D 全息显示效果

将 HelloHolo 制作出的相息图加载到 SLM 上，用参考平行光进行照射，可以得到如下的显示效果：

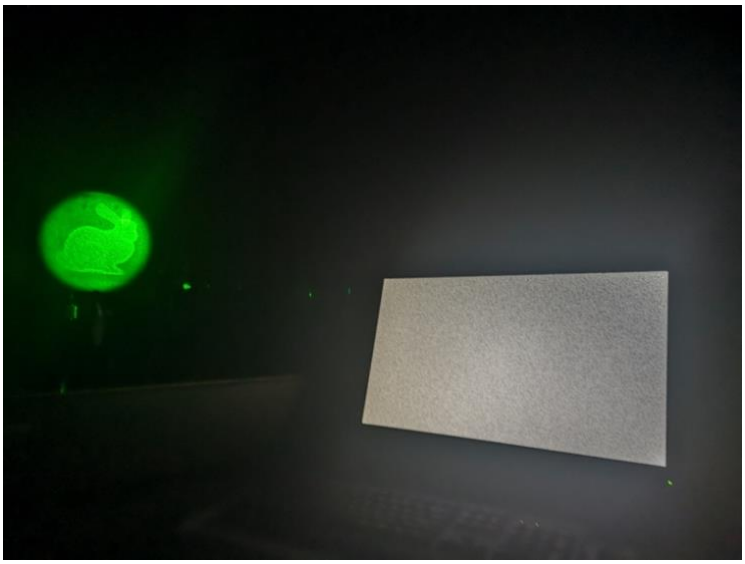


图 7 相息图与显示像

事实上，相息图不止可以显示一个三维物体，多个也是可以实现的。比如我们制作的两个小黄人的模型：

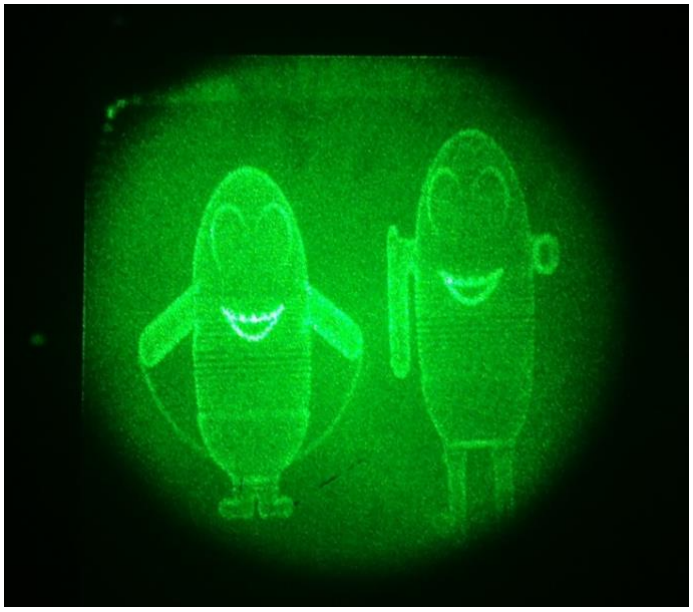


图 8 多人 3D 全息显示

为了展现 3D 物体的空间分布，我们制作了特殊的模型。模型中的二个小黄人横向错开，而纵向一前一后。当制作出相息图并用参考光进行照射时，可以看到在不同位置的显示中，前面和后面小黄人的清晰度是不一样的：

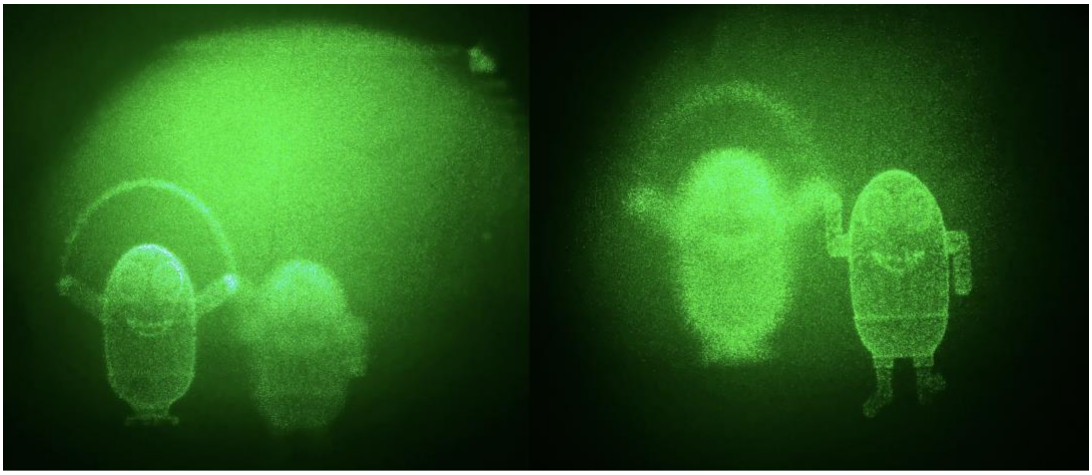


图 9 不同距离处一前一后的小黄人

3.4 3D 动态全息显示效果

我们通过 SolidWorks 软件，制作了简单的视频。由于视频由不同的帧组成，可以利用人体的视觉暂留效应实现动态显示。（见下页）

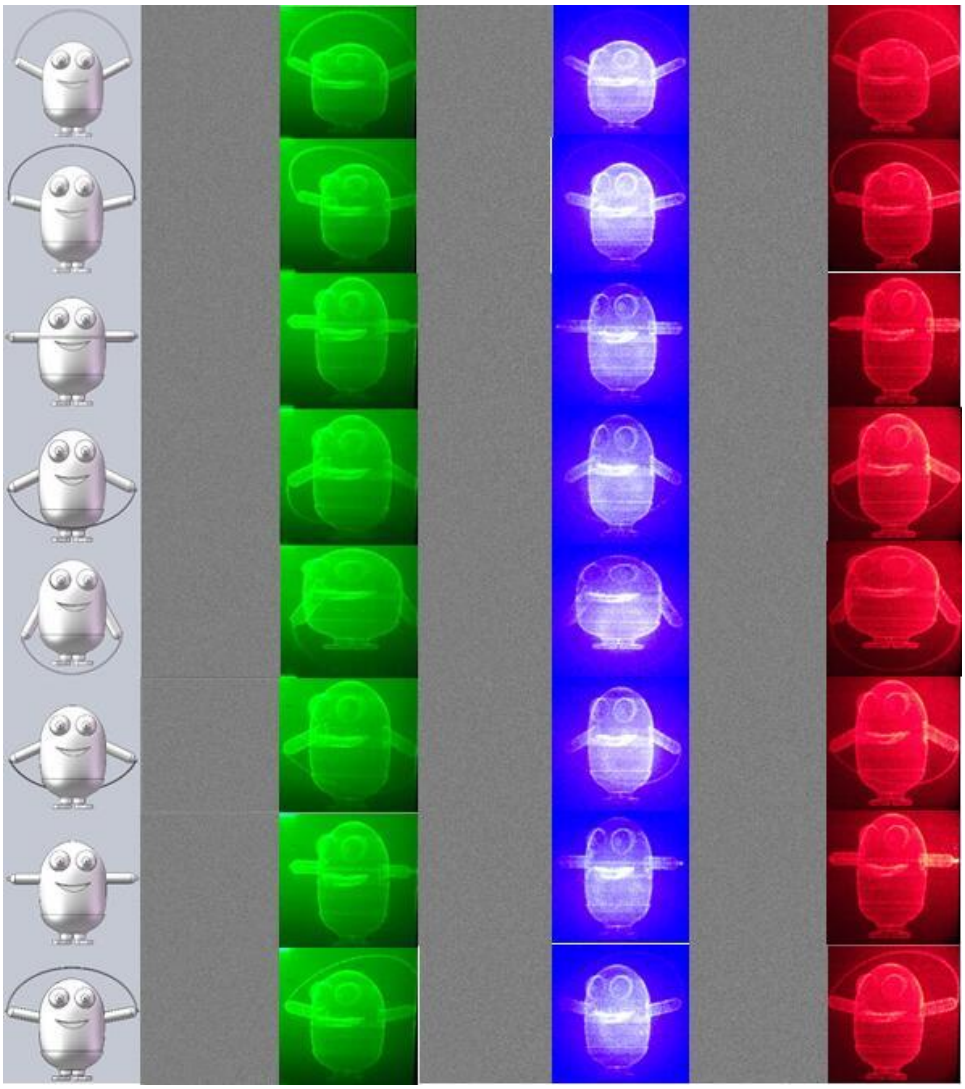


图 10 3D 动态全息显示效果图（模型—相息图—再现像—相息图—再现象—相息图—再现象）

如上图，我们设计了一个小黄人跳绳的模型，通过构建八个模型---模型中小黄人对应的手臂角度和绳子所在的位置不同，将这些模型分别生成 STL 文件,用 Meshlab 进行处理后（预处理过程请见附录）再用我们编写的 HelloHolo App 制作相息图，将相息图按一定频率进行播放，可以实现 3D 物体的动态全息显示。

我们进行了红绿蓝三个通道的实验，并将模型、相息图、成像结果整合到了上图中。

4 应用前景与未来展望

1992 年中国改革开放转型后，正式向全面建设小康社会转型。建设小康社会是改革开放战略之一。所谓全面的小康社会，不仅仅是解决温饱问题，而是要从政治、经济、文化、社会、生态等各方面满足城乡发展需要。3D 光显示能够更真实地重现或模拟与客观世界相同的场景，增强表达图像的深度感、层次感和真实性。我们研究的 3D 全息技术，在建设小康社会的过程中，拥有非常广阔的应用前景：

1.娱乐活动。如今，佩戴 3D 眼镜的 3D 影院已经非常普及，3D 全息技术则会为裸眼 3D 观影等技术打开广阔市场。我们制作了简单的 3D 全息视频，我们相信这将是未来电影的 3D 裸眼电影的一个缩影。3D 全息技术同样可以应用于二次元虚拟歌手的演唱会舞台立体成像。

2.医学教学。利用 3D 全息技术，能够全方位地展示手术过程，医生可以观摩手术操作来学习手术技术；也可以应用于医疗过程中，将人体信息变得三维可视化，利于确定对患者的治疗方案。

3.体育运动。应用 3D 全息技术，可以通过建立运动模型来全方位示范动作要领和产生的动作效果，或是用于体育训练时运动员分析动作特点来纠正自身动作。而在体育比赛里，可以通过多个机位获取运动员的点云数据。可以在运动员名次出现争议时，进行三位全息投影进行再现，也可以在判断运动员动作是否标准时使用。

4.显示各种布局。项目经过完善和拓展，还可以应用于体育比赛的战术布局、楼市楼盘的立体显示等。三维全息影像将为教练提供一个动态、立体的战术布局板，为楼市楼盘提供一个长期无损耗、节省资金的新的展示方式。

我们思考了 3D 全息的未来，随着人工智能的普及，深度学习技术和 5G 将与 3D 全息无缝衔接。可以利用神经网络替换掉迭代算法以提高计算速度，还可以结合 5G 将其部署与移动端，实现 3D 全息的移动应用，这将有非常大的应用潜力。

全息的发展将为人类带来的科技进步，可以应用于社会的方方面面，丰富人们的文化生活和娱乐生活，推动小康社会的全面实现。

参考文献

[1] Optical Holography: Materials, Theory and Applications[M]. Elsevier, 2019.

[2] Gabor D. Microscopy by reconstructed wave-fronts[J]. Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences, 1949, 197(1051): 454-487.

[3] 贾甲,王涌天,刘娟,etal.计算全息三维实时显示的研究进展[J].激光与光电子学进展，2012(05):16-24. [4] 吕乃光. 傅里叶光学.第 3 版[M]. 机械工业出版社，2016.

[5] 马建设,夏飞鹏,苏萍,etal. 数字全息三维显示关键技术与系统综述[J].光学精密工程，2012,20(5).

[6] 丁华锋,王卓,刘婧芳,etal. 一种基于 MATLAB 的 STL 文件分层切片算法[J].机床与液压,2018,046(005):102-105.

[7] 裴闯,蒋晓瑜,王加,etal. 一种三维物体相息图的迭代计算方法[J].光子学报,2013,042(003):348-353.

[8] 林培秋. 基于纯相位型液晶空间光调制器的相息图三维显示的研究[D]. 浙江师范大学，2010.

[9] 郭苗苗. 计算全息生成和再现方法研究[D]. 西安电子科技大学，2014.

[10] Leseberg D, Frère C. Computer-generated holograms of 3-D objects composed of tilted planar segments. Appl Opt. 1988;27(14):3020–3024. [11] Benton S A, Bove Jr V M. Holographic imaging[M]. John Wiley and Sons, 2008.

[12] 杨上供,甘亮勤,熊飞兵. 基于 LC-SLM 的 CT 图像全息三维重构与实时显示[J]. 激光与红外，2014,44(12): 1360-1363.

[13] 杨上供,王辉,金洪震. CT 图像的计算机制全息三维重建[J]. 光电工程，2006，33(12): 23-26.

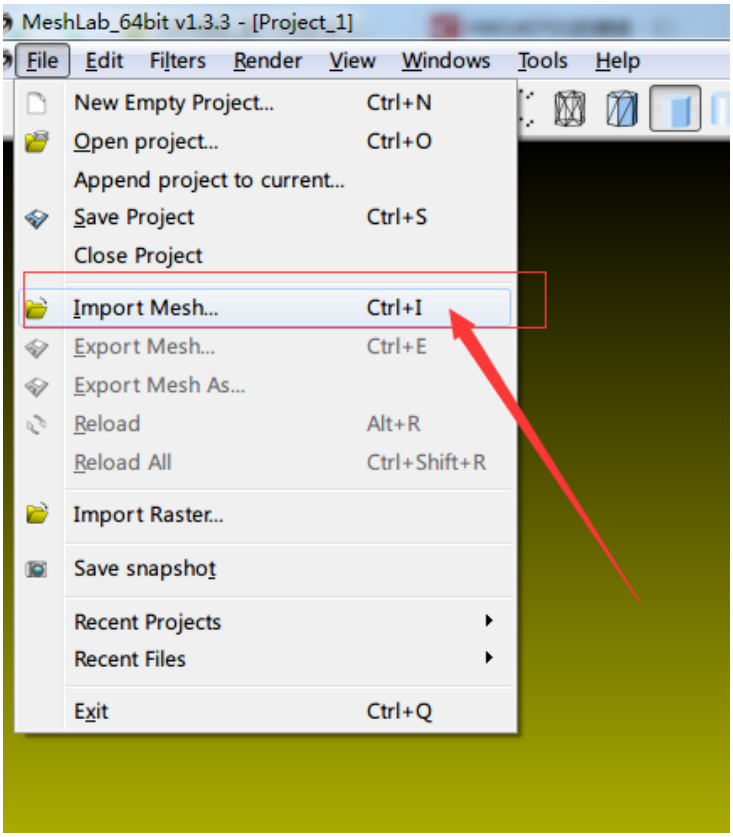
[14] Di Leonardo, Roberto, Francesca Ianni, and Giancarlo Ruocco. "Computer generation of optimal holograms for optical trap arrays." Optics Express 15.4 (2007): 1913-1922.

附　录

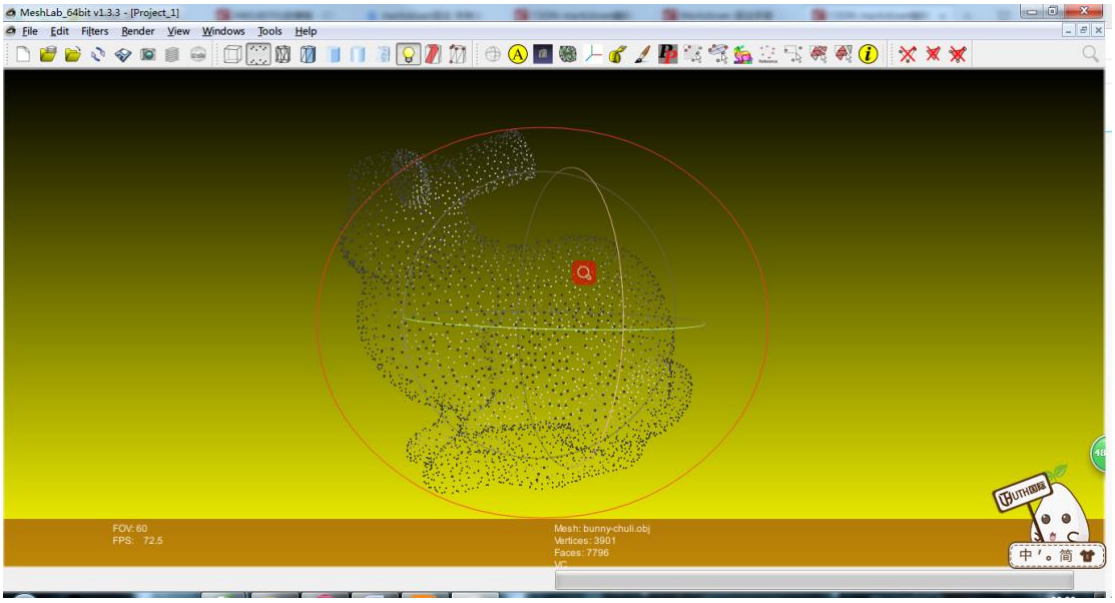
1. 利用 Meshlab 对 3D 数据进行预处理

（1）载入三维模型

点击 File-Import Mesh 可以载入各种格式的三维模型，包括 obj,ply,stl,off 等知名的三维模型文件，但是除了 dxf 的 3DFace 文件。

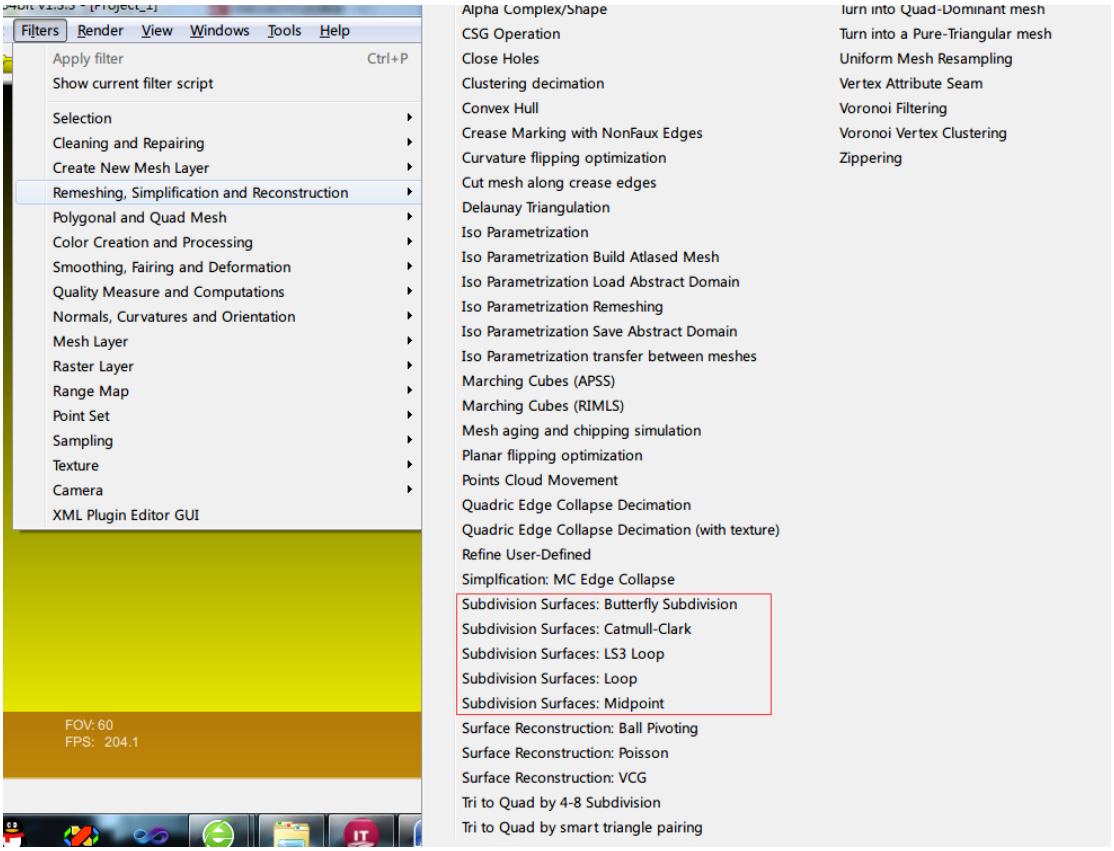


在这里以 bunny 即斯坦福兔子为例：



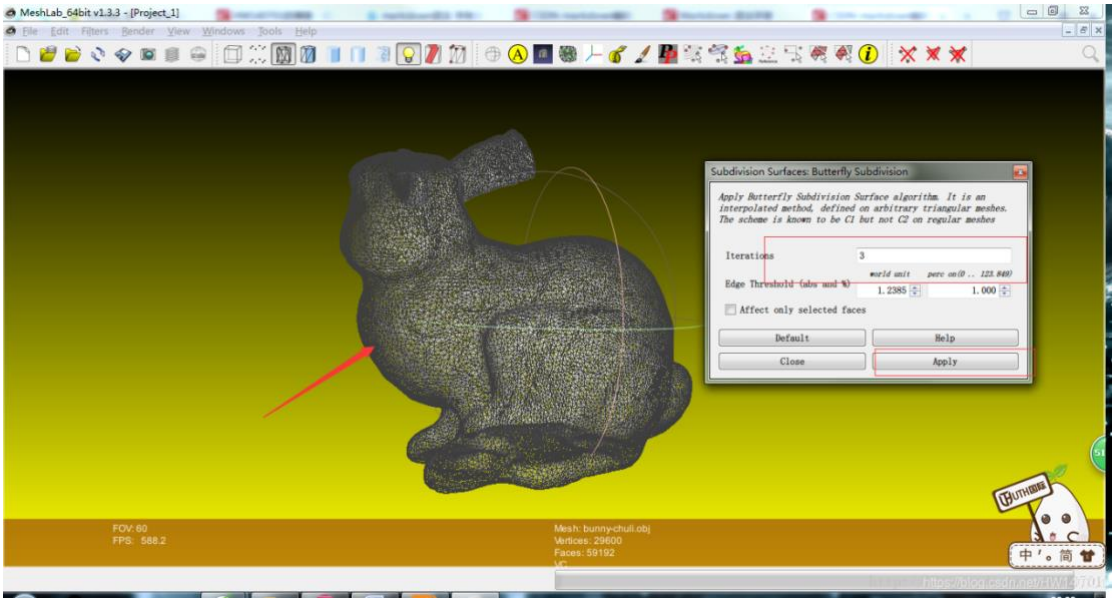
如果需要大量的三维点云数据集，那么可以对载入的三维网格模型进行**平滑加密**，由此可以获得更多的三维点；相反，如果机器内存不足，不足以处理大数据量的点云，可以选择对载入的三维网格模型进行简化，那么自然，点的数量也会跟着减少。

（2）网格的细分加密



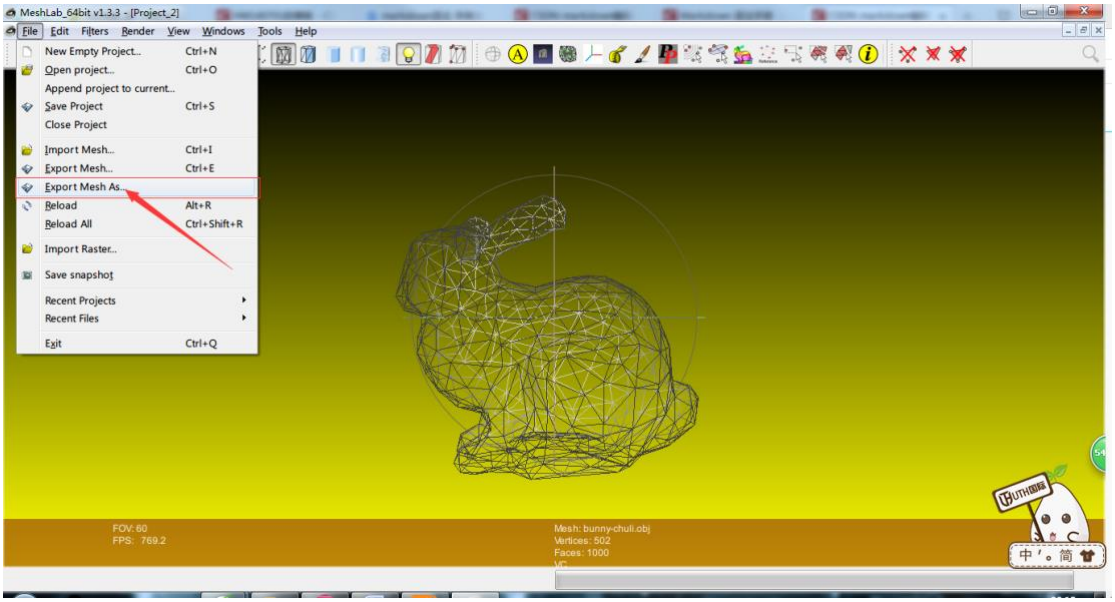
可以选择上图中的任意的算法进行网格细分，增加三角形，间接增加点的数量

下图，是以**蝴蝶细分算法**所做的网格细分

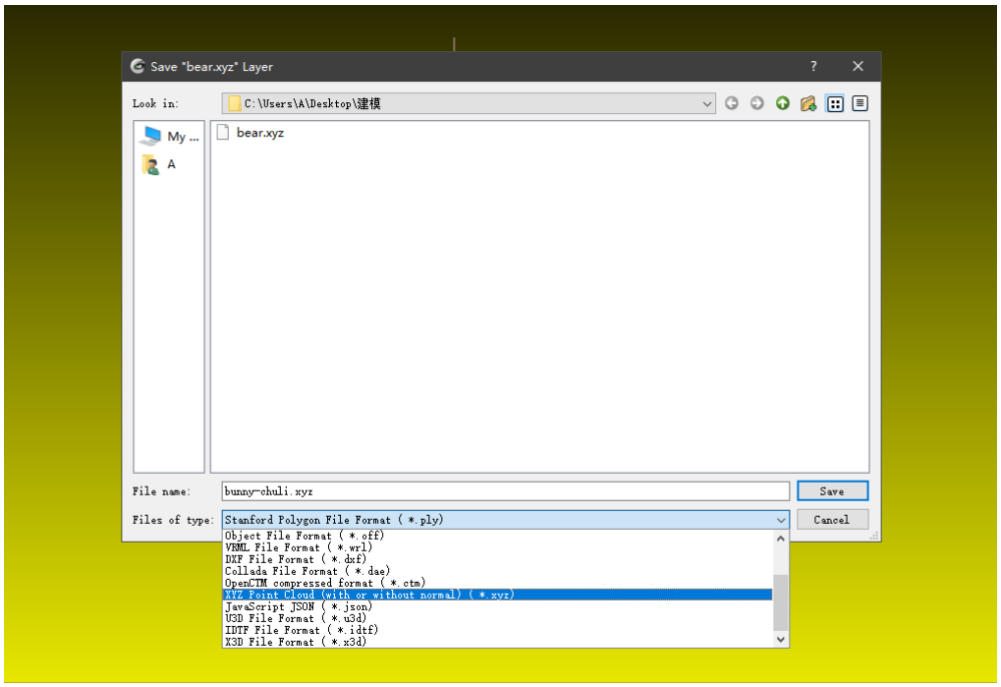


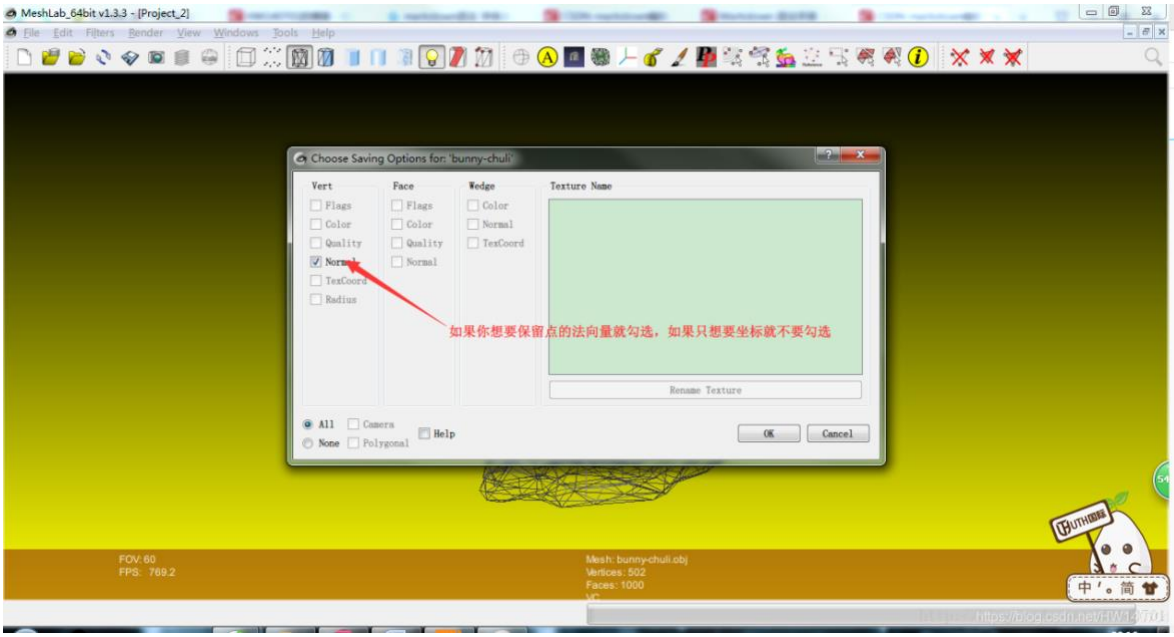
迭代次数默认为 3，数字越大点的数量越多。

（3）点云导出为*.xyz 规则排列的文本格式文件



选择导出格式为 xyz 并保存





导出的数据格式为*.xyz 后缀名，其实是文本文件格式，修改后缀名为.txt,就可以用记事本打开了。

2. 程序核心代码

（1）主程序代码：

```
function[pog,rcst_img]=calculate_3dgray(model,lambda,z0,depth,slices,iter_num,m0)
%    parameters
M = 1920; N = 1080; % slm resolution: horizontal and vertical pixels
% m0 = 0.5 ;    % image zoom factor: to prevent image superposition
pix=0.008;    % unit pixel width / mm
z=(1:slices)/slices*depth+z0; % different diffraction distance because of the depth of the
object
L=lambda.*z/pix; % length of the observation plane
%    cut pieces and get the slices
cutted_pieces=cut_pieces(model,slices);


%    resize the slices and add random phase on each slice
[U0,A0,xx0s,yy0s,xx,yy]=initialize(cutted_pieces,M,N,m0,L,pix);
f = uifigure;
d = uiprogressdlg(f,'Title','Calculating Phase Only Graph',...
            'Message','Please wait...','Cancelable','on');


 % GS iteration
for iter=1:iter_num
        if d.CancelRequested
```

```
            break
        end
        U_slm=zeros(N,M);
        for i=1:slices
            tmp=s_fft(U0{i},M,N,lambda,z(i),xx0s{i},yy0s{i},xx,yy);
            U_slm=U_slm+tmp; % complex applitudes superposition
        end

        phase=angle(U_slm)+pi; % takeout angle to get pahse graph, angle is range from -pi to pi
        for i=1:slices
            tmp=i_fft(exp(1i.*(phase-pi)),M,N,lambda,z(i),xx0s{i},yy0s{i},xx,yy);
            U0{i}=A0{i}.*exp(1i*(angle(tmp)));
        end

    d.Value=iter/iter_num;
    d.Message = sprintf("Process: %.2f/1.00",d.Value);
    disp(d.Value);


end
```

```
d.Message="Done.";

pause(0.2);

close(d);

close(f);


pog=uint8(phase/2/pi*255);

pog=cat(3,pog,pog,pog);

rcst_img=reconstruction(tmp,6);
```

（2）切片算法代码：

```
function cutted_pieces = cut_pieces(model, slices)


    % Cut pieces from 3-D objects.

    % INPUT: number of slices -- n

    % OUT: write images of slices into tmp file


    %sort by z coordinate

    cutted_pieces = cell(slices, 1);

    ver = sortrows(model, 3);

    X = ver(:, 1);

    Y = ver(:, 2);

    num = size(X, 1);
```

```
XMIN = min(X);

XMAX = max(X);

YMIN = min(Y);

YMAX = max(Y);


% adjust z coordinate

min_z = min(ver(:, 3));

ver(:, 3) = ver(:, 3) - min_z;


% get z-range of one slice per layer

ver1 = ver(1:ceil(num / slices):num, :); % ver1=ver(1:floor(num/slices):num,:);

z1 = ver1(:, 3);


figure;


for i = 1:length(z1)


    if i < length(z1)

        % add all the points whose z within the z-range

        xy = ver(ver(:, 3) > z1(i) & ver(:, 3) < z1(i + 1), 1:2);

    else % the last z-range

        xy = ver(ver(:, 3) > z1(i), 1:2);
```

```
            end
            plot(xy(:, 1), xy(:, 2), '.w');
            axis equal;
            axis([XMIN XMAX YMIN YMAX])
            axis off;
            set(gcf, 'Color', 'black');
            frame = getframe(gca);
            frame = frame2im(frame);
            frame = rgb2gray(frame);
            imwrite(frame, ['./' num2str(i) '.jpg']);

        end
        close all;

        for i = 1:slices
            cutted_pieces{i} = imread(['./' num2str(i) '.jpg']);
            delete(['./' num2str(i) '.jpg']);
        end
end
```

（3）旋转算法代码（部分）:

```
function model = roty(model, y)
```

```
    % clock-wise
    y = -y;
    Ry = [cos(y) 0 sin(y); 0 1 0; -sin(y) 0 cos(y)];
    model = model * Ry;
end
```

（4）S-FFT 和 I-FFT 代码：

```
function U = s_fft(U0, M, N, lambda, z, xx0, yy0, xx, yy)

    % Fresnel Diffraction Integral
    % Use single fft to calculate Fresnel diffraction integral
    % --------
    % INPUT
    % --------
    % M,N: the number of pixels along, the respective x and y direction of the slm
    % lambda: wavelength of the light
    % z:distance between the diffracted plane(slm) and the observation plane
    % xx0,yy0: points on the diffracted plane
    % xx,yy: points on the observation plane
    % U0: complex amplitudes on the diffraction screen; [ type: slices*1 cell ]
    % --------
    % OUTPUT
```

```
    % --------
    % U: complex amplitudes on the observation screen after s-fft

    k = 2 * pi / lambda;
    U = fftshift (fft2 (U0 .* (exp(1i * k / 2 / z * (xx0.^2 + yy0.^2))), N, M)); % add fftshift
    U = (exp(1i * k * z) / (1i * lambda * z) * exp(1i * k / 2 / z * (xx.^2 + yy.^2))) .* U;
end
```

```
function U0 = i_fft(U, M, N, lambda, z, xx0, yy0, xx, yy)

    % Invert of Fresnel Diffraction Integral
    % --------
    % INPUT
    % --------
    % M,N: the number of pixels along, the respective x and y direction of the slm
    % lambda: wavelength of the light
    % z:distance between the diffracted plane(slm) and the observation plane
    % xx0,yy0: points on the diffracted plane
    % xx,yy: points on the observation plane
    % U: complex amplitudes on the observation screen after Fresnel diffraction
    % integral
    % --------
    % OUTPUT
```

```
    % --------
    % U0: complex amplitudes on the diffraction screen

    k = 2 * pi / lambda;
    G = (1i * lambda * z) * exp(-1i * k * z) * exp(-1i * k / 2 / z * (xx.^2 + yy.^2));
    U0 = ifft2(U .* G, N, M) .* (exp(-1i * k / 2 / z * (xx0.^2 + yy0.^2)));

end
```

3．3D 多物体动态全息显示效果