基于弧形CAD Wall的裸眼3D技术—中期答辩报告

By 李佳奇，黄梓通，沈徐檑

12012710@mail.sustech.edu.cn
12012420@mail.sustech.edu.cn
12012428@mail.sustech.edu.cn

2022年11月30日

1. 背景介绍

1.1. 技术路线

本项目为一种裸眼 3D 显示系统，所述裸眼 3D 显示系统包括：位置指示装置和显示装置；所述位置指示装置设置于用户头部，提供用于指示用户双眼所在位置的指示信号；所述显示装置用于根据所述位置指示装置发出的指示信号确定所述用户双眼相对于所述显示装置的位置，并根据所述位置显示 3D 图像。通过位置指示装置指示出用户双眼所在位置，显示装置根据位置指示装置信号，计算出用户观看姿态和用户相对于显示装置的距离，从而在显示时实时自动调整显示内容，使屏幕内容能够自动适应用户的位置、角度和距离，极大的扩大了裸眼 3D有效观看区域，提高了整体观看效果。

1.2. 本项目的应用领域

本项目可以应用于家庭影院，个人游戏主机等追求沉浸感，针对个人用户，兼容弧形屏幕的场景，为个人用户提供良好的3D体验氛围

2. 项目改进内容

2.1. 技术路线调整内容

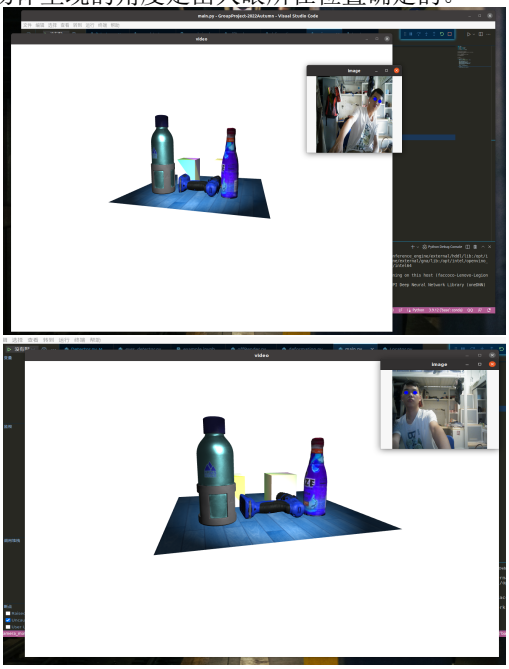
- 经过测试， 即使在不使用多线程优化的情况下，画面帧数仍然能够满足大部分的日常使用需求（30fps以上）[1]，因此决定不使用多线程优化。
- 画面形变方案调整， 增加了用户“假设屏幕”的存在。 由原先的等分弧形屏幕进行横向缩放， 改进为按照特定比例切分屏幕后再进行缩放，详细步骤见相应报告内容。

2.2. 中期答辩实现内容

- 完成pyrender实现的场景建模渲染demo。
- 完成各个模块的整合。
- 完成画面形变的优化方案。

3. 项目样例

本项目采用pyrender进行虚拟场景渲染， 可以看到，3D物体呈现的角度是由人眼所在位置确定的。[2]



4. 新实现及改进内容

4.1. pyrender画面渲染

4.1.1. 物体创建以及导入

```
bottle_gltf = trimesh('waterBottle.glb')
bottle_trimesh =
    bottle_gltf.geometry[\
```

1. 测试机CPU为Intel Core i7 10875H,GPU为NVIDIA Geforce RTX 2060 Notebook。

2. 书面材料不便于播放视频，如有需要欢迎通过邮箱联系我们

```
        list(bottle_gltf.geometry.keys())[0]]
    bottle_mesh = Mesh.from_trimesh\
        (bottle_trimesh)
```

4.1.2. 场景创建与添加物体

```
    scene = Scene(ambient_light =\
        np.array([0.02, 0.02, 0.02, 1.0]))
    scene.add(bottle_mesh, pose=bottle_pose)
```

4.1.3. 摄像头位姿矩阵获取

利用开题报告提到实现的Locator模块（即将人眼位姿的图像坐标转换为世界坐标系的模块），获取世界坐标系坐标

```
    def getLocation(position_x, position_y,
                    position_depth):
        x, y, z = Locator(position_x,
            position_y, position_depth)
        return x, y, z
```

利用pyrr.matrix44中的create_look_at()函数获取摄像头的位姿矩阵。

```
    def createCamPos(x=0.5, y=0, z=0.4):
        #create_look_at坐标系与Locator坐
        #标系不相同
        y_d, x_d, z_d = x, y, z
        x_d = 0 - x_d
        cam_pose = pyrr.matrix44.\
            create_look_at((y_d, x_d, z_d),\
            (0, 0, 0), (0, 0, 1))
        cam_pose = np.linalg.inv(cam_pose.T)
        return cam_pose
```

4.1.4. 按照摄像头位姿矩阵创建及更新camera对象位置

```
    while True:
        ...
        scene.set_pose(node=cam_node,
            pose = getCamPosByCap(position_x,\
            position_y, position_depth))
        ...
```

4.2. 画面形变的优化

根据用户在空间中三维空间坐标不同，利用“视野n分法”计算画面形变效果。本技术中仅考虑画面的横向形变，故以下均为俯视视角（该方 法示意图如图 3 所示）。定义曲面屏的中心为坐标原点$O(0,0)$，以此得到屏幕左 侧A点与屏幕右侧B点坐标，曲屏的半径r,曲屏的圆心角β, 用户眼睛所在的二维 坐标系坐标$N(a,b)$，画面形变的等分数量n（在一定范围内，参数n越大，视觉 效果越好）。当前用户的视野角度α可由如下方法计算：

将视野角α分为 n 等份，定义角等分线与屏幕的交点为,$a_1,a_2,...,a_n$ (点 A 为 a_0，点 B 为a_n)。$a_1,a_2,...,a_n$的坐标计算方法如下,a_i的坐标记为 (x_i,y_i)：

$$y_i-b=k_i(x_i-a) \tag{1}$$

$$x_i^2+(y_i+r)^2=r^2 \tag{2}$$

由（1）（2）可得：

$$y_i=\frac{|k_i|m-ak_i+b-k_i}{k_i^2+1} \tag{3}$$

其中，$m=\sqrt{-a^2k_i^2+2abk_i+2ak_ir-b^2-2br+k^2r^2}$

$$x_i=a+\frac{y_i-b}{k_i} \tag{4}$$

其中k_i为直线l_{Na_i}的斜率，计算方法如下：

$$k_i=\tan\left(-i\frac{a}{n}+\arctan\left(\frac{q}{p}\right)\right) \tag{5}$$

得到所有的,$a_1,a_2,...,a_n$坐标后，我们便可以算出被分割的每一段的弧长：

$$\widehat{a_ia_{i+1}}=\left(\arctan\frac{y_i+r}{x_i}-\arctan\frac{y_{i+1}+r}{x_{i+1}}\right)r \tag{6}$$

由此可以得出画面n等分中每一等分对应的屏幕的弧长。每段弧长与n分之一的 AB 弧长之比，定义该比例为r_n，则有

$$r_n=\frac{n\widehat{a_ia_{i+1}}}{\widehat{AB}} \tag{7}$$

其中

$$\widehat{AB}=\beta r \tag{8}$$

得到所有的r_n之后，我们用相同的方法计算每一等份对应的“假设画面” 的线段长度。并计算每段长度占总长的比例。这里我们假设一个垂直于人物视线方向的平面屏幕为“假设画面”。

我们可以通过一下公式获得角平分线与“假设画面”的交点：

$$y_i-b=k_i(x_i-a) \tag{9}$$

$$y_i=k'x_i \tag{10}$$

得到：

$$x_i=\frac{b-k_ia}{k'-k_i} \tag{11}$$

$$y_i=k'x_i \tag{12}$$

其中， k'是“假设屏幕”的斜率， 它垂直于人物的视线。即：

$$k'=\frac{-1}{\tan\left(-\frac{\alpha}{2}+\arctan\left(\frac{q}{p}\right)\right)}$$

此可以得出画面n等分中每一等分对应的“假设画面”的段长。每段长度与n分之一的“假设画面”总长之比，即为该段分区的形变比例，定义该比例为R_n, 则有

$$R_n = \frac{x_{i+1} - x_i}{x_n - x_0}$$

我们将虚拟摄像头传回的三维渲染画面称为原始图像，具体画面形变的操作是将原始图像按照R_n的比例分为n份，然后将这n份按照$\frac{r_n}{R_n}$的比例进行放缩后 拼接成输出的形变图像。