几何计算前沿第一次作业报告
TSDF Fusion 的实现

沈千帆 2200013220

2024 年 3 月 21 日

1 简述

TSDF Fushion 是 Kinect Fusion 算法的基础。在本次 Lab 中主要可以分为这几个部分：

- `demo.py`

  这个文件实现整个算法流程框架。进行文件的读取。读取相机位姿和深度图后进行坐标变换，确定空间的大小，初始化 TSDF 场。接着依次读取每张图片的深度图和颜色图，在坐标转换后融合 TSDF。最后对于融合好的 TSDF 场输出 mesh。

- `fushion.py`

  定义了 tsdf 类，主要负责初始化 tsdf 场；每次读取新的图片之后进行 tsdf 融合；返回最终的 mesh；相机坐标到世界坐标的转换函数。

- data

  包含相机内参，相机位姿，深度图，颜色图和重力方向。

2 具体实现

2.1 从深度图生成点云

根据 readme 提供的公式

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = z \begin{bmatrix} 1/f_x & 0 & c_x/f_x \\ 0 & 1/f_y & -c_y/f_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

其中 z 可以由深度图得到，$x = z \cdot (\frac{1}{f_x}u + \frac{c_x}{f_x}), y = \cdot(\frac{1}{f_y}v + \frac{c_y}{f_y})$，其中 c_x, x_y, f_x, f_y 分别为相机内参，这样就得到了相机坐标下的点的坐标。接着转换成齐次坐标，并和每张图片的相机位姿矩阵相乘得到世界坐标系下的点的坐标，再转换回齐次。最后输出在`pointcloud.ply`中，如图 1.

根据得到的点云坐标可以确定整个空间的实际范围，从而确定体素整数坐标，初始化 TSDF 场。

2.2 从深度图采样

从vox_coords->wor_pts->cam_pts_homo->cam_pts->pix_x,pix_y得到采样点的像素坐标。通过

```
valid_pix = np.logical_and(pix_x >= 0,
            np.logical_and(pix_x < depth_im.shape[1],
            np.logical_and(pix_y >= 0, pix_y < depth_im.shape[0])))
```

过滤出在深度有效范围内的像素坐标并进行采样得到dep_val。

2.3 计算单帧 TSDF

由dep_val和pix_z可得到采样点的 sdf 值。用本身定义的trunc_margin进行截断。
可以由 $tsdf(x) = \max(-1, \min(1, sdf(x)/t))$ 得到单帧的 TSDF 值。

2.4 融合多帧 TSDF

首先要确定的是在截断边界内的是哪些像素坐标（valid_pts）。接着对目前的 tsdf 值和权重进行更新，参考公式：

$$D_{i+1}(x) = \frac{W_i(x)D_i(x) + w_{i+1}d_{i+1}(x)}{W_i(x) + w_{i+1}}$$

$$W_{i+1}(x) = W(x) + w_{i+1}$$

具体实现：

```
weight_new = weight_old + obs_weight
tsdf_vals_new = (weight_old * tsdf_vals + obs_weight * valid_dist) / weight_new
```

对于 tsdf 的属性进行更新，生成的mesh.obj效果如图 2.

2.5 实现带颜色的 TSDF Fusion

要实现带颜色的 TSDF Fusion 需要在以下几个地方进行改进：

1. 在demo.py里读 rgb 图像得到color_im，作为integrate函数的参数传入。
2. 将颜色图像转换成单通道形式，即组合成一个浮点数。

```
color_im = np.floor(color_im[...,0] * 256 * 256 + color_im[...,1] * 256
+ color_im[..., 2])
```

3. 在对 tsdf 融合之后，对 rgb 值同样进行融合，类似的公式为 $C_{i+1}(x) = \frac{W_i(x)C_i(x) + w_{i+1}c_{i+1}(x)}{W_i(x) + w_{i+1}}$
4. 提取 mesh 的时候要对颜色也进行处理。首先把顶点转成整数，采样颜色；再提取 rgb 值。

```
r(g,b)_new = np.minimum(255., np.round((weight_old * r(g,b)_old +
obs_weight * r(g,b)_new) / weight_new))
```

5. 利用文件中的gravity-direction.txt中的参数，把空间中的物体旋转至正确的视角位置。

最后输出到mesh_color.ply中，呈现效果如图 3.

3 实验结果

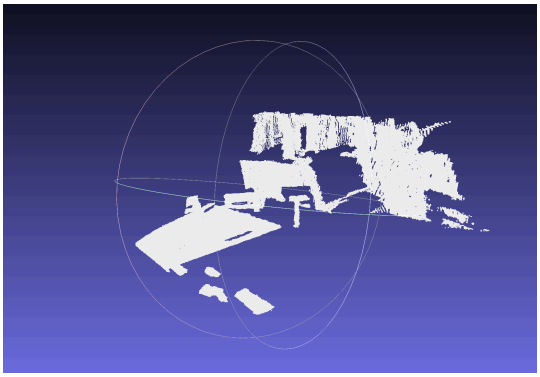

图 1: 点云

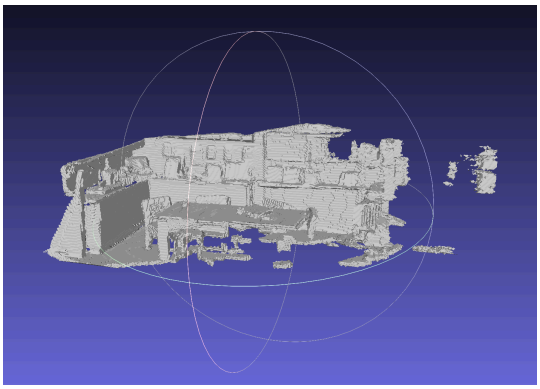

图 2: 无颜色的 mesh

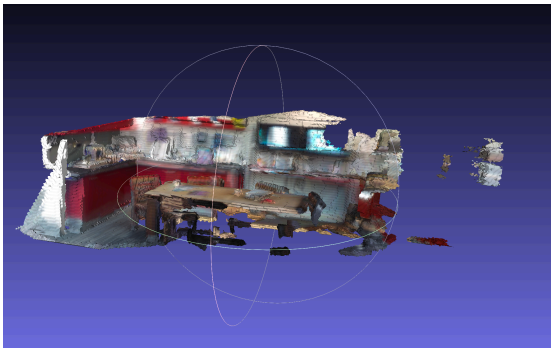

图 3: 带颜色的 mesh