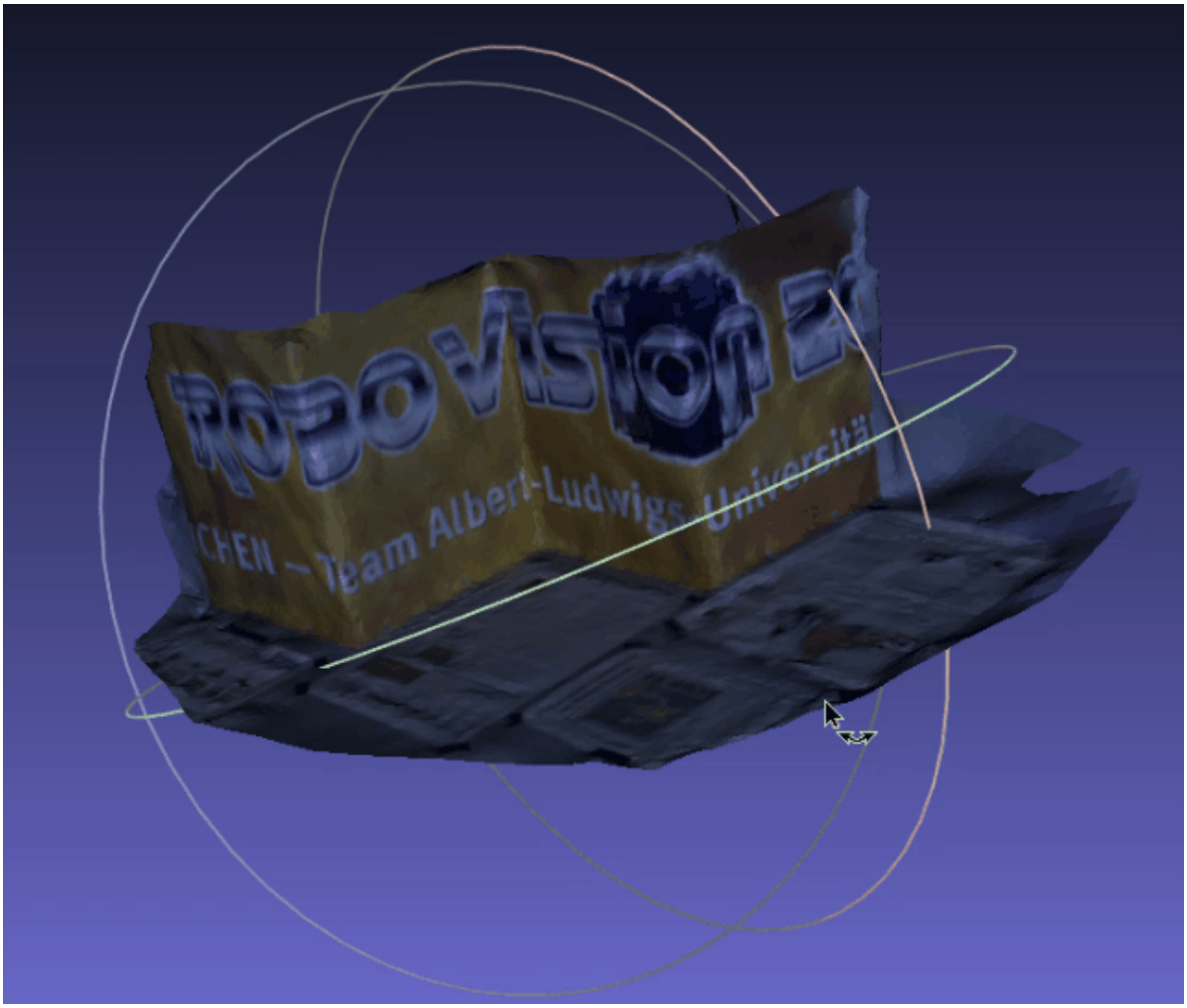
ORB-SLAM2同OpenMVS实现三维重建



ORB-SLAM2 位姿导出

Note:

为与OpenMVS进行对接本次进对ORB-SLAM2进行部分修改，使之可以为OpenMVS提供稀疏点云、关键帧的位姿、内参，以及稀疏点云在各个View中的可见性。

主要更改如下

- 在Map文件下增添如下函数

```
public:
    void Save(const string &filename,const cv::MatSize
image_size);
    void SaveMapPoint(ofstream &f, MapPoint* mp);
    void SaveKeyFrame(ofstream &f, KeyFrame* kf);
protected:
std::vector<int> KeyId;
```

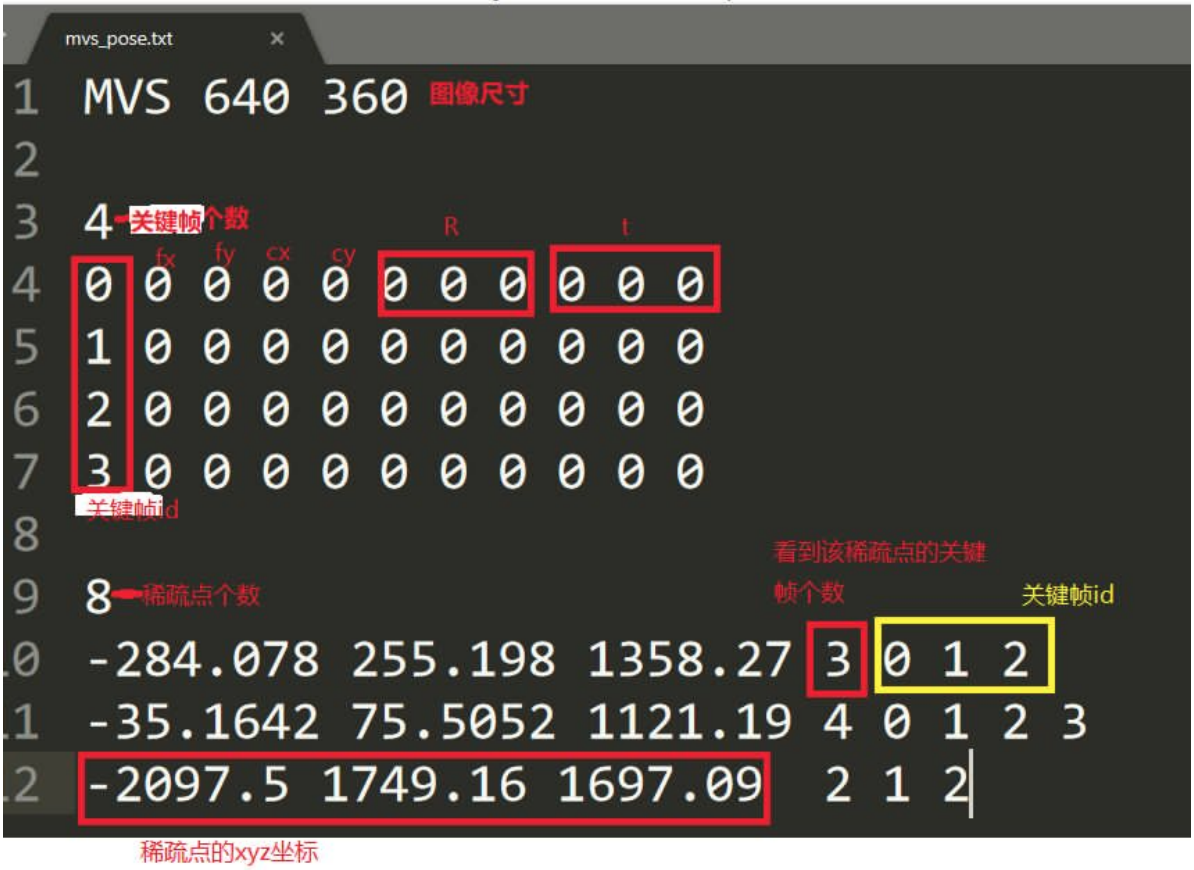
- 在System下增加:

```
void System::SaveMap(const string &filename,const
cv::MatSize image_size);
```

- 在mono_tum.cpp或者orbslam的其他Examples中对**System::SaveMap(const string &filename,const cv::MatSize image_size)**这个函数进行调用即可。

```
SLAM.SaveMap("../Examples/output/sfm.txt",im.size);
```

OpenMVS接受SLAM的数据格式



代码实现

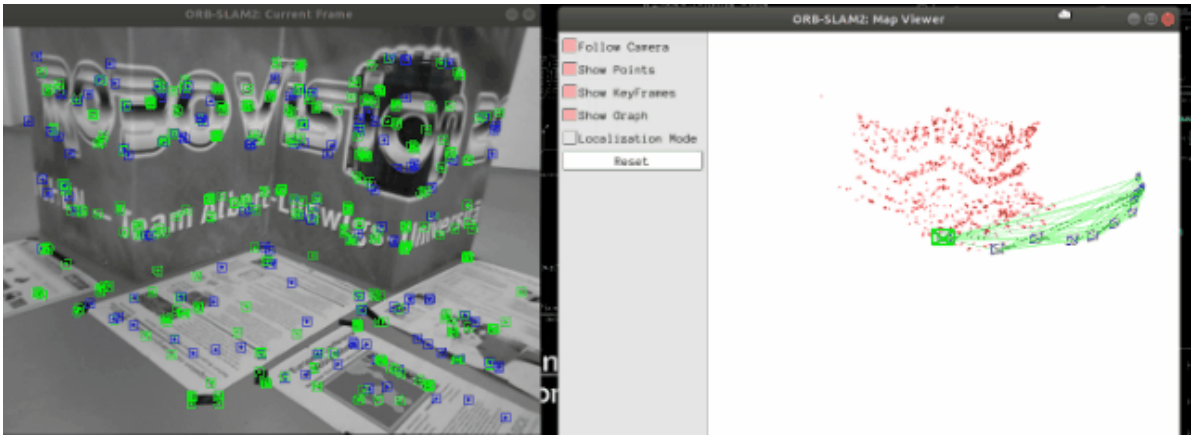
```cpp
//  保存关键帧
void Map::SaveKeyFrame(ofstream &f, KeyFrame *kf)
{
    KeyId.push_back(kf->mnId);
    // 保存当前关键帧的id
    f << KeyId.end() - KeyId.begin() - 1<< " ";
    // 关键帧内参
    f << kf->fx << " " << kf->fy << " " << kf->cx << " " << kf->cy << " ";
    // 保存当前关键帧的位姿
    cv::Mat Tcw = kf->GetPose();
    cout << "GetPose " << std::to_string(kf->mTimeStamp) <<"\nTcw\n" <<Tcw<< endl;
    cv::Mat Rcw = Tcw.rowRange(0,3).colRange(0,3);
    cout << "Rcw\n" << Rcw << endl;
    // 通过四元数保存旋转矩阵
    std::vector<float> Quat = Converter::toQuaternion(Rcw);

    for(int i=0; i<4; i++)
    {
        f << Quat[(3+i)%4] << " ";// qw qx qy qz
    }
    //保存平移
    for(int i=0; i<3; i++)
    {
        f << Tcw.at<float>(i,3) << " ";
    }
    ostringstream sTimeStamp;
    sTimeStamp << std::to_string(kf->mTimeStamp);
    f << sTimeStamp.str();
    f << "\n";
}
```

```cpp
//  保存地图点
void Map::SaveMapPoint(ofstream &f, MapPoint *mp)
{
    //保存当前MapPoint世界坐标值
    cv::Mat mpWorldPos = mp->GetWorldPos();
    f <<" " <<mpWorldPos.at<float>(0)<<" " << mpWorldPos.at<float>(1)<<" " << mpWorldPos.at<float>(2) << " ";
    f << (mp->nObs)/2<< " ";

    std::map<KeyFrame*,size_t> mapObservation = mp->GetObservations();
    for(auto mit = mapObservation.begin(); mit != mapObservation.end(); mit++)
    {
        int Frameid;
        Frameid = mit->first->mnId;
        auto keyid = find(KeyId.begin(),KeyId.end(),Frameid) - KeyId.begin();
        f << keyid << " ";
    }
    f << "\n";
}
```

```cpp
// 对关键帧相关数据进行保存
void Map::Save(const string &filename,const cv::MatSize image_size)
{
    std::cout << "SFM Saving to "<< filename << std::endl;
    ofstream f;
    f.open(filename.c_str());

    f << "MVS "<< image_size[1] << " "<< image_size[0] << endl;
    // 输出关键帧的数量
    cout << "The number of KeyFrames: " << mspKeyFrames.size() << endl;

    unsigned long int nKeyFrames = mspKeyFrames.size();
    f << nKeyFrames << endl;
    for(auto kf:mspKeyFrames)
        SaveKeyFrame(f,kf);

    // 输出空间三维点的数目
    cout << "The number of MapPoints: " << mspMapPoints.size();
    unsigned long int nMapPoints = mspMapPoints.size();
    f << nMapPoints << endl;

    for(auto mp:mspMapPoints)
        SaveMapPoint(f,mp);

    f.close();

}
```

ORB-SLAM2

数据集地址： https://vision.in.tum.de/data/datasets/rgbd-dataset/download#freiburg1_plant



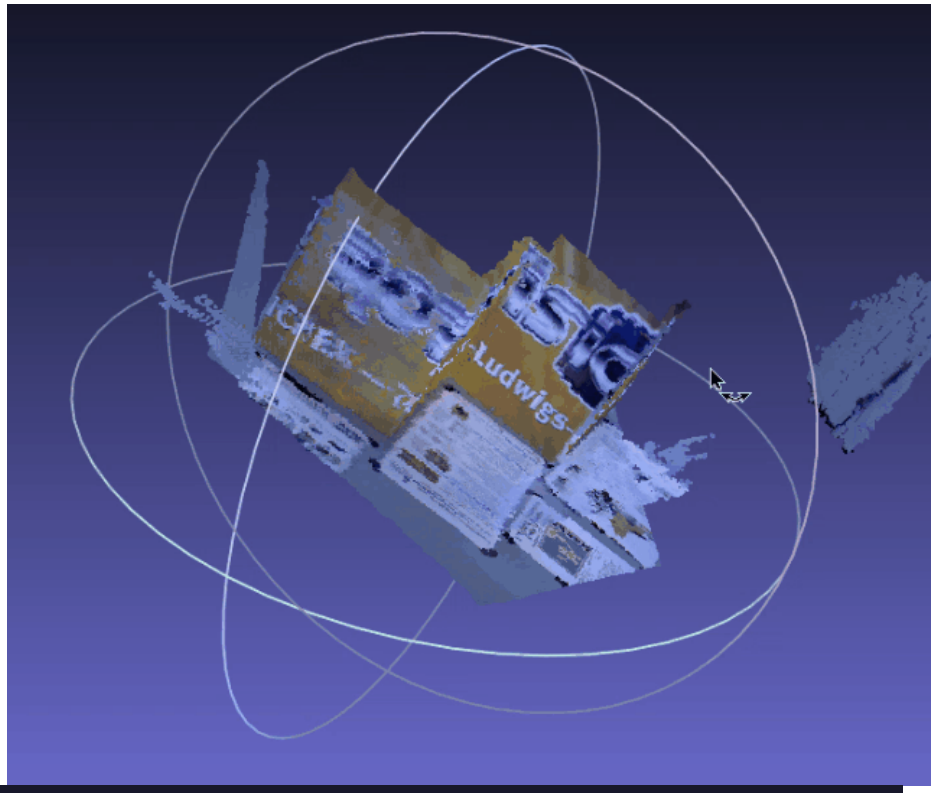
ORB-SLAM2位姿导出结果

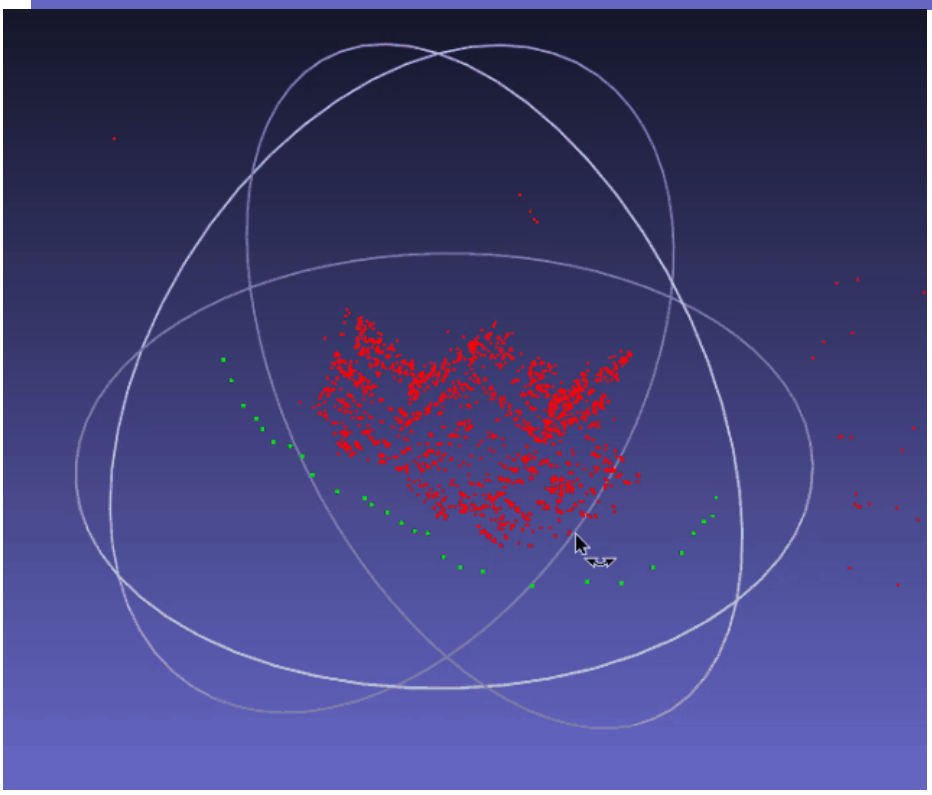
```
MVS 640 480
30
0 517.306 516.469 318.643 255.314 0.86853 0.0213837 -0.466605 -0.165767 2.21616 -0.562988 1.55612 1341839130.049672
1 517.306 516.469 318.643 255.314 0.994282 0.0287422 -0.0968751 -0.0345255 0.440294 -0.122722 0.213835 1341839117.145451
2 517.306 516.469 318.643 255.314 0.997234 0.0242307 -0.0616286 -0.0337579 0.272533 -0.0941566 0.12149 1341839116.205705
3 517.306 516.469 318.643 255.314 0.999146 0.0310468 -0.0251215 -0.0105882 0.148862 -0.0736897 0.0929523 1341839115.569633
4 517.306 516.469 318.643 255.314 1 0 0 0 0 0 0 1341839113.657637
```

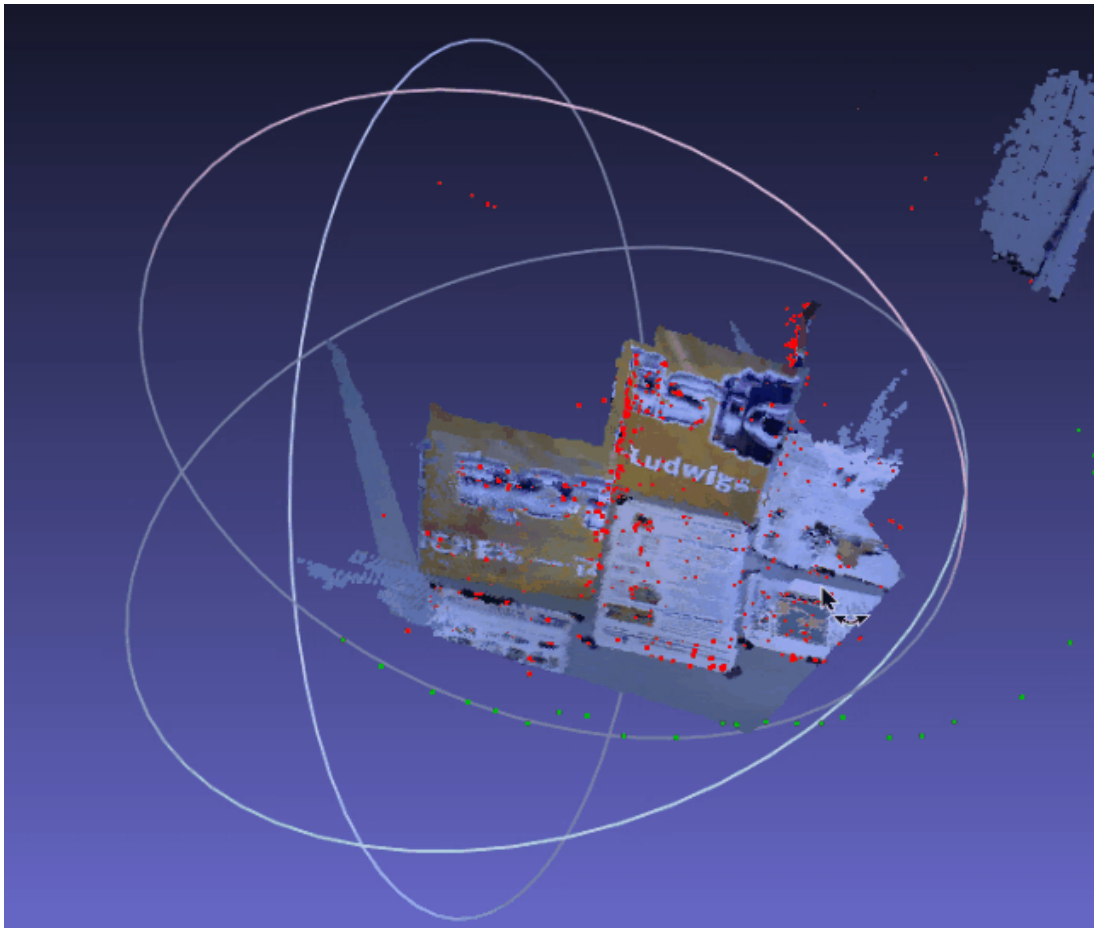
```
28 517.306 516.469 318.643 255.314 0.529015 0.0604791 -0.792094 -0.298452 1.91362 -1.56164 3.89984 1341839144.661818
29 517.306 516.469 318.643 255.314 0.569329 0.0480394 -0.770117 -0.283683 2.0505 -1.39629 3.73374 1341839142.985900
2668
 0.0897022 0.706964 1.34879 2 3 5
 -0.644566 0.630212 1.51819 6 1 3 5 6 10 12
 0.257397 0.658318 1.44936 5 1 2 3 5 6
 0.474476 0.667432 1.46848 4 1 2 3 5
 -0.255735 0.633334 1.51285 4 2 3 5 9
```

ORB-SLAM2导出位姿验证

在与OpenMVS 进行对接之前，一定确保自己导出的信息是准确的，可以将相机三位空间坐标点以及相机在空间中的位置保存成ply、obj等三维格式，在meshlab中进行查看，或者如果你用的rgb-d相机的话，同样可以将深度图、rgb图一同投影下来，在meshlab下进行查看


Ludwigs-





####

OPENMVS接口

为与SLAM进行对接，我们加入了read_pose.cpp、read_pose.h这两个c++文件，目的是对SLAM导出的位姿和稀疏点云进行读取，并对OpenMVS进行初始化。

主要核心函数有

```
bool load_scene(string file,Scene &scene);
bool read_mvs_pose(string file,MVSPOSE &mvs_pose);
bool save_pointcloud_obj(string name, vector<POINT3F>
points,int num_keyframes,RGB color)
```

InterfaceColMap接口理解

首先明确一点，我们SLAM的内容要如实的传入到**Scene**这个类中

```
PlatformArr platforms; // 相机内参和位姿。 camera platforms, each containing the mounted cameras and all known poses
ImageArr images;       // 图像 每帧对应的位姿在platforms中。 images, each referencing a platform's camera pose
PointCloud pointcloud; // 点云（开始存储的是读入的稀疏点云，后续存储的是深度图计算融合后稠密点云。point-cloud (sparse or dense),
Mesh mesh;             // 网格模型（顶点和faces），由pointcloud计算得到的。mesh, represented as vertices and triangles, co
```

首先将数据传入imgaes中

```
// a view instance seeing the scene
class MVS_API Image
{
public:
    uint32_t platformID; // ID of the associated platform
    uint32_t cameraID; // ID of the associated camera on the associated platform
    uint32_t poseID; // ID of the pose of the associated platform
    uint32_t ID; // global ID of the image
    String name; // image file name (relative path)
    Camera camera; // view's pose
    uint32_t width, height; // image size
    Image8U3 image; // image color pixels
    ViewScoreArr neighbors; // scored neighbor images
    float scale; // image scale relative to the original size
    float avgDepth; // average depth of the points seen by this camera
```

然后将数据传入platforms中

```
String name; // platform's name
CameraArr cameras; // cameras mounted on the platform
PoseArr poses; // trajectory of the platform
```

以及pointcloud

```
public:
    PointArr points;
    PointViewArr pointViews; // array of views for each point (ordered increasing)
    PointWeightArr pointWeights;
    NormalArr normals;
    ColorArr colors;
```

具体代码实现(节选)

```
bool load_scene(string file,Scene &scene)
{
    MVSPOSE mvs_pose;
    if(!read_mvs_pose(file,mvs_pose))
    {
        return false;
    }
    std::cout << " load mvs ok "  << std::endl;
    int numViews =mvs_pose.poses.size();
    scene.platforms.Reserve(numViews);
    scene.images.Reserve(numViews);
    scene.nCalibratedImages = 0;
    cout << "numViews" << mvs_pose.poses.size() << endl;
    vector<POINT3F> points;
    for (int count = 0; count < numViews; count++)
    {
        int idx = count; //true idx
        MVS::Image& image = scene.images.AddEmpty();

        string name="/home/wangwen/Desktop/三维重建/MVS/rgb/"+mvs_pose.images_name[idx]+".png";
        image.name=name;
        image.platformID = scene.platforms.GetSize();
        image.cameraID = 0;
        image.ID = idx;
        image.scale = 1.0;
```

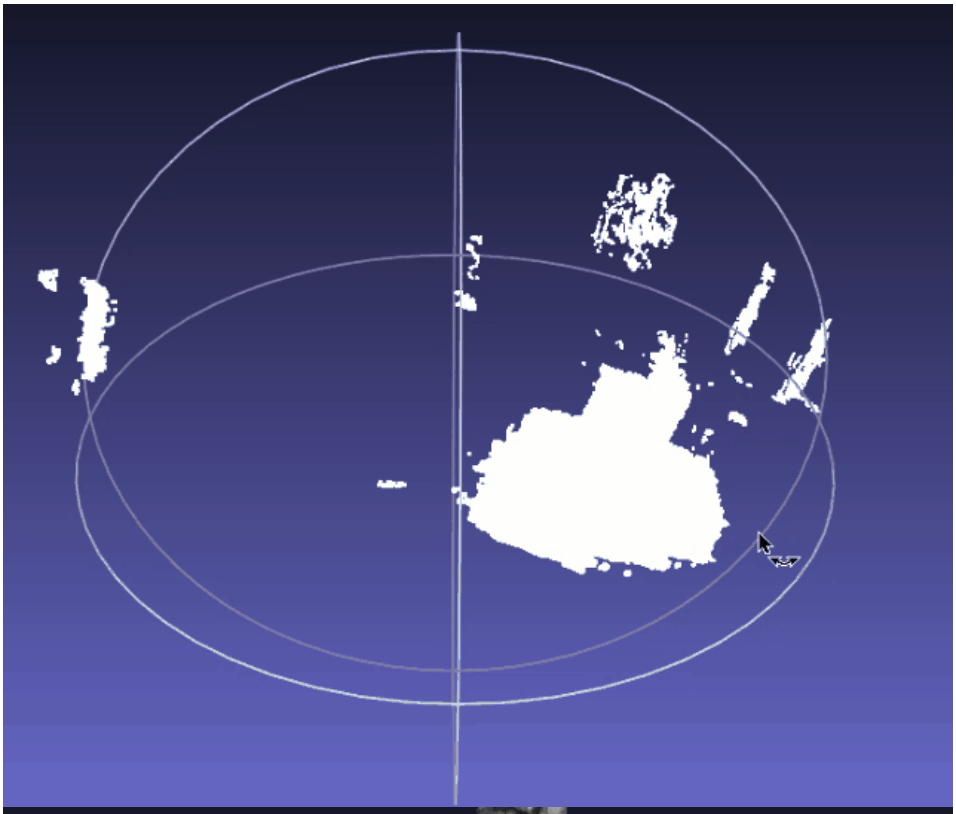
```
197         image.width = mvs_pose.width;
198         image.height = mvs_pose.height;
199 
200         MVS::Platform& platform = scene.platforms.AddEmpty();
201         MVS::Platform::Camera& camera = platform.cameras.AddEmpty();
202         cout << "wangwen "  <<mvs_pose.poses[idx].K[2] << "    "<< mvs_pose.poses[idx].K[5] << endl;
203         camera.K = MVS::Platform::Camera::ComposeK<REAL, REAL>(mvs_pose.poses[idx].K[0], mvs_pose.poses[idx].K[4] ,
204         camera.R = RMatrix::IDENTITY;
205         camera.C = CMatrix::ZERO;
206         cout << "camera.K" << camera.K << endl;
207         // normalize camera intrinsics
208         const REAL fScale(REAL(1) / MVS::Camera::GetNormalizationScale(image.width, image.height));
209         camera.K(0, 0) *= fScale;
210         camera.K(1, 1) *= fScale;
211         camera.K(0, 2) *= fScale;
212         camera.K(1, 2) *= fScale;
213         // set pose
214         image.poseID = platform.poses.GetSize();
215         MVS::Platform::Pose& pose = platform.poses.AddEmpty();
216 
217         for (int n = 0; n < 9; n++)
218         {
219             pose.R.val[n] = mvs_pose.poses[idx].rot[n];
220         }
```

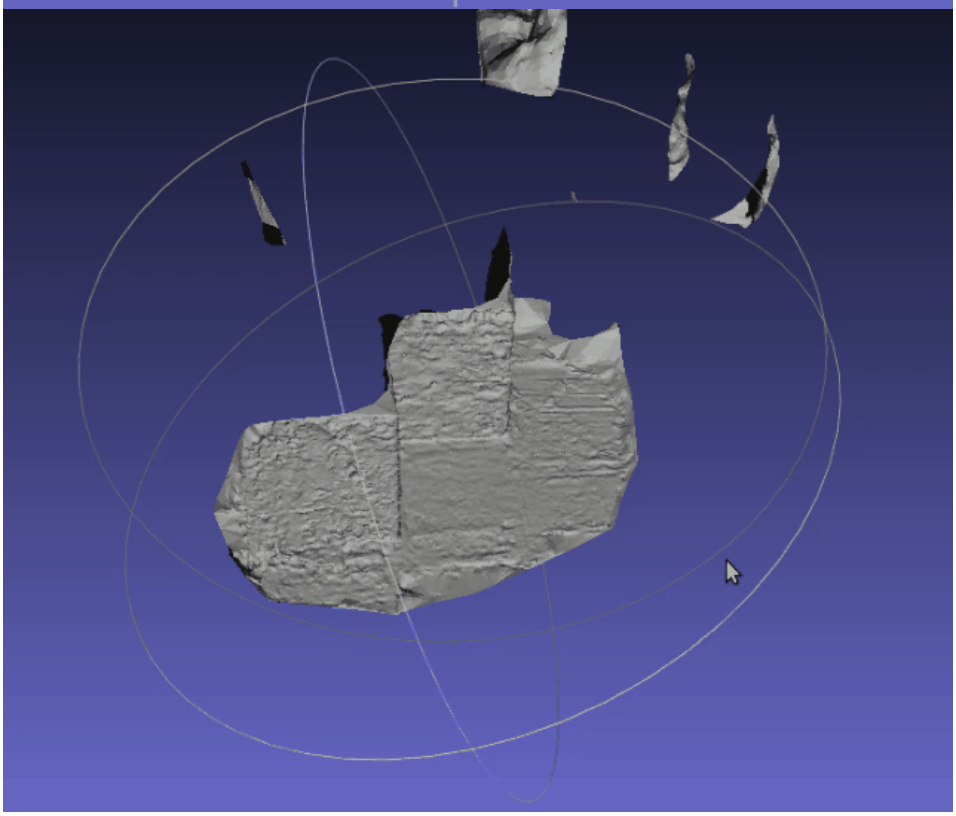
```
222         for (int j = 0; j < 3; ++j)
223         {
224             pose.C.ptr()[j] = -float(double(mvs_pose.poses[idx].rot[j])*double(mvs_pose.poses[idx].trans[0]) + doub
225         }
226 
227         cout <<"image.poseID " << image.poseID <<"image.platformID"<<image.platformID<<endl;
228         cout << "camera.R\n"<<camera.R << "camera.C" << camera.C << endl;
229 
230         POINT3F point_tmp(pose.C.x,pose.C.y,pose.C.z);
231         points.push_back(point_tmp);
232 
233         image.UpdateCamera(scene.platforms);
234 
235         ++scene.nCalibratedImages;
236 
237 //      cout <<"nCalibratedImages" << scene.nCalibratedImages << endl;
238     }
239     std::cout << "deal with feature points" << std::endl;
240     //deal with feature points
241     scene.pointcloud.points.Reserve(mvs_pose.spare_points.size());
242     scene.pointcloud.pointViews.Resize(mvs_pose.spare_points.size());
243     for (size_t idx = 0; idx<mvs_pose.spare_points.size(); ++idx)
244     {
245         POINT3F X = mvs_pose.spare_points[idx];
246         scene.pointcloud.points.AddConstruct(X.x, X.y, X.z);
247         MVS::PointCloud::ViewArr& views = scene.pointcloud.pointViews[idx];
```

```
248         int numview = mvs_pose.views[idx].size();
249         for (int viewId = 0; viewId<numview; viewId++)
250         {
251             views.InsertSort(mvs_pose.views[idx][viewId]);
252         }
253     }
254     RGB color(255,0,0);
255     save_pointcloud_obj("/home/wangwen/Desktop/三维重建/MVS/data/track_1.obj", points,mvs_pose.poses.size(),co
256     return true;
257 }
```

```
// load and estimate a dense point-cloud
#define use_custom_pose
#ifdef use_custom_pose

 if(!load_scene(string(MAKE_PATH_SAFE(OPT::strInputFileNam
e)),scene))
        return EXIT_FAILURE;
#else
    if
(!scene.Load(MAKE_PATH_SAFE(OPT::strInputFileName)))
        return EXIT_FAILURE;
#endif
```

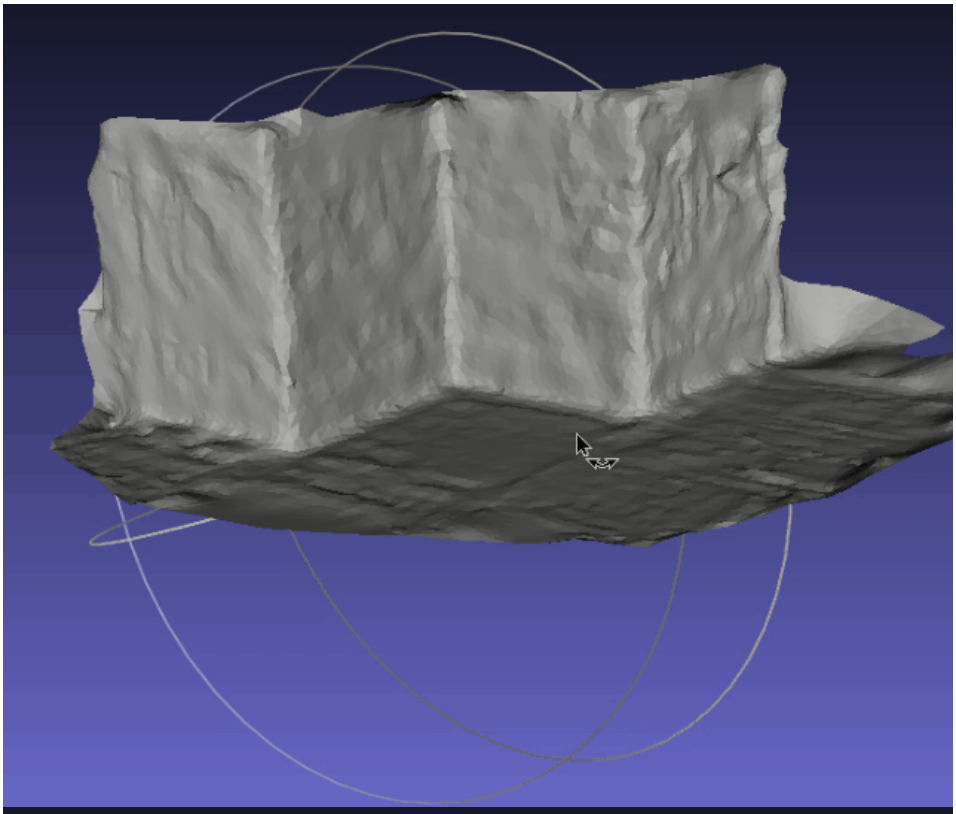
三维重建过程

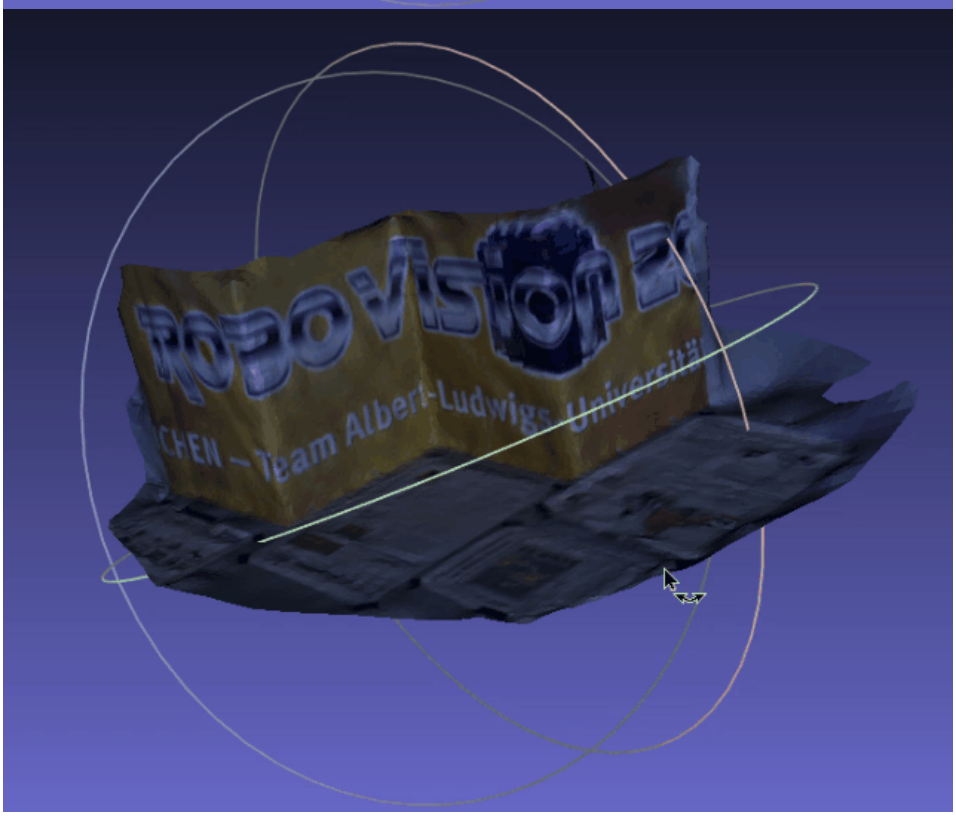




稠密重建

mesh重构





mesh优化

纹理贴图