csm改动代码

初始化PLicp参数

```
    void SetPIICPParams(){
        //设置激光的范围
        icpParam.min_reading = 0.1;
        icpParam.max_reading = 20;

        //设置位姿最大的变化范围
        icpParam.max_angular_correction_deg = 20.0;
        icpParam.max_linear_correction = 1;

        //设置迭代停止的条件
        icpParam.max_iterations = 50;
        icpParam.epsilon_xy = 0.000001;
        icpParam.epsilon_theta = 0.0000001;

        //设置correspondence相关参数
        icpParam.max_correspondence_dist = 1;
        icpParam.sigma = 0.01;
        icpParam.use_corr_tricks = 1;

        //设置restart过程，因为不需要restart所以可以不管
        icpParam.restart = 0;
        icpParam.restart_threshold_mean_error = 0.01;
        icpParam.restart_dt = 1.0;
        icpParam.restart_dtheta = 0.1;

        //设置聚类参数
        icpParam.clustering_threshold = 0.2;

        //用最近的10个点来估计方向
        icpParam.orientation_neighbourhood = 10;

        //设置使用PI-ICP
        icpParam.use_point_to_line_distance = 1;

        //不进行alpha_test
        icpParam.do_alpha_test = 0;
        icpParam.do_alpha_test_thresholdDeg = 5;

        //设置trimmed参数 用来进行outlier remove
        icpParam.outliers_maxPerc = 0.9;
        icpParam.outliers_adaptive_order = 0.7;
        icpParam.outliers_adaptive_mult = 2.0;

        //进行visibility_test 和 remove double
        icpParam.do_visibility_test = 1;
        icpParam.outliers_remove_doubles = 1;
        icpParam.do_compute_covariance = 0;
        icpParam.debug_verify_tricks = 0;
        icpParam.use_ml_weights = 0;
```

```
        icpParam.use_sigma_weights = 0;

    }
```

回调函数修改

```
void championLaserScanCallback(const
champion_nav_msgs::ChampionNavLaserScanConstPtr& msg)
{
    if(m_isFirstFrame == true)
    {
        std::cout <<"First Frame"<<std::endl;
        m_isFirstFrame = false;
        m_prevLaserPose = Eigen::Vector3d(0, 0, 0);
        pubPath(m_prevLaserPose, m_imlsPath, m_imlsPathPub);
        laserScanToLDP(msg,prevLDPScan);
        return ;
    }

    LDP curLDPScan;
    laserScanToLDP(msg,curLDPScan);

    prevLDPScan->odometry[0] = 0.0;
    prevLDPScan->odometry[1] = 0.0;
    prevLDPScan->odometry[2] = 0.0;

    prevLDPScan->estimate[0] = 0.0;
    prevLDPScan->estimate[1] = 0.0;
    prevLDPScan->estimate[2] = 0.0;

    prevLDPScan->true_pose[0] = 0.0;
    prevLDPScan->true_pose[1] = 0.0;
    prevLDPScan->true_pose[2] = 0.0;

    icpParam.laser_ref = prevLDPScan;
    icpParam.laser_sens = curLDPScan;

    icpParam.first_guess[0] = 0;
    icpParam.first_guess[1] = 0;
    icpParam.first_guess[2] = 0;
    icpParam.use_point_to_line_distance = 1;

    out.cov_x_m = 0;
    out.dx_dy1_m = 0;
    out.dx_dy2_m = 0;

    sm_icp(&icpParam,&out);

    if (out.valid)
    {
        std::cout << "transfrom: (" << out.x[0] << ", " << out.x[1] << ", "
            << out.x[2] * 180 / M_PI << ")" << std::endl;

    Eigen::Matrix3d lastPose;
    lastPose << cos(m_prevLaserPose(2)), -sin(m_prevLaserPose(2)),
m_prevLaserPose(0),
```

```
                sin(m_prevLaserPose(2)),  cos(m_prevLaserPose(2)),
m_prevLaserPose(1),
                0, 0, 1;
    Eigen::Matrix3d rPose;
    rPose << cos(out.x[2]), -sin(out.x[2]), out.x[0],
                sin(out.x[2]),  cos(out.x[2]), out.x[1],
                0, 0, 1;
    Eigen::Matrix3d nowPose = lastPose * rPose;

    m_prevLaserPose << nowPose(0, 2), nowPose(1, 2), atan2(nowPose(1,0),
nowPose(0,0));
    pubPath(m_prevLaserPose, m_imlsPath, m_imlsPathPub);
    }
    else
    {
        std::cout << "not Converged" << std::endl;
    }

    prevLDPScan = curLDPScan;

}
```

将自定义激光数据转换成csm中所需的LDP数据

```
    void laserScanToLDP(const champion_nav_msgs::ChampionNavLaserScanConstPtr&
msg,
                        LDP& out){

        int n = msg->ranges.size();
        out = ld_alloc_new(n);

        for(int i=0;i<n;i++){

            float r = msg->ranges[i];

            if (r > 0.1 && r < 20)
            {
                // 填充雷达数据
                out->valid[i] = 1;
                out->readings[i] = r;

                out->theta[i] = msg->angles[i];
            }
            else
            {
                out->valid[i] = 0;
                out->readings[i] = -1; // for invalid range

                out->theta[i] = msg->angles[i];
            }

        }
        out->min_theta = out->theta[0];
        out->max_theta = out->theta[n-1];

        out->odometry[0] = 0.0;
        out->odometry[1] = 0.0;
```

```
        out->odometry[2] = 0.0;

        out->true_pose[0] = 0.0;
        out->true_pose[1] = 0.0;
        out->true_pose[2] = 0.0;

    }
```