

aloam\_factor.hpp 7

aloam\_registration.cpp 3

src &gt; lidar\_localization &gt; src &gt; models &gt; loam &gt; aloam\_registration.cpp &gt; {} lidar\_localization &gt; AddPlaneNormFactor(const Eigen::Vector3d &amp;, co

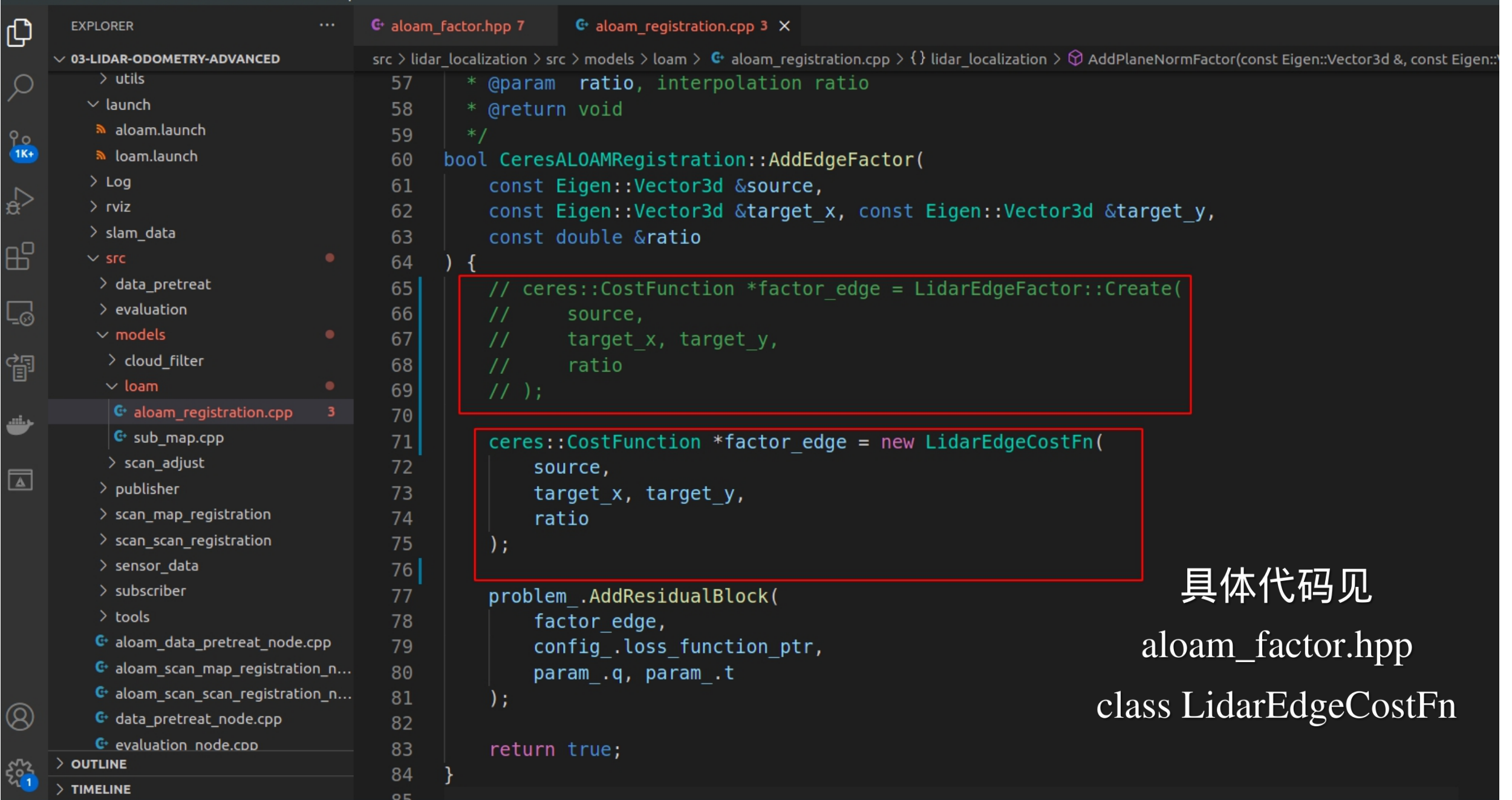
```
120  * @return void
121  */
122  bool CeresALOAMRegistration::AddPlaneNormFactor(
123      const Eigen::Vector3d &source,
124      const Eigen::Vector3d &norm, const double &negative_oa_dot_norm
125  ) {
126
127      // ceres::CostFunction *factor_plane = LidarPlaneNormFactor::Create(
128      //     source,
129      //     norm, negative_oa_dot_norm
130      // );|
131
132      ceres::CostFunction *factor_plane = new LidarPlaneNormCostFn(
133          source,
134          norm, negative_oa_dot_norm
135      );
136
137      problem_.AddResidualBlock(
138          factor_plane,
139          config_.loss_function_ptr,
140          param_.q, param_.t
141      );
142
143      return true;
144  }
```

具体代码见

aloam\_factor.hpp

class

LidarPlaneNormCostFn



```
57  * @param ratio, interpolation ratio
58  * @return void
59  */
60  bool CeresALOAMRegistration::AddEdgeFactor(
61      const Eigen::Vector3d &source,
62      const Eigen::Vector3d &target_x, const Eigen::Vector3d &target_y,
63      const double &ratio
64  ) {
65      // ceres::CostFunction *factor_edge = LidarEdgeFactor::Create(
66      //     source,
67      //     target_x, target_y,
68      //     ratio
69      // );
70
71      ceres::CostFunction *factor_edge = new LidarEdgeCostFn(
72          source,
73          target_x, target_y,
74          ratio
75      );
76
77      problem_.AddResidualBlock(
78          factor_edge,
79          config_.loss_function_ptr,
80          param_.q, param_.t
81      );
82
83      return true;
84  }
```

具体代码见  
aloam\_factor.hpp  
class LidarEdgeCostFn

