深蓝学院
shenlanxueyuan.com

第十章作业讲解

主讲人 Teamo

纲要

- 第一部分：补全代码，梳理逻辑
- 第二部分：与EKF进行对比
- 第三部分：不同长度窗口对于结果的影响

补全代码梳理逻辑

深蓝学院
shenlanxueyuan.com

```
// TODO: add residual blocks:
// b.1. marginalization constraint:
if (
    !residual_blocks_.map_matching_pose.empty() &&
    !residual_blocks_.relative_pose.empty() &&
    !residual_blocks_.imu_pre_integration.empty()
) {
    auto &key_frame_m = optimized_key_frames_.at(N - kWindowSize - 1);
    auto &key_frame_r = optimized_key_frames_.at(N - kWindowSize - 0);

    const ceres::CostFunction *factor_map_matching_pose = GetResMapMatchingPose(
        residual_blocks_.map_matching_pose.front()
    );
    const ceres::CostFunction *factor_relative_pose = GetResRelativePose(
        residual_blocks_.relative_pose.front()
    );
    const ceres::CostFunction *factor_imu_pre_integration = GetResIMUPreIntegration(
        residual_blocks_.imu_pre_integration.front()
    );

    sliding_window::FactorPRVAGMarginalization *factor_marginalization = new sliding_window::FactorPRVAGMarginaliz

    factor_marginalization->SetResMapMatchingPose(
        factor_map_matching_pose,
        std::vector<double *>{key_frame_m.prvag}
    );
    factor_marginalization->SetResRelativePose(
        factor_relative_pose,
        std::vector<double *>{key_frame_m.prvag, key_frame_r.prvag}
    );
    factor_marginalization->SetResIMUPreIntegration(
        factor_imu_pre_integration,
        std::vector<double *>{key_frame_m.prvag, key_frame_r.prvag}
    );
    factor_marginalization->Marginalize(key_frame_r.prvag);

}
```

补全代码梳理逻辑

深蓝学院 shenlanxueyuan.com

- 雷达位姿单边约束

```
void SetResMapMatchingPose(
  const ceres::CostFunction *residual,
  const std::vector<double *> &parameter_blocks
) {
  // init:
  ResidualBlockInfo res_map_matching_pose(residual, parameter_blocks);
  Eigen::VectorXd residuals;
  std::vector<Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic, Eigen::RowMajor>> jacobians;

  // compute:
  Evaluate(res_map_matching_pose, residuals, jacobians);
  const Eigen::MatrixXd &J_m = jacobians.at(0);

  //
  // TODO: Update H:
  //
  // a. H_mm:
  H_.block<15, 15>(INDEX_M, INDEX_M) += J_m.transpose() * J_m;

  //
  // TODO: Update b:
  //
  // a. b_m:
  b_.block<15,  1>(INDEX_M,       0) += J_m.transpose() * residuals;
}
```

由于是单边约束，所以这里面的H矩阵只有最左上角的一部分，同样b也只有最上面一部分。

补全代码梳理逻辑

```
void SetResIMUPreIntegration(
  const ceres::CostFunction *residual,
  const std::vector<double *> &parameter_blocks
) {
  // init:
  ResidualBlockInfo res_imu_pre_integration(residual, parameter_blocks);
  Eigen::VectorXd residuals;
  std::vector<Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic, Eigen::RowMajor>> jacobians;

  // compute:
  Evaluate(res_imu_pre_integration, residuals, jacobians);
  const Eigen::MatrixXd &J_m = jacobians.at(0);
  const Eigen::MatrixXd &J_r = jacobians.at(1);

  //
  // TODO: Update H:
  //
  // a. H_mm:
  H_.block<15, 15>(INDEX_M, INDEX_M) += J_m.transpose() * J_m;
  // b. H_mr:
  H_.block<15, 15>(INDEX_M, INDEX_R) += J_m.transpose() * J_r;
  // c. H_rm:
  H_.block<15, 15>(INDEX_R, INDEX_M) += J_r.transpose() * J_m;
  // d. H_rr:
  H_.block<15, 15>(INDEX_R, INDEX_R) += J_r.transpose() * J_r;


  //
  // Update b:
  //
  // a. b_m:
  b_.block<15,  1>(INDEX_M,       0) += J_m.transpose() * residuals;
  // a. b_r:
  b_.block<15,  1>(INDEX_R,       0) += J_r.transpose() * residuals;
}
```

```
    void SetResRelativePose(
  const ceres::CostFunction *residual,
  const std::vector<double *> &parameter_blocks
) {
  // init:
  ResidualBlockInfo res_relative_pose(residual, parameter_blocks);
  Eigen::VectorXd residuals;
  std::vector<Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic, Eigen::RowMajor>> jacobians;

  // compute:
  Evaluate(res_relative_pose, residuals, jacobians);
  const Eigen::MatrixXd &J_m = jacobians.at(0);
  const Eigen::MatrixXd &J_r = jacobians.at(1);

  //
  // TODO: Update H:
  //
  // a. H_mm:
  H_.block<15, 15>(INDEX_M, INDEX_M) += J_m.transpose() * J_m;
  // b. H_mr:
  H_.block<15, 15>(INDEX_M, INDEX_R) += J_m.transpose() * J_r;
  // c. H_rm:
  H_.block<15, 15>(INDEX_R, INDEX_M) += J_r.transpose() * J_m;
  // d. H_rr:
  H_.block<15, 15>(INDEX_R, INDEX_R) += J_r.transpose() * J_r;

  //
  // TODO: Update b:
  //
  // a. b_m:
  b_.block<15,  1>(INDEX_M,       0) += J_m.transpose() * residuals;
  // a. b_r:
  b_.block<15,  1>(INDEX_R,       0) += J_r.transpose() * residuals;
}
```

雷达相对位姿和IMU预积分为双边约束，所以对于第一帧和第二帧各15维的状态都有约束，H矩阵分为了四块，b矩阵也被分为了两部分

补全代码梳理逻辑

```cpp
void Marginalize(
    const double *raw_param_r_0
  ) {
    // TODO: implement marginalization logic
        // save x_m_0:
    Eigen::Map<const Eigen::Matrix<double, 15, 1>> x_0(raw_param_r_0);
    x_0_ = x_0;

    // marginalize:
    const Eigen::MatrixXd &H_mm = H_.block<15, 15>(INDEX_M, INDEX_M);
    const Eigen::MatrixXd &H_mr = H_.block<15, 15>(INDEX_M, INDEX_R);
    const Eigen::MatrixXd &H_rm = H_.block<15, 15>(INDEX_R, INDEX_M);
    const Eigen::MatrixXd &H_rr = H_.block<15, 15>(INDEX_R, INDEX_R);

    const Eigen::VectorXd &b_m = b_.block<15, 1>(INDEX_M, 0);
    const Eigen::VectorXd &b_r = b_.block<15, 1>(INDEX_R, 0);

    //
    // TODO: shall we improve numeric stability following VIO/LIO-mapping's practice?
    //
    Eigen::MatrixXd H_mm_inv = H_mm.inverse();
    Eigen::MatrixXd H_marginalized = H_rr - H_rm * H_mm_inv * H_mr;
    Eigen::MatrixXd b_marginalized = b_r - H_rm * H_mm_inv * b_m;

    //
    // solve linearized residual & Jacobian:
    //
    Eigen::SelfAdjointEigenSolver<Eigen::MatrixXd> saes(H_marginalized);
    Eigen::VectorXd S = Eigen::VectorXd(
      (saes.eigenvalues().array() > 1.0e-5).select(saes.eigenvalues().array(), 0)
    );
    Eigen::VectorXd S_inv = Eigen::VectorXd(
      (saes.eigenvalues().array() > 1.0e-5).select(saes.eigenvalues().array().inverse(), 0)
    );

    Eigen::VectorXd S_sqrt = S.cwiseSqrt();
    Eigen::VectorXd S_inv_sqrt = S_inv.cwiseSqrt();

    // finally:
    J_ = S_sqrt.asDiagonal() * saes.eigenvectors().transpose();
    e_ = S_inv_sqrt.asDiagonal() * saes.eigenvectors().transpose() * b_marginalized;
  }
```

补全代码梳理逻辑

深蓝学院
shenlanxueyuan.com

这里面的具体逻辑与ppt一致，需要注意的是如何从H和b矩阵中获得相应的雅各比和残差。这里我们使用奇异值分解

$$H = J^T J = QSQ^T$$

其中S为奇异值，Q为奇异值对应的右奇异向量，那么

$$J = \sqrt{S}Q^T$$

根据$b = J^T e$可以得到

$$e = J^{T^{-1}} b = \sqrt{S^{-1}}Q^T$$

补全代码梳理逻辑

深蓝学院
shenlanxueyuan.com

```
// add marginalization factor into sliding window
            problem.AddResidualBlock(
                factor_marginalization,
                NULL,
                key_frame_r.prvag
            );

            residual_blocks_.map_matching_pose.pop_front();
            residual_blocks_.relative_pose.pop_front();
            residual_blocks_.imu_pre_integration.pop_front();
```

补全代码梳理逻辑

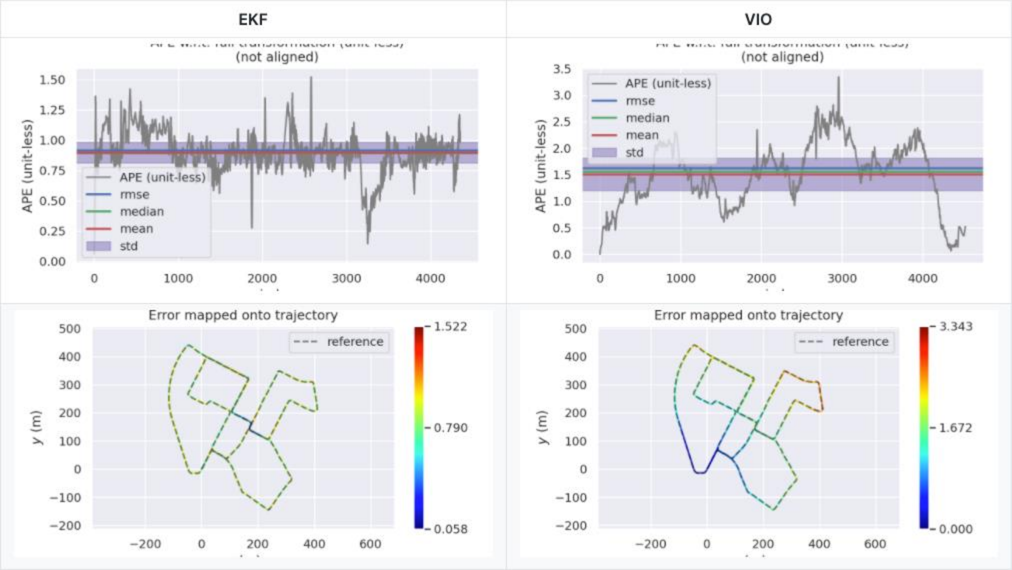
深蓝学院
shenlanxueyuan.com

```
virtual bool Evaluate(double const *const *parameters, double *residuals, double **jacobians) const {
  //
  // parse parameters:
  //
  Eigen::Map<const Eigen::Matrix<double, 15, 1>> x(parameters[0]);
  Eigen::VectorXd dx = x - x_0_;

  //
  // TODO: compute residual:
  //
  Eigen::Map<Eigen::Matrix<double, 15, 1>> residual(residuals);
  residual = e_ + J_ * dx;

  //
  // TODO: compute jacobian:
  //
  if ( jacobians ) {
    if ( jacobians[0] ) {
      // implement computing:
      Eigen::Map<Eigen::Matrix<double, 15, 15, Eigen::RowMajor> > jacobian_marginalization( jacobians[0] );
      jacobian_marginalization.setZero();

      jacobian_marginalization = J_;
    }
  }

  return true;
}
```

纲要

- 第一部分：补全代码，梳理逻辑
- 第二部分：与EKF进行对比
- 第三部分：不同长度窗口对于结果的影响

与EKF进行对比



EKF

```
APE w.r.t. full transformation (unit-less)
(not aligned)

       max      1.521982
      mean      0.897535
    median      0.894192
       min      0.058380
      rmse      0.913938
       sse      3635.146581
       std      0.172374
```

VIO

```
APE w.r.t. full transformation (unit-less)
(not aligned)

       max      3.342999
      mean      1.504630
    median      1.555986
       min      0.000001
      rmse      1.623278
       sse      11931.417004
       std      0.609196
```

纲要

- 第一部分：补全代码，梳理逻辑
- 第二部分：与EKF进行对比
- 第三部分：不同长度窗口对于结果的影响

纲要

- `WINDOWSIZE` = 10

```
APE w.r.t. full transformation (unit-less)
(not aligned)

       max	3.341343
      mean	1.486215
    median	1.527445
       min	0.000001
      rmse	1.599735
       sse	11587.833756
       std	0.591875
```

- `WINDOWSIZE` = 20

```
APE w.r.t. full transformation (unit-less)
(not aligned)

       max	3.342999
      mean	1.504630
    median	1.555986
       min	0.000001
      rmse	1.623278
       sse	11931.417004
       std	0.609196
```

- `WINDOWSIZE` = 30

```
APE w.r.t. full transformation (unit-less)
(not aligned)

       max	3.349993
      mean	1.512396
    median	1.562529
       min	0.000001
      rmse	1.634980
       sse	12104.070253
       std	0.621142
```

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening