UDACITY

‹ Return to Classroom

DISCUSS ON STUDENT HUB

Extended Kalman Filters

审阅

代码审阅 4

HISTORY

src/kalman_filter.cpp 3

```
#include "kalman_filter.h"
#include <iostream>


using Eigen::MatrixXd;
using Eigen::VectorXd;
using std::cout;
using std::endl;

/*
 * Please note that the Eigen library does not initialize
 *   VectorXd or MatrixXd objects with zeros upon creation.
 */

KalmanFilter::KalmanFilter() {}

KalmanFilter::~KalmanFilter() {}

void KalmanFilter::Init(VectorXd &x_in, MatrixXd &P_in, MatrixXd &F_in,
                        MatrixXd &H_in, MatrixXd &R_in, MatrixXd &Q_in) {
  x_ = x_in;
  P_ = P_in;
  F_ = F_in;
  H_ = H_in;
  R_ = R_in;
  Q_ = Q_in;
}

void KalmanFilter::Predict() {
  /**
   * TODO: predict the state
   */
  cout << "Initial x_ : " << x_ <<endl;
  cout << "Initial P_ : " << P_ <<endl;
  cout << "Initial Q_ : " << Q_ <<endl;
  x_ = F_ * x_;
  MatrixXd Ft = F_.transpose();
  P_ = F_ * P_ * Ft + Q_;
}

void KalmanFilter::Update(const VectorXd &z) {
  /**
   * TODO: update the state by using Kalman Filter equations
   */
  VectorXd z_pred = H_ * x_;
  VectorXd y = z - z_pred;
  MatrixXd Ht = H_.transpose();
  MatrixXd S = H_ * P_ * Ht + R_;
  MatrixXd Si = S.inverse();
  MatrixXd PHt = P_ * Ht;
  MatrixXd K = PHt * Si;

  //new estimate
  x_ = x_ + (K *y);
  long x_size = x_.size();
  MatrixXd I = MatrixXd::Identity(x_size, x_size);
  P_ = (I - K*H_) * P_;
```

建议

`Update()` and `UpdateEKF()` have common code. You could **refactor** the last few lines into a **c**
and `UpdateEKF()`.

```
KalmanFilter::UpdateCommon(const VectorXd& y)
   Ht = H_Transpose()
   S = H * P * Ht + R
   Si = S_Inverse()
   K = P_ * Ht * Si;

   x_ = x_ + (K * y);
   P_ = (I - K * H_) * P_;
```

```
}

void KalmanFilter::UpdateEKF(const VectorXd &z) {
  /**
   * TODO: update the state by using Extended Kalman Filter equations
   */
  float px = x_(0);
  float py = x_(1);
  float vx = x_(2);
  float vy = x_(3);

  // measurements
  float rho = sqrt(px*px + py*py);
  float theta = atan2(py, px);
  float rho_dot = (px*vx + py*vy) / rho;

```

建议

🌱 Whenever there is **division**, it's always a good idea to protect against **division by zero**. There are s
one:
`rho_dot = (px*vx + py*vx) / std:max(rho, eps);` with some small `eps`. e.g.: 0.001

```
    // map from cartesian to polar coordinates
    VectorXd h = VectorXd(3);
    h << rho, theta, rho_dot;
    //error function
    VectorXd y = z - h;

    // normalizing the angles
    while( y(1) > M_PI){
      y(1) -= 2* M_PI;
    }
    while(y(1)<-M_PI){
      y(1) += 2*M_PI;
```

棒极了

Excellent, Normalizing the resultant angle is a must ✅

🌱 Alternative way for normalization: There is a mathematical trick that you can use here: For an ang

```
y(1) = atan2(sin(y(1)), cos(y(1)));
```

```
    }

    MatrixXd Ht = H_.transpose();
    MatrixXd S = H_ * P_ * Ht + R_;
    MatrixXd Si = S.inverse();
    MatrixXd PHt = P_ * Ht;
    MatrixXd K = PHt * Si;

    //new estimate
    x_ = x_ + (K *y);
    long x_size = x_.size();
    MatrixXd I = MatrixXd::Identity(x_size, x_size);
    P_ = (I - K*H_) * P_;

}

```

▶ **src/FusionEKF.cpp** 1

▶ **src/tools.cpp**

▶ **src/main.cpp**

▶ **src/Eigen/src/plugins/CMakeLists.txt**

▶ src/Eigen/src/misc/CMakeLists.txt

▶ src/Eigen/src/UmfPackSupport/CMakeLists.txt

▶ src/Eigen/src/SuperLUSupport/CMakeLists.txt

▶ src/Eigen/src/StlSupport/CMakeLists.txt

▶ src/Eigen/src/SparseQR/CMakeLists.txt

▶ src/Eigen/src/SparseLU/CMakeLists.txt

▶ src/Eigen/src/SparseCore/CMakeLists.txt

▶ src/Eigen/src/SparseCholesky/CMakeLists.txt

▶ src/Eigen/src/SVD/CMakeLists.txt

▶ src/Eigen/src/SPQRSupport/CMakeLists.txt

▶ src/Eigen/src/QR/CMakeLists.txt

▶ src/Eigen/src/PardisoSupport/CMakeLists.txt

▶ src/Eigen/src/PaStiXSupport/CMakeLists.txt

▶ src/Eigen/src/OrderingMethods/CMakeLists.txt

▶ src/Eigen/src/MetisSupport/CMakeLists.txt

▶ src/Eigen/src/LU/arch/CMakeLists.txt

▶ src/Eigen/src/LU/CMakeLists.txt

▶ src/Eigen/src/Jacobi/CMakeLists.txt

▶ src/Eigen/src/IterativeLinearSolvers/CMakeLists.txt

▶ src/Eigen/src/Householder/CMakeLists.txt

▶ src/Eigen/src/Geometry/arch/CMakeLists.txt

▶ src/Eigen/src/Geometry/CMakeLists.txt

▸ src/Eigen/src/Eigenvalues/CMakeLists.txt

▸ src/Eigen/src/Eigen2Support/Geometry/CMakeLists.txt

▸ src/Eigen/src/Eigen2Support/CMakeLists.txt

▸ src/Eigen/src/Core/util/CMakeLists.txt

▸ src/Eigen/src/Core/products/CMakeLists.txt

▸ src/Eigen/src/Core/arch/SSE/CMakeLists.txt

▸ src/Eigen/src/Core/arch/NEON/CMakeLists.txt

▸ src/Eigen/src/Core/arch/Default/CMakeLists.txt

▸ src/Eigen/src/Core/arch/CMakeLists.txt

▸ src/Eigen/src/Core/arch/AltiVec/CMakeLists.txt

▸ src/Eigen/src/Core/CMakeLists.txt

▸ src/Eigen/src/CholmodSupport/CMakeLists.txt

▸ src/Eigen/src/Cholesky/CMakeLists.txt

▸ src/Eigen/src/CMakeLists.txt

▸ src/Eigen/CMakeLists.txt

▸ ide_profiles/xcode/README.md

▸ ide_profiles/README.md

▸ ide_profiles/Eclipse/README.md

▸ data/obj_pose-laser-radar-synthetic-input.txt

▸ cmakepatch.txt

▸ README.md

▸ Docs/Input_Output File Format.txt

▸ Docs/Data_Flow_Doc.txt

▸ CMakeLists.txt

返回 PATH

给这次审阅打分

开始