

无人驾驶技术——无损卡尔曼滤波 (UKF)

原创 xiao_lxl 发布于2019-09-02 18:06:38 阅读数 308 ☆ 收藏

展开



文章目录



首页

博客

学院

下载

论坛

问答

活动

专题

招聘

APP

VIP会员

免费8折

Python进阶之路



写博客



登录/注册

无损卡尔曼滤波 (UKF)

UKF Predict

1.选择预测点sigma点

sigma点计算代码实现

添加过程噪声处理

构建噪声扩充矩阵 c++

2.预测sigma点

预测sigma点方法

预测sigma点 c++代码实现

3.根据预测的sigma点预测状态均值和协方差矩阵

预测均值和协方差c++代码实现

UKF Update

测量预测

测量预测c++代码实现

Update之更新状态

更新状态c++代码实现

噪音参数和评估选择

如何选择噪音参数

直线加速度噪音参数直观

偏航加速度噪音参数直观

测量噪声参数

怎么评估选择

无人驾驶技术——无损卡尔曼滤波 (UKF)

前面总结了卡尔曼滤波 【无人驾驶技术——初探Kalman滤波器(KF)】 和
扩展卡尔曼滤波 【无人驾驶技术——扩展Kalman滤波 (EKF)】，
今天主要学习下无损卡尔曼滤波 (UKF)

无损卡尔曼滤波 (UKF)

无损卡尔曼滤波 (UKF) 是处理非线性过程模型或非线性测量模型的替代方案。但UKF不会对非线性函数进行线性化处理，它使用所谓的sigma点来近似概率分布。它有两个好处：

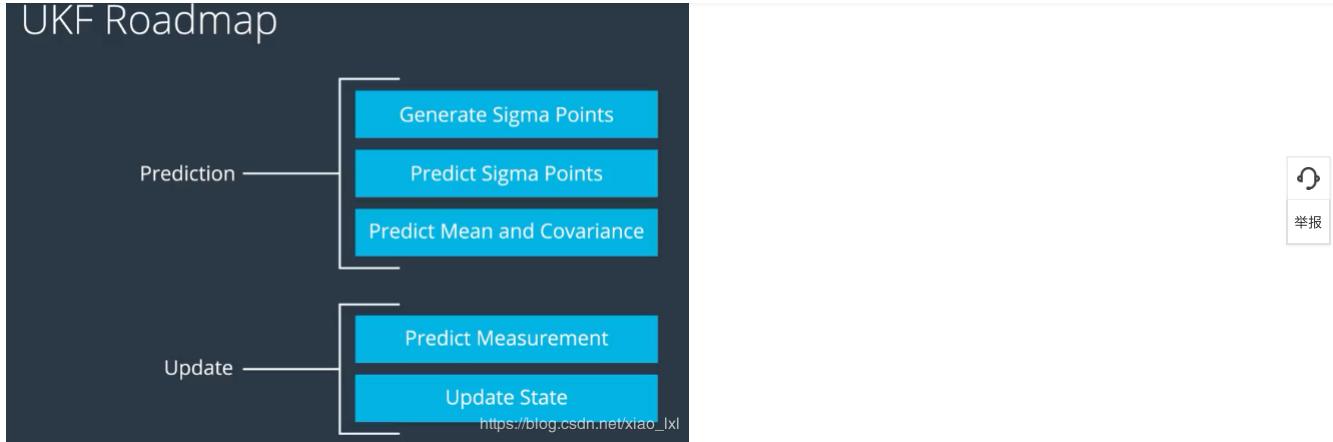
1. sigma点近似非线性转换的效果比线性化更好；
2. 可以省去计算雅可比矩阵的过程。

无损卡尔曼滤波器不需要将非线性函数线性化；相反，无损卡尔曼滤波器从高斯分布中取代表点。这些点将被插入非线性方程。

无损卡尔曼滤波器随时间处理测量值的方式和扩展卡尔曼滤波器是一样的。它们都有预测步骤，这个步骤是独立于测量模型的，在这一步里，需要使用上一节学到的CTRV模型。接下来是更新步骤，我们使用雷达测量模型或激光测量模型。

对于UKF 和 EKF,我们可以使用同样的顶层处理链，两者的不同之处是，UKF处理的是非线性过程模型或非线性测量模型的方法，它使用的方法叫无损变换。

UKF Roadmap



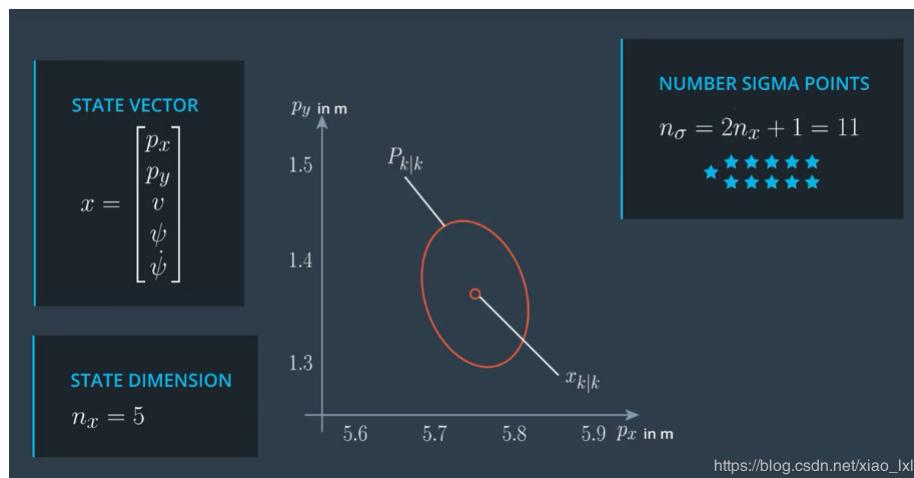
举报

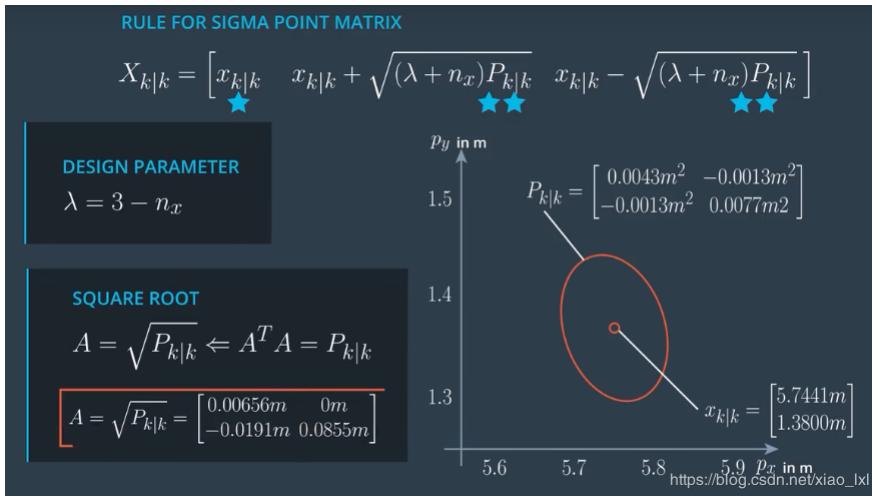
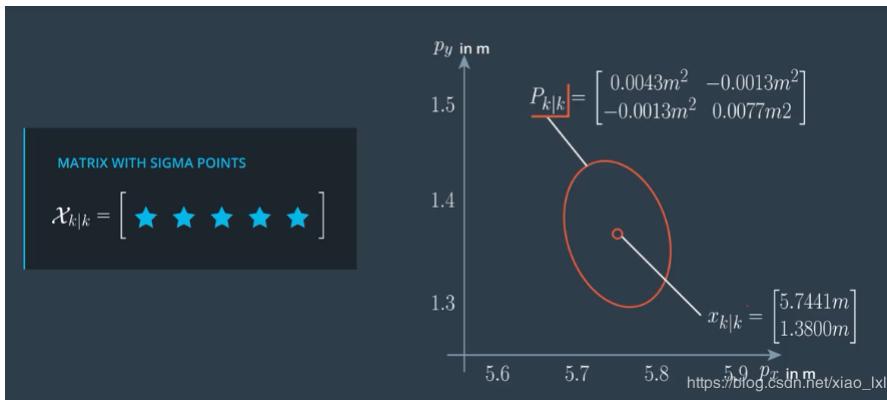
UKF Predict

无损滤波预测步骤主要分为三步：

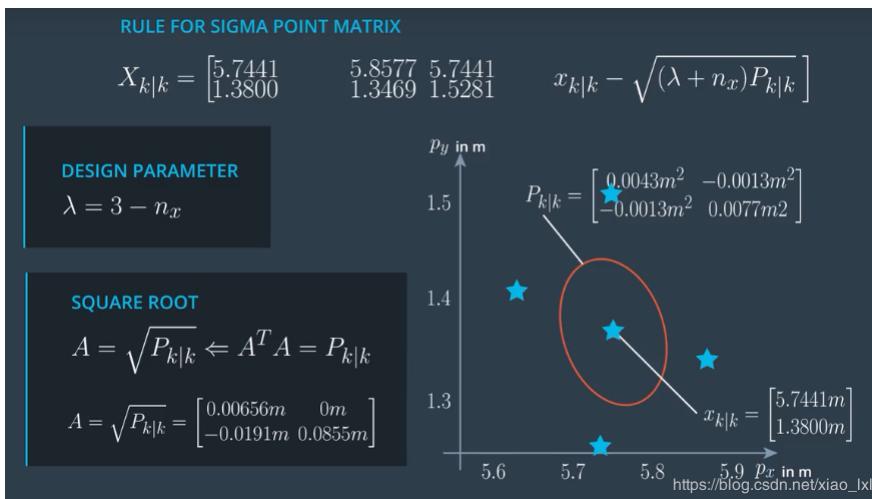
1. 需要知道选择sigma点的好办法
2. 需要知道如何预测sigma点
3. 需要知道如何根据预测的sigma点计算预测、均值和协方差

1. 选择预测点sigma点





将下面各个值代入上面的sigma点矩阵后，计算结果如下：



可见，主要是根据以下公式求得sigma值：

$$X_{k|k} = \begin{bmatrix} x_{k|k} & x_{k|k} + \sqrt{(\lambda + n_x)P_{k|k}} & x_{k|k} - \sqrt{(\lambda + n_x)P_{k|k}} \end{bmatrix}$$

remember that $x_{k|k}$ is the first column of the Sigma matrix.

$x_{k|k} + \sqrt{(\lambda + n_x)P_{k|k}}$ is the second through $n_x + 1$ column.

$x_{k|k} - \sqrt{(\lambda + n_x)P_{k|k}}$ is the $n_x + 2$ column through $2n_x + 1$ column.

sigma点计算代码实现

ukf.h

```
1 #ifndef UKF_H
2 #define UKF_H
3
4 #include "Dense"
```

```

7 | public:
8 | /**
9 |  * Constructor
10|  */
11| UKF();
12|
13| /**
14|  * Destructor
15|  */
16| virtual ~UKF();
17|
18| /**
19|  * Init Initializes Unscented Kalman filter
20|  */
21| void Init();
22|
23| /**
24|  * Student assignment functions
25|  */
26| void GenerateSigmaPoints(Eigen::MatrixXd* Xsig_out);
27| void AugmentedSigmaPoints(Eigen::MatrixXd* Xsig_out);
28| void SigmaPointPrediction(Eigen::MatrixXd* Xsig_out);
29| void PredictMeanAndCovariance(Eigen::VectorXd* x_pred,
30|                                Eigen::MatrixXd* P_pred);
31| void PredictRadarMeasurement(Eigen::VectorXd* z_out,
32|                               Eigen::MatrixXd* S_out);
33| void UpdateState(Eigen::VectorXd* x_out,
34|                  Eigen::MatrixXd* P_out);
35| };
36|
37| #endif // UKF_H

```

ukf.cpp

```

1 | #include "ukf.h"
2 | #include <iostream>
3 |
4 | using Eigen::MatrixXd;
5 | using Eigen::VectorXd;
6 |
7 | UKF::UKF() {
8 |     Init();
9 | }
10|
11| UKF::~UKF() {
12|
13| }
14|
15| void UKF::Init() {
16|
17| }
18|
19| /**
20|  * Programming assignment functions:
21|  */
22| void UKF::GenerateSigmaPoints(MatrixXd* Xsig_out) {
23|
24|     // set state dimension
25|     int n_x = 5;
26|
27|     // define spreading parameter
28|     double lambda = 3 - n_x;
29|
30|     // set example state
31|     VectorXd x = VectorXd(n_x);
32|     x << 5.7441,
33|          1.3800,
34|          2.2049,
35|          0.5015,
36|          0.3528;
37|
38|     // set example covariance matrix
39|     MatrixXd P = MatrixXd(n_x, n_x);
40|     P << 0.0043, -0.0013, 0.0030, -0.0022, -0.0020,
41|          -0.0013, 0.0077, 0.0011, 0.0071, 0.0060,
42|          0.0030, 0.0011, 0.0054, 0.0007, 0.0008,
43|          -0.0022, 0.0071, 0.0007, 0.0098, 0.0100,
44|          -0.0020, 0.0060, 0.0008, 0.0100, 0.0123;
45|
46|     // create sigma point matrix
47|     MatrixXd Xsig = MatrixXd(n_x, 2 * n_x + 1);
48|
49|     // calculate square root of P
50|     MatrixXd A = P.llt().matrixL();
51|
52|     /**
53|      * Student part begin
54|      */
55|
56|     // your code goes here
57|     // calculate sigma points ...
58|

```

```

61 // set remaining sigma points
62 for (int i = 0; i < n_x; ++i)
63 {
64     Xsig.col(i+1) = x + sqrt(lambda+n_x) * A.col(i);
65     Xsig.col(i+1+n_x) = x - sqrt(lambda+n_x) * A.col(i);
66 }
67
68 /**
69 * Student part end
70 */
71
72
73 // print result
74 // std::cout << "Xsig = " << std::endl << Xsig << std::endl;
75
76 // write result
77 *Xsig_out = Xsig;
78 }
79
80 /**
81 * expected result:
82 * Xsig =
83 * 5.7441 5.85768 5.7441 5.7441 5.7441 5.7441 5.63052 5.7441 5.7441 5.7441
84 * 1.38 1.34566 1.52806 1.38 1.38 1.38 1.41434 1.23194 1.38 1.38 1.38
85 * 2.2049 2.28414 2.24557 2.29582 2.2049 2.2049 2.12566 2.16423 2.11398 2.2049 2.2049
86 * 0.5015 0.44339 0.631886 0.516923 0.595227 0.5015 0.55961 0.371114 0.486077 0.407773 0.5015
87 * 0.3528 0.299973 0.462123 0.376339 0.48417 0.418721 0.405627 0.243477 0.329261 0.22143 0.286879
88 */

```

main.cpp

```

1 #include <iostream>
2 #include "Dense"
3 #include "ukf.h"
4
5 using Eigen::MatrixXd;
6
7 int main() {
8
9     // Create a UKF instance
10    UKF ukf;
11
12    /**
13     * Programming assignment calls
14     */
15    MatrixXd Xsig = MatrixXd(5, 11);
16    ukf.GenerateSigmaPoints(&Xsig);
17
18    // print result
19    std::cout << "Xsig = " << std::endl << Xsig << std::endl;
20
21    return 0;
22 }

```

添加过程噪声处理

为什么需要添加噪声进行扩充?

因为过程噪声对状态有非线性影响。

The diagram illustrates the process noise model. It shows the process noise vector $\nu_k = \begin{bmatrix} \nu_{a,k} \\ \nu_{\psi,k} \end{bmatrix}$ and the process model equation $x_{k+1} = f(x_k, \nu_k) = x_k + \begin{bmatrix} \frac{v_k}{\psi_k} (\sin(\psi_k + \dot{\psi}_k \Delta t) - \sin(\psi_k)) \\ \frac{v_k}{\psi_k} (-\cos(\psi_k + \dot{\psi}_k \Delta t) + \cos(\psi_k)) \\ 0 \\ \dot{\psi}_k \Delta t \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}(\Delta t)^2 \cos(\psi_k) \cdot \nu_{a,k} \\ \frac{1}{2}(\Delta t)^2 \sin(\psi_k) \cdot \nu_{a,k} \\ \Delta t \cdot \nu_{a,k} \\ \frac{1}{2}(\Delta t)^2 \cdot \nu_{\psi,k} \\ \Delta t \cdot \nu_{\psi,k} \end{bmatrix}$. A red box highlights the text: 'INDEPENDENT NOISE PROCESSES DOESN'T EXPRESS EFFECT ON STATE VECTOR INDEPENDENT OF Δt '.

PROCESS NOISE $\nu_k = \begin{bmatrix} \nu_{a,k} \\ \nu_{\dot{\psi},k} \end{bmatrix}$	STOCHASTIC PROPERTIES $\nu_{a,k} \sim N(0, \sigma_a^2)$ $\nu_{\dot{\psi},k} \sim N(0, \sigma_{\dot{\psi}}^2)$
PROCESS NOISE COVARIANCE MATRIX $Q = E \left\{ \nu_k \cdot \nu_k^T \right\} = \begin{bmatrix} \sigma_a^2 & 0 \\ 0 & \sigma_{\dot{\psi}}^2 \end{bmatrix}$	

https://blog.csdn.net/xiao_lxl

POSTERIOR DISTRIBUTION $x_{k k}$ $P_{k k}$	PROCESS NOISE $\nu_k = \begin{bmatrix} \nu_{a,k} \\ \nu_{\dot{\psi},k} \end{bmatrix}$
PROCESS MODEL $x_{k+1} = f(x_k, \nu_k)$	
PROCESS NOISE COVARIANCE MATRIX $Q = E \left\{ \nu_k \cdot \nu_k^T \right\} = \begin{bmatrix} \sigma_a^2 & 0 \\ 0 & \sigma_{\dot{\psi}}^2 \end{bmatrix}$	

https://blog.csdn.net/xiao_lxl

STATE VECTOR $x_k = \begin{bmatrix} p_x \\ p_y \\ v \\ \psi \\ \dot{\psi} \end{bmatrix}$	STATE DIMENSION $n_x = 5$	NUMBER SIGMA POINTS $n_\sigma = 2n_x + 1 = 11$ ★ ★ ★ ★ ★ ★ ★
CALCULATE SIGMA POINTS		
$X_{k k} = \begin{bmatrix} x_{k k} & x_{k k} + \sqrt{(\lambda + n_x)P_{k k}} & x_{k k} - \sqrt{(\lambda + n_x)P_{k k}} \end{bmatrix}$ <p style="text-align: center;">with scaling factor $\lambda = 3 - n_x$</p>		

https://blog.csdn.net/xiao_lxl

AUGMENTED STATE $x_{a,k} = \begin{bmatrix} p_x \\ p_y \\ v \\ \psi \\ \dot{\psi} \\ \nu_a \\ \nu_{\dot{\psi}} \end{bmatrix}$	AUGMENTED STATE DIMENSION $n_a = 7$	NUMBER SIGMA POINTS $n_\sigma = 2n_a + 1 = 15$ ★ ★ ★ ★ ★ ★ ★	AUGMENTED COVARIANCE MATRIX $P_{a,k k} = \begin{bmatrix} P_{k k} & 0 \\ 0 & Q \end{bmatrix}$
CALCULATE AUGMENTED SIGMA POINTS			
$X_{a,k k} = \begin{bmatrix} x_{a,k k} & x_{a,k k} + \sqrt{(\lambda + n_a)P_{a,k k}} & x_{a,k k} - \sqrt{(\lambda + n_a)P_{a,k k}} \end{bmatrix}$ <p style="text-align: center;">with scaling factor $\lambda = 3 - n_a$</p>			

https://blog.csdn.net/xiao_lxl

构建噪声扩充矩阵 C++

```

1 #ifndef UKF_H
2 #define UKF_H
3
4 #include "Dense"
5
6 class UKF {
7 public:
8     /**
9      * Constructor
10     */
11     UKF();
12
13     /**
14      * Destructor
15     */
16     virtual ~UKF();
17
18     /**
19      * Init Initializes Unscented Kalman filter
20     */
21     void Init();
22
23     /**
24      * Student assignment functions
25     */
26     void GenerateSigmaPoints(Eigen::MatrixXd* Xsig_out);
27     void AugmentedSigmaPoints(Eigen::MatrixXd* Xsig_out);
28     void SigmaPointPrediction(Eigen::MatrixXd* Xsig_out);
29     void PredictMeanAndCovariance(Eigen::VectorXd* x_pred,
30                                     Eigen::MatrixXd* P_pred);
31     void PredictRadarMeasurement(Eigen::VectorXd* z_out,
32                                  Eigen::MatrixXd* S_out);
33     void UpdateState(Eigen::VectorXd* x_out,
34                      Eigen::MatrixXd* P_out);
35 };
36
37 #endif // UKF_H

```

ukf.cpp

```

1 #include <iostream>
2 #include "ukf.h"
3
4 using Eigen::MatrixXd;
5 using Eigen::VectorXd;
6
7 UKF::UKF() {
8     Init();
9 }
10
11 UKF::~UKF() {
12 }
13
14
15 void UKF::Init() {
16 }
17
18
19 /**
20  * Programming assignment functions:
21 */
22
23
24 void UKF::AugmentedSigmaPoints(MatrixXd* Xsig_out) {
25
26     // set state dimension
27     int n_x = 5;
28
29     // set augmented dimension
30     int n_aug = 7;
31
32     // Process noise standard deviation longitudinal acceleration in m/s^2
33     double std_a = 0.2;
34
35     // Process noise standard deviation yaw acceleration in rad/s^2
36     double std_yawdd = 0.2;
37
38     // define spreading parameter
39     double lambda = 3 - n_aug;
40
41     // set example state
42     VectorXd x = VectorXd(n_x);
43     x << 5.7441,
44             1.3800,
45             2.2049,
46             0.5015,
47             0.3528;
48
49     // create example covariance matrix
50     MatrixXd P = MatrixXd(n_x, n_x);
51     P << 0.0043, -0.0013, 0.0030, -0.0022, -0.0020,
52             -0.0013, 0.0077, 0.0011, 0.0071, 0.0060,

```

```

55      -0.0020,     0.0060,    0.0008,    0.0100,    0.0123;
56
57 // create augmented mean vector
58 VectorXd x_aug = VectorXd(7);
59
60 // create augmented state covariance
61 MatrixXd P_aug = MatrixXd(7, 7);
62
63 // create sigma point matrix
64 MatrixXd Xsig_aug = MatrixXd(n_aug, 2 * n_aug + 1);
65
66 /**
67  * Student part begin
68 */
69
70 // create augmented mean state
71 x_aug.head(5) = x;
72 x_aug(5) = 0;
73 x_aug(6) = 0;
74
75 // create augmented covariance matrix
76 P_aug.fill(0.0);
77 P_aug.topLeftCorner(5,5) = P;
78 P_aug(5,5) = std_a*std_a;
79 P_aug(6,6) = std_yawdd*std_yawdd;
80
81 // create square root matrix
82 MatrixXd L = P_aug.llt().matrixL();
83
84 // create augmented sigma points
85 Xsig_aug.col(0) = x_aug;
86 for (int i = 0; i < n_aug; ++i) {
87     Xsig_aug.col(i+1) = x_aug + sqrt(lambda+n_aug) * L.col(i);
88     Xsig_aug.col(i+1+n_aug) = x_aug - sqrt(lambda+n_aug) * L.col(i);
89 }
90
91 /**
92  * Student part end
93 */
94
95 // print result
96 std::cout << "Xsig_aug = " << std::endl << Xsig_aug << std::endl;
97
98 // write result
99 *Xsig_out = Xsig_aug;
100}
101
102 /**
103  * expected result:
104  * Xsig_aug =
105  * 5.7441 5.85768 5.7441 5.7441 5.7441 5.7441 5.7441 5.63052 5.7441 5.7441 5.7441 5.7441 5.7441 5.7441 5.7441
106  * 1.38 1.34566 1.52806 1.38 1.38 1.38 1.38 1.41434 1.23194 1.38 1.38 1.38 1.38 1.38 1.38
107  * 2.2049 2.28414 2.24557 2.29582 2.2049 2.2049 2.2049 2.2049 2.12566 2.16423 2.11398 2.2049 2.2049 2.2049 2.2049
108  * 0.5015 0.44339 0.631886 0.516923 0.595227 0.5015 0.5015 0.5015 0.55961 0.371114 0.486077 0.407773 0.5015 0.5015 0.5015
109  * 0.3528 0.299973 0.462123 0.376339 0.48417 0.418721 0.3528 0.3528 0.405627 0.243477 0.329261 0.22143 0.286879 0.3528 0.3528
110  * 0 0 0 0 0 0 0 0.34641 0 0 0 0 0 0 -0.34641 0
111  * 0 0 0 0 0 0 0 0.34641 0 0 0 0 0 0 -0.34641 0
112 */

```

main.cpp

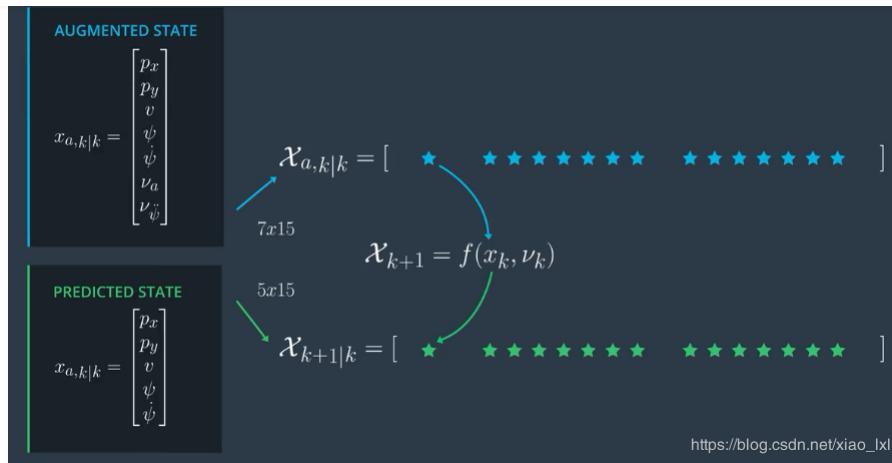
```

1 #include "Dense"
2 #include "ukf.h"
3
4 using Eigen::MatrixXd;
5
6 int main() {
7
8     // Create a UKF instance
9     UKF ukf;
10
11    /**
12     * Programming assignment calls
13     */
14    MatrixXd Xsig_aug = MatrixXd(7, 15);
15    ukf.AugmentedSigmaPoints(&Xsig_aug);
16
17    return 0;
18 }

```

2. 预测sigma点

预测sigma点方法



$$x = \begin{bmatrix} p_x \\ p_y \\ v \\ \psi \\ \dot{\psi} \end{bmatrix}$$

If $\dot{\psi}_k$ is not zero:

$$\text{State} = x_{k+1} = x_k + \begin{bmatrix} \frac{v_k}{\dot{\psi}_k} (\sin(\psi_k + \dot{\psi}_k \Delta t) - \sin(\psi_k)) \\ \frac{v_k}{\dot{\psi}_k} (-\cos(\psi_k + \dot{\psi}_k \Delta t) + \cos(\psi_k)) \\ 0 \\ \dot{\psi}_k \Delta t \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}(\Delta t)^2 \cos(\psi_k) \nu_{a,k} \\ \frac{1}{2}(\Delta t)^2 \sin(\psi_k) \nu_{a,k} \\ \Delta t \cdot \nu_{a,k} \\ \frac{1}{2}(\Delta t)^2 \nu_{\dot{\psi},k} \\ \Delta t \cdot \nu_{\dot{\psi},k} \end{bmatrix}$$

If $\dot{\psi}_k$ is zero:

$$\text{State} = x_{k+1} = x_k + \begin{bmatrix} v_k \cos(\psi_k) \Delta t \\ v_k \sin(\psi_k) \Delta t \\ 0 \\ \dot{\psi}_k \Delta t \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}(\Delta t)^2 \cos(\psi_k) \nu_{a,k} \\ \frac{1}{2}(\Delta t)^2 \sin(\psi_k) \nu_{a,k} \\ \Delta t \cdot \nu_{a,k} \\ \frac{1}{2}(\Delta t)^2 \nu_{\dot{\psi},k} \\ \Delta t \cdot \nu_{\dot{\psi},k} \end{bmatrix}$$

https://blog.csdn.net/xiao_lxl

Notice that when $\dot{\psi}_k = 0$,

the term $\dot{\psi}_k \Delta t$ would also equal zero.

预测sigma点 c++代码实现

ukf.h

```

1 #ifndef UKF_H
2 #define UKF_H
3
4 #include "Dense"
5
6 class UKF {
7 public:
8     /**
9      * Constructor
10     */
11     UKF();
12
13     /**
14      * Destructor
15     */
16     virtual ~UKF();
17
18     /**
19      * Init Initializes Unscented Kalman filter
20     */
21     void Init();
22
23     /**
24      * Student assignment functions
25     */
26     void GenerateSigmaPoints(Eigen::MatrixXd* Xsig_out);
27     void AugmentedSigmaPoints(Eigen::MatrixXd* Xsig_out);
28     void SigmaPointPrediction(Eigen::MatrixXd* Xsig_out);
29     void PredictMeanAndCovariance(Eigen::VectorXd* x_pred,
30                                     Eigen::MatrixXd* P_pred);
31     void PredictRadarMeasurement(Eigen::VectorXd* z_out,
```

```

34     Eigen::MatrixXd* P_out);
35 };
36
37 #endif // UKF_H

ukf.cpp

1 #include <iostream>
2 #include "ukf.h"
3
4 using Eigen::MatrixXd;
5 using Eigen::VectorXd;
6
7 UKF::UKF() {
8     Init();
9 }
10
11 UKF::~UKF() {
12 }
13
14
15 void UKF::Init() {
16 }
17
18
19
20 /**
21 * Programming assignment functions:
22 */
23
24 void UKF::SigmaPointPrediction(MatrixXd* Xsig_out) {
25
26     // set state dimension
27     int n_x = 5;
28
29     // set augmented dimension
30     int n_aug = 7;
31
32     // create example sigma point matrix
33     MatrixXd Xsig_aug = MatrixXd(n_aug, 2 * n_aug + 1);
34     Xsig_aug <<
35         5.7441, 5.85768, 5.7441, 5.7441, 5.7441, 5.7441, 5.7441, 5.63052, 5.7441, 5.7441, 5.7441, 5.7441, 5.7441, 5.7441,
36         1.38, 1.34566, 1.52806, 1.38, 1.38, 1.38, 1.38, 1.41434, 1.23194, 1.38, 1.38, 1.38, 1.38, 1.38,
37         2.2049, 2.28414, 2.24557, 2.29582, 2.2049, 2.2049, 2.2049, 2.12566, 2.16423, 2.11398, 2.2049, 2.2049, 2.2049,
38         0.5015, 0.44339, 0.631886, 0.516923, 0.595227, 0.5015, 0.5015, 0.5015, 0.55961, 0.371114, 0.486077, 0.407773, 0.5015, 0.5015, 0.5015,
39         0.3528, 0.299973, 0.462123, 0.376339, 0.48417, 0.418721, 0.3528, 0.3528, 0.405627, 0.243477, 0.329261, 0.22143, 0.286879, 0.3528, 0.3528,
40         0, 0, 0, 0, 0, 0, 0.34641, 0, 0, 0, 0, 0, 0, -0.34641, 0,
41         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -0.34641;
42
43     // create matrix with predicted sigma points as columns
44     MatrixXd Xsig_pred = MatrixXd(n_x, 2 * n_aug + 1);
45
46     double delta_t = 0.1; // time diff in sec
47
48 /**
49 * Student part begin
50 */
51
52     // predict sigma points
53
54     // avoid division by zero
55
56     // write predicted sigma points into right column
57
58     // predict sigma points
59     for (int i = 0; i < 2*n_aug+1; ++i) {
60         // extract values for better readability
61         double p_x = Xsig_aug(0,i);
62         double p_y = Xsig_aug(1,i);
63         double v = Xsig_aug(2,i);
64         double yaw = Xsig_aug(3,i);
65         double yawd = Xsig_aug(4,i);
66         double v_a = Xsig_aug(5,i);
67         double v_b = Xsig_aug(6,i);
68
69         // predicted state values
70         double px_p, py_p;
71
72         // avoid division by zero
73         if (fabs(yawd) > 0.001) {
74             px_p = p_x + v/yawd * ( sin (yaw + yawd*delta_t) - sin(yaw));
75             py_p = p_y + v/yawd * ( cos(yaw) - cos(yaw+yawd*delta_t) );
76         } else {
77             px_p = p_x + v*delta_t*cos(yaw);
78             py_p = p_y + v*delta_t*sin(yaw);
79         }
80
81         double v_p = v;
82         double yaw_p = yaw + yawd*delta_t;
83         double yawd_p = yawd;
84
85         // add noise

```

```

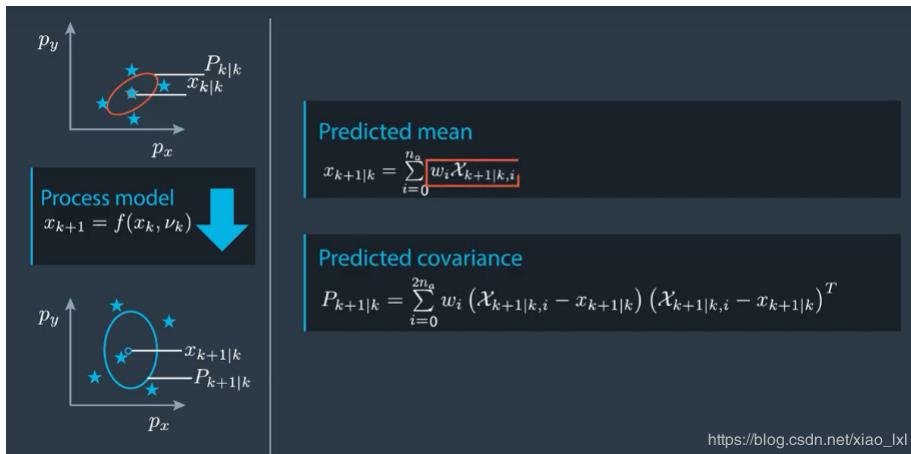
88     v_p = v_p + v_a*delta_t;
89
90     yaw_p = yaw_p + 0.5*v_b*delta_t*delta_t;
91     yawd_p = yawd_p + v_b*delta_t;
92
93     // write predicted sigma point into right column
94     Xsig_pred(0,i) = px_p;
95     Xsig_pred(1,i) = py_p;
96     Xsig_pred(2,i) = v_p;
97     Xsig_pred(3,i) = yaw_p;
98     Xsig_pred(4,i) = yawd_p;
99 }
100
101
102 /**
103 * Student part end
104 */
105
106 // print result
107 std::cout << "Xsig_pred = " << std::endl << Xsig_pred << std::endl;
108
109 // write result
110 *Xsig_out = Xsig_pred;
111 }
```

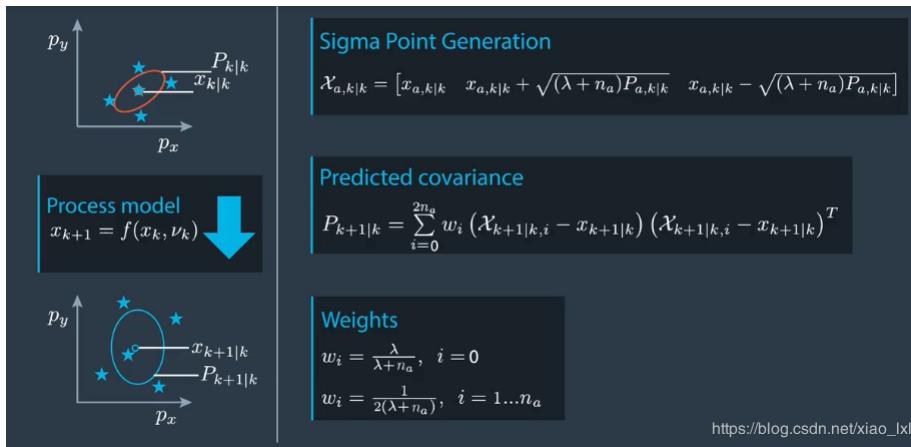
main.cpp

```

1 #include "Dense"
2 #include "ukf.h"
3
4 using Eigen::MatrixXd;
5
6 int main() {
7
8     // Create a UKF instance
9     UKF ukf;
10
11    /**
12     * Programming assignment calls
13     */
14    MatrixXd Xsig_pred = MatrixXd(15, 5);
15    ukf.SigmaPointPrediction(&Xsig_pred);
16
17    return 0;
18 }
```

3.根据预测的sigma点预测状态均值和协方差矩阵





预测均值和协方差c++代码实现

ukf.h

```

1 #ifndef UKF_H
2 #define UKF_H
3
4 #include "Dense"
5
6 class UKF {
7 public:
8     /**
9      * Constructor
10     */
11     UKF();
12
13     /**
14      * Destructor
15     */
16     virtual ~UKF();
17
18     /**
19      * Init Initializes Unscented Kalman filter
20     */
21     void Init();
22
23     /**
24      * Student assignment functions
25     */
26     void GenerateSigmaPoints(Eigen::MatrixXd* Xsig_out);
27     void AugmentedSigmaPoints(Eigen::MatrixXd* Xsig_out);
28     void SigmaPointPrediction(Eigen::MatrixXd* Xsig_out);
29     void PredictMeanAndCovariance(Eigen::VectorXd* x_pred,
30                                     Eigen::MatrixXd* P_pred);
31     void PredictRadarMeasurement(Eigen::VectorXd* z_out,
32                                  Eigen::MatrixXd* S_out);
33     void UpdateState(Eigen::VectorXd* x_out,
34                      Eigen::MatrixXd* P_out);
35 };
36
37 #endif // UKF_H

```

ukf.cpp

```

1 #include <iostream>
2 #include "ukf.h"
3
4 using Eigen::MatrixXd;
5 using Eigen::VectorXd;
6
7 UKF::UKF() {
8     Init();
9 }
10
11 UKF::~UKF() {
12 }
13
14 void UKF::Init() {
15 }
16
17 /**
18  * Programming assignment functions:
19 */
20
21 void UKF::PredictMeanAndCovariance(VectorXd* x_out, MatrixXd* P_out) {
22 }
```

```

28 // set augmented dimension
29 int n_aug = 7;
30
31 // define spreading parameter
32 double lambda = 3 - n_aug;
33
34 // create example matrix with predicted sigma points
35 MatrixXd Xsig_pred = MatrixXd(n_x, 2 * n_aug + 1);
36 Xsig_pred <<
37     5.9374, 6.0640, 5.925, 5.9436, 5.9266, 5.9374, 5.9389, 5.9374, 5.8106, 5.9457, 5.9310, 5.9465, 5.9374, 5.9359, 5.93744,
38     1.48, 1.4436, 1.660, 1.4934, 1.5036, 1.48, 1.4868, 1.48, 1.5271, 1.3104, 1.4787, 1.4674, 1.48, 1.4851, 1.486,
39     2.204, 2.2841, 2.2455, 2.2958, 2.204, 2.204, 2.2395, 2.204, 2.1256, 2.1642, 2.1139, 2.204, 2.1702, 2.2049,
40     0.5367, 0.47338, 0.67809, 0.55455, 0.64364, 0.54337, 0.5367, 0.53851, 0.60017, 0.39546, 0.51900, 0.42991, 0.530188, 0.5367, 0.535048,
41     0.352, 0.29997, 0.46212, 0.37633, 0.4841, 0.41872, 0.352, 0.38744, 0.40562, 0.24347, 0.32926, 0.2214, 0.28687, 0.352, 0.318159;
42
43 // create vector for weights
44 VectorXd weights = VectorXd(2*n_aug+1);
45
46 // create vector for predicted state
47 VectorXd x = VectorXd(n_x);
48
49 // create covariance matrix for prediction
50 MatrixXd P = MatrixXd(n_x, n_x);
51
52
53 /**
54 * Student part begin
55 */
56
57 // set weights
58 weights(0) = lambda / (lambda + n_aug);
59
60 double weight = 0.5 / (lambda + n_aug);
61 for(int i = 1; i < 2 * n_aug + 1; ++i)
62 {
63     weights(i) = weight;
64 }
65
66
67
68 // predict state mean
69 x.fill(0.0);
70 for (int i = 0; i < 2 * n_aug + 1; ++i)
71 { // iterate over sigma points
72     x = x + weights(i) * Xsig_pred.col(i);
73 }
74
75
76
77
78 // predict state covariance matrix
79
80 P.fill(0.0);
81 for (int i = 0; i < 2 * n_aug + 1; ++i)
82 { // iterate over sigma points
83     // state difference
84     VectorXd x_diff = Xsig_pred.col(i) - x;
85
86     //angle normalization
87     while (x_diff(3)> M_PI) x_diff(3)-=2.*M_PI;
88     while (x_diff(3)<-M_PI) x_diff(3)+=2.*M_PI;
89
90     P = P + weights(i) * x_diff * x_diff.transpose() ;
91 }
92
93 /**
94 * Student part end
95 */
96
97 // print result
98 std::cout << "Predicted state" << std::endl;
99 std::cout << x << std::endl;
100 std::cout << "Predicted covariance matrix" << std::endl;
101 std::cout << P << std::endl;
102
103 // write result
104 *x_out = x;
105 *P_out = P;
106
107 }
```

main.cpp

```

1 #include "Dense"
2 #include "ukf.h"
3
4 using Eigen::MatrixXd;
5 using Eigen::VectorXd;
6
7 int main() {
8     // Create a UKF instance
9 }
```

```

12  /**
13  * Programming assignment calls
14  */
15  VectorXd x_pred = VectorXd(5);
16  MatrixXd P_pred = MatrixXd(5, 5);
17  ukf.PredictMeanAndCovariance(&x_pred, &P_pred);
18
19  return 0;
20 }

```

UKF Update

测量预测

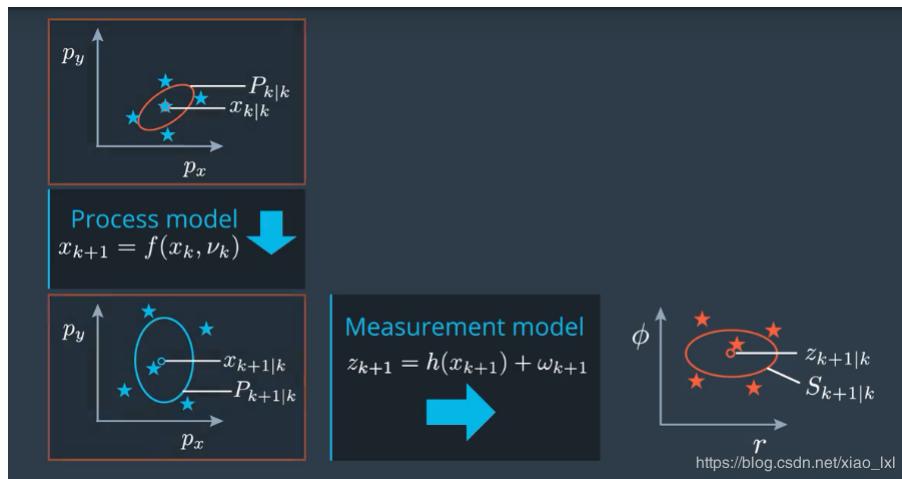
将预测状态转换为测量空间，定义转换的模型叫测量模型（Measurement model）。

测量模型是一个非线性模型，测量噪音具有单纯的累加效果，因此这里我们不需要扩充噪音，有更好的处理方法。

1.首先，将预测到的sigma点转换到测量空间

2.利用这些转换后的点计算预测测量值的均值和协方差矩阵S

同样，转换后的预测sigma点存储为矩阵的列（col）。

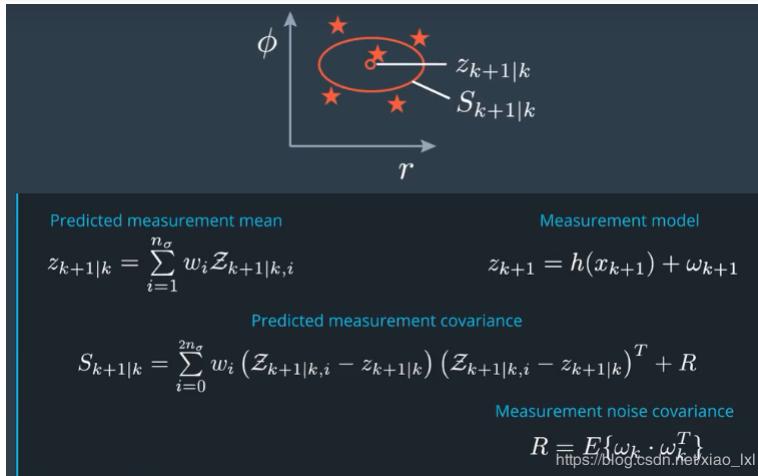


将预测到的sigma点转换到测量空间：

Predicted sigma points $\mathcal{X}_{k+1 k} = [\star \star \star]$ 5×15	State vector $x_{k+1 k} = \begin{bmatrix} p_x \\ p_y \\ v \\ \psi \\ \dot{\psi} \end{bmatrix}$
Measurement model $z_{k+1} = h(x_{k+1}) + \omega_{k+1}$	
Measurement sigma points $\mathcal{Z}_{k+1 k} = [\star \star \star]$ 3×15	Measurement vector $z_{k+1 k} = \begin{bmatrix} \rho \\ \varphi \\ \dot{\rho} \end{bmatrix}$

and 15 columns in our case. https://blog.csdn.net/xiao_lxl

这里，我们不使用扩充噪音矩阵了，故等式中的 w_{k+1} 设为0。



这里，用测量噪音矩阵R来替代扩充噪音矩阵。

测量预测C++代码实现

ukf.h

```

1 | #ifndef UKF_H
2 | #define UKF_H
3 |
4 | #include "Dense"
5 |
6 | class UKF {
7 | public:
8 | /**
9 |  * Constructor
10 | */
11 | UKF();
12 |
13 | /**
14 |  * Destructor
15 | */
16 | virtual ~UKF();
17 |
18 | /**
19 |  * Init Initializes Unscented Kalman filter
20 | */
21 | void Init();
22 |
23 | /**
24 |  * Student assignment functions
25 | */
26 | void GenerateSigmaPoints(Eigen::MatrixXd* Xsig_out);
27 | void AugmentedSigmaPoints(Eigen::MatrixXd* Xsig_out);
28 | void SigmaPointPrediction(Eigen::MatrixXd* Xsig_out);
29 | void PredictMeanAndCovariance(Eigen::VectorXd* x_pred,
30 |                                Eigen::MatrixXd* P_pred);
31 | void PredictRadarMeasurement(Eigen::VectorXd* z_out,
32 |                               Eigen::MatrixXd* S_out);
33 | void UpdateState(Eigen::VectorXd* x_out,
34 |                   Eigen::MatrixXd* P_out);
35 | };
36 |
37 | #endif // UKF_H

```

ukf.cpp

```

1 | #include <iostream>
2 | #include "ukf.h"
3 |
4 | using Eigen::MatrixXd;
5 | using Eigen::VectorXd;
6 |
7 | UKF::UKF() {
8 |     Init();
9 | }
10 |
11 | UKF::~UKF() {
12 | }
13 |
14 | void UKF::Init() {
15 |
16 | }
17 |
18 |
19 | /**
20 |  * Programming assignment functions:
21 | */
22 |

```

```

25 // set state dimension
26 int n_x = 5;
27
28 // set augmented dimension
29 int n_aug = 7;
30
31 // set measurement dimension, radar can measure r, phi, and r_dot
32 int n_z = 3;
33
34 // define spreading parameter
35 double lambda = 3 - n_aug;
36
37 // set vector for weights
38 VectorXd weights = VectorXd(2*n_aug+1);
39 double weight_0 = lambda/(lambda+n_aug);
40 double weight = 0.5/(lambda+n_aug);
41 weights(0) = weight_0;
42
43 for (int i=1; i<2*n_aug+1; ++i) {
44     weights(i) = weight;
45 }
46
47 // radar measurement noise standard deviation radius in m
48 double std_radr = 0.3;
49
50 // radar measurement noise standard deviation angle in rad
51 double std_radphi = 0.0175;
52
53 // radar measurement noise standard deviation radius change in m/s
54 double std_radrd = 0.1;
55
56 // create example matrix with predicted sigma points
57 MatrixXd Xsig_pred = MatrixXd(n_x, 2 * n_aug + 1);
58 Xsig_pred <<
59     5.9374,  6.0640,   5.925,   5.9436,   5.9266,   5.9374,   5.9389,   5.9374,   5.8106,   5.9457,   5.9310,   5.9465,   5.9374,   5.9359,   5.93744,
60     1.48,   1.4436,   1.660,   1.4934,   1.5036,   1.48,   1.4868,   1.48,   1.5271,   1.3104,   1.4787,   1.4674,   1.48,   1.4851,   1.486,
61     2.204,   2.2841,   2.2455,   2.2958,   2.204,   2.204,   2.2395,   2.204,   2.1256,   2.1642,   2.1139,   2.204,   2.204,   2.1702,   2.2049,
62     0.5367,   0.47338,   0.67809,   0.55455,   0.64364,   0.54337,   0.5367,   0.53851,   0.60017,   0.39546,   0.51900,   0.42991,   0.530188,   0.5367,   0.535048,
63     0.352,   0.29997,   0.46212,   0.37633,   0.4841,   0.41872,   0.352,   0.38744,   0.40562,   0.24347,   0.32926,   0.2214,   0.28687,   0.352,   0.318159;
64
65 // create matrix for sigma points in measurement space
66 MatrixXd Zsig = MatrixXd(n_z, 2 * n_aug + 1);
67
68 // mean predicted measurement
69 VectorXd z_pred = VectorXd(n_z);
70
71 // measurement covariance matrix S
72 MatrixXd S = MatrixXd(n_z,n_z);
73
74 /**
75 * Student part begin
76 */
77 // transform sigma points into measurement space
78 for(int i=0; i < 2 * n_aug + 1; ++i)
79 {
80     float p_x = Xsig_pred(0,i);
81     float p_y = Xsig_pred(1,i);
82     float v = Xsig_pred(2,i);
83     float yaw = Xsig_pred(3,i);
84     float yawd = Xsig_pred(4,i);
85
86     float th_2 = sqrt(p_x*p_x + p_y*p_y);
87
88     float rho_z = th_2;
89     float yaw_z = atan2(p_y,p_x);
90     float rhod_z = (p_x*cos(yaw)*v + p_y*sin(yaw)*v) / th_2;
91
92
93     Zsig(0,i) = rho_z;
94     Zsig(1,i) = yaw_z;
95     Zsig(2,i) = rhod_z;
96 }
97
98
99
100 // calculate mean predicted measurement
101 z_pred.fill(0.0);
102 for(int i=0; i < 2 * n_aug + 1; ++i)
103 {
104     z_pred = z_pred + weights(i) *Zsig.col(i);
105 }
106
107
108
109
110 // calculate innovation covariance matrix S
111 S.fill(0.0);
112 for (int i=0; i < 2*n_aug+1; ++i)
113 {
114     VectorXd z_diff = Zsig.col(i) - z_pred;
115
116     // angle normalization
117     while (z_diff(1)> M_PI) z_diff(1)-=2.*M_PI;

```

```

120     S = S + weights(i) * z_diff * z_diff.transpose();
121
122 }
123
124
125 // add measurement noise covariance matrix
126 MatrixXd R = MatrixXd(n_z,n_z);
127 R << std_radr*std_radr, 0, 0,
128     0, std_radphi*std_radphi, 0,
129     0, 0,std_radr*std_radr;
130 S = S + R;
131
132 /**
133 * Student part end
134 */
135
136 // print result
137 std::cout << "z_pred: " << std::endl << z_pred << std::endl;
138 std::cout << "S: " << std::endl << S << std::endl;
139
140 // write result
141 *z_out = z_pred;
142 *S_out = S;
143 }
```

main.cpp

```

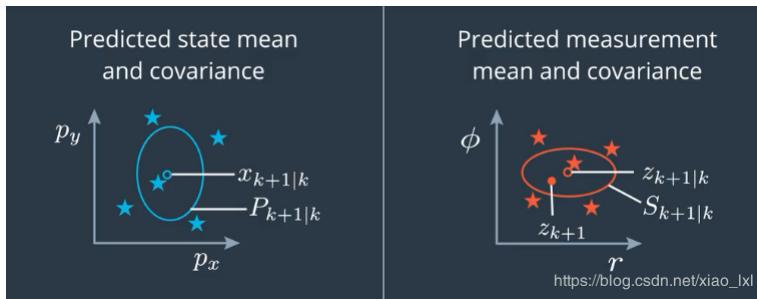
1 #include "Dense"
2 #include "ukf.h"
3
4 using Eigen::MatrixXd;
5 using Eigen::VectorXd;
6
7 int main() {
8
9     // Create a UKF instance
10    UKF ukf;
11
12 /**
13 * Programming assignment calls
14 */
15 VectorXd z_out = VectorXd(3);
16 MatrixXd S_out = MatrixXd(3, 3);
17 ukf.PredictRadarMeasurement(&z_out, &S_out);
18
19 return 0;
20 }
```

预测结果如下：

```

z_pred:
6.12155
0.245993
2.10313
S:
0.0946171 -0.000139447 0.00407016
-0.000139447 0.000617548 -0.000770652
0.00407016 -0.000770652 0.0180917
```

Update之更新状态



Kalman Gain

$$K_{k+1|k} = T_{k+1|k} S_{k+1|k}^{-1}$$

State update

$$x_{k+1|k+1} = x_{k+1|k} + K_{k+1|k}(z_{k+1} - z_{k+1|k})$$

Covariance matrix update

$$P_{k+1|k+1} = P_{k+1|k} - K_{k+1|k} S_{k+1|k} K_{k+1|k}^T$$

New here: Cross-correlation between sigma points in state space and measurement space

$$T_{k+1|k} = \sum_{i=0}^{2n_\sigma} w_i (\mathcal{X}_{k+1|k,i} - x_{k+1|k}) (\mathcal{Z}_{k+1|k,i} - z_{k+1|k})^T$$

https://blog.csdn.net/xiao_lxl

Cross-correlation Matrix

$$T_{k+1|k} = \sum_{i=1}^{n_\sigma} w_i (X_{k+1|k,i} - x_{k+1|k}) (Z_{k+1|k,i} - z_{k+1|k})^T$$

Kalman gain K

$$K_{k+1|k} = T_{k+1|k} S_{k+1|k}^{-1}$$

Update State

$$x_{k+1|k+1} = x_{k+1|k} + K_{k+1|k}(z_{k+1} - z_{k+1|k})$$

Covariance Matrix Update

$$P_{k+1|k+1} = P_{k+1|k} - K_{k+1|k} S_{k+1|k} K_{k+1|k}^T$$

更新状态c++代码实现

```

1 void UKF::UpdateState(VectorXd* x_out, MatrixXd* P_out) {
2
3     // set state dimension
4     int n_x = 5;
5
6     // set augmented dimension
7     int n_aug = 7;
8
9     // set measurement dimension, radar can measure r, phi, and r_dot
10    int n_z = 3;
11
12    // define spreading parameter
13    double lambda = 3 - n_aug;
14
15    // set vector for weights
16    VectorXd weights = VectorXd(2*n_aug+1);
17    double weight_0 = lambda/(lambda+n_aug);
18    double weight = 0.5/(lambda+n_aug);
19    weights(0) = weight_0;
20
21    for (int i=1; i<2*n_aug+1; ++i) {
22        weights(i) = weight;
23    }
24
25    // create example matrix with predicted sigma points in state space
26    MatrixXd Xsig_pred = MatrixXd(n_x, 2 * n_aug + 1);
27    Xsig_pred <<
28        5.9374,  6.0640,  5.925,  5.9436,  5.9266,  5.9374,  5.9389,  5.9374,  5.8106,  5.9457,  5.9310,  5.9465,  5.9374,  5.9359,  5.9374,
29        1.48,   1.4436,  1.660,  1.4934,  1.5036,  1.48,   1.4868,  1.48,   1.5271,  1.3104,  1.4787,  1.4674,  1.48,   1.4851,  1.486,
30        2.204,  2.2841,  2.2455,  2.2958,  2.204,   2.204,  2.2395,  2.204,  2.1256,  2.1642,  2.1139,  2.204,  2.204,  2.1702,  2.2049,
31        0.5367, 0.47338, 0.67809, 0.55455, 0.64364, 0.54337, 0.5367, 0.53851, 0.60017, 0.39546, 0.51900, 0.42991, 0.530188, 0.5367, 0.535048,
32        0.352,  0.29997, 0.46212, 0.37633, 0.4841,  0.41872, 0.352, 0.38744, 0.40562, 0.24347, 0.32926, 0.2214, 0.28687, 0.352, 0.318159;

```

```

35 VectorXd x = VectorXd(n_x);
36 x <<
37     5.93637,
38     1.49035,
39     2.20528,
40     0.536853,
41     0.353577;
42
43 // create example matrix for predicted state covariance
44 MatrixXd P = MatrixXd(n_x,n_x);
45 P <<
46     0.0054342, -0.002405, 0.0034157, -0.0034819, -0.00299378,
47     -0.002405, 0.01084, 0.001492, 0.0098018, 0.00791091,
48     0.0034157, 0.001492, 0.0058012, 0.00077863, 0.000792973,
49     -0.0034819, 0.0098018, 0.00077863, 0.011923, 0.0112491,
50     -0.0029937, 0.0079109, 0.00079297, 0.011249, 0.0126972;
51
52 // create example matrix with sigma points in measurement space
53 MatrixXd Zsig = MatrixXd(n_z, 2 * n_aug + 1);
54 Zsig <<
55     6.1190, 6.2334, 6.1531, 6.1283, 6.1143, 6.1190, 6.1221, 6.1190, 6.0079, 6.0883, 6.1125, 6.1248, 6.1190, 6.1188, 6.12057,
56     0.24428, 0.2337, 0.27316, 0.24616, 0.24846, 0.24428, 0.24530, 0.24428, 0.25700, 0.21692, 0.24433, 0.24193, 0.24428, 0.24515, 0.245239,
57     2.1104, 2.2188, 2.0639, 2.187, 2.0341, 2.1061, 2.1450, 2.1092, 2.0016, 2.129, 2.0346, 2.1651, 2.1145, 2.0786, 2.11295;
58
59 // create example vector for mean predicted measurement
60 VectorXd z_pred = VectorXd(n_z);
61 z_pred <<
62     6.12155,
63     0.245993,
64     2.10313;
65
66 // create example matrix for predicted measurement covariance
67 MatrixXd S = MatrixXd(n_z,n_z);
68 S <<
69     0.0946171, -0.000139448, 0.00407016,
70     -0.000139448, 0.000617548, -0.000770652,
71     0.00407016, -0.000770652, 0.0180917;
72
73 // create example vector for incoming radar measurement
74 VectorXd z = VectorXd(n_z);
75 z <<
76     5.9214, // rho in m
77     0.2187, // phi in rad
78     2.0062; // rho_dot in m/s
79
80 // create matrix for cross correlation Tc
81 MatrixXd Tc = MatrixXd(n_x, n_z);
82
83 /**
84 * Student part begin
85 */
86
87 // calculate cross correlation matrix
88
89 for(int i=0; i < 2 * n_aug + 1; i++)
90 {
91     VectorXd x_diff = Xsig_pred.col(i) - x;
92     // angle normalization
93     while (x_diff(3)> M_PI) x_diff(3)-=2.*M_PI;
94     while (x_diff(3)<-M_PI) x_diff(3)+=2.*M_PI;
95
96     VectorXd z_diff = Zsig.col(i) - z_pred;
97     // angle normalization
98     while (z_diff(1)> M_PI) z_diff(1)-=2.*M_PI;
99     while (z_diff(1)<-M_PI) z_diff(1)+=2.*M_PI;
100
101     Tc = Tc + weights(i)*x_diff *z_diff.transpose();
102 }
103
104 // calculate Kalman gain K;
105 MatrixXd K = Tc * S.inverse();
106
107 // update state mean and covariance matrix
108 VectorXd z_diff = z - z_pred;
109
110 // angle normalization
111 while (z_diff(1)> M_PI) z_diff(1)-=2.*M_PI;
112 while (z_diff(1)<-M_PI) z_diff(1)+=2.*M_PI;
113
114 // update state mean and covariance matrix
115 x = x + K * z_diff;
116 P = P - K*S*K.transpose();
117
118 /**
119 * Student part end
120 */
121
122 // print result
123 std::cout << "Updated state x: " << std::endl << x << std::endl;
124 std::cout << "Updated state covariance P: " << std::endl << P << std::endl;
125
126 // write result
127

```

130 | }

预测结果如下：

Updated state x:

5.92276

1.41823

2.15593

0.489274

0.321338

Updated state covariance P:

0.00361579 -0.000357881 0.00208316 -0.000937196 -0.00071727

-0.000357881 0.00539867 0.00156846 0.00455342 0.00358885

0.00208316 0.00156846 0.00410651 0.00160333 0.00171811

-0.000937196 0.00455342 0.00160333 0.00652634 0.00669436

-0.00071719 0.00358884 0.00171811 0.00669426 0.00881797

噪音参数和评估选择

对于CTRV模型，有两个参数定义了过程噪声：

- σ_a^2 representing longitudinal acceleration noise (you might see this referred to as linear acceleration)
- σ_ψ^2 representing yaw acceleration noise (this is also called angular acceleration)

分别代表了纵向加速度噪声（您可能会看到这称为线性加速度）和偏航加速度噪声（也称为角加速度）

在项目中，这两个值都需要调整。为了得到有效的解决方案，您必须测试不同的值。在视频中，dominik提到使用

$$\sigma_a^2 = 9 \frac{m^2}{s^4}$$

作为跟踪车辆的起点。在UKF项目中，您将跟踪自行车而不是车辆。所以9可能不是一个合适的加速度噪声参数。调谐将涉及：

- | | |
|---|------------|
| 1 | 猜测适当的参数值 |
| 2 | 运行UKF过滤器 |
| 3 | 决定结果是否足够好 |
| 4 | 调整参数并重复该过程 |

如何选择噪音参数和评估选择，这不仅适用于无损卡尔曼滤波，同样适用于贝叶斯滤波器。

如何选择噪音参数

直线加速度噪声参数直观

Let's get some intuition for these noise parameters. The units for the acceleration noise parameter σ_a^2 are $\frac{m^2}{s^4}$. Taking the square root, we get σ_a with units $\frac{m}{s^2}$. So the square root of the acceleration noise parameter has the same units as acceleration: $\frac{m}{s^2}$

The parameter σ_a is the standard deviation of linear acceleration! Remember from the "CTRV Process Noise Vector" lecture that the linear acceleration is being modeled as a Gaussian distribution with mean zero and standard deviation σ_a . In a Gaussian distribution, about 95% of your values are within $2\sigma_a$.

So if you choose $\sigma_a^2 = 9 \frac{m^2}{s^4}$, then you expect the acceleration to be between $-6 \frac{m}{s^2}$ and $+6 \frac{m}{s^2}$ about 95% of the time.

Tuning parameters involves some trial and error. Using your intuition can help you find reasonable initial values.

https://blog.csdn.net/xiao_lxl

偏航加速度噪声参数直观

parameter σ_{ψ}^2 ?

(A) rad/s^2

(B) rad^2/s^2

(C) rad/s^4

(D) rad^2/s^4

https://blog.csdn.net/xiao_lxl

Let's think about what values might be reasonable for the yaw acceleration noise parameter.

Imagine the bicycle is traveling in a circle with a constant yaw rate (angular velocity) of $\frac{\pi \text{ rad}}{8 \text{ s}}$. That means the bicycle would complete a full circle in 16 seconds: $\frac{\pi \text{ rad}}{8 \text{ s}} \cdot 16 \text{ s} = 2\pi$.

That seems reasonable for an average bike rider traveling in a circle with a radius of maybe 16 meters.

The bike rider would have also have a tangential velocity of 6.28 meters per second because $\frac{\pi \text{ rad}}{8 \text{ s}} \cdot 16 \text{ meters} = 6.28 \text{ meters per second}$.

What if the angular acceleration were now $-2\pi \frac{\text{rad}}{\text{s}^2}$ instead of zero? In just one second, the angular velocity would go from $\frac{\pi \text{ rad}}{8 \text{ s}}$ to $-\frac{15\pi \text{ rad}}{8 \text{ s}}$. This comes from $\frac{\pi \text{ rad}}{8 \text{ s}} - 2\pi \frac{\text{rad}}{\text{s}^2} \cdot 1 \text{ s} = -\frac{15\pi \text{ rad}}{8 \text{ s}}$.

The bicycle has been completing a complete circle in 16 seconds. But with such a high angular acceleration, then all of a sudden the bicycle is going around the circle in the opposite direction and only takes about 1.1 second to complete the circle.

From a bicycle, a setting in the range of $\sigma_{\dot{\psi}}^2 = 2\pi \frac{\text{rad}}{\text{s}^2}$ seems too high. In the project, you'll have to experiment with different values to see what works well.

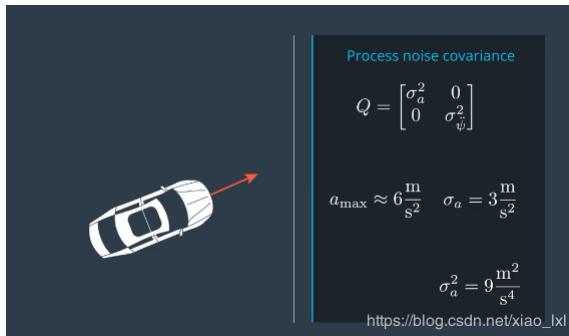
https://blog.csdn.net/xiao_lxl

测量噪声参数

测量噪声参数表示传感器测量的不确定性。一般来说，制造商将在传感器手册中提供这些值。在UKF项目中，不需要调整这些参数。

Where do I get these numbers?		
Process model	Process noise	Process noise covariance
$x_{k+1} = f(x_k, \nu_k)$	$\nu_k = \begin{bmatrix} \nu_{a,k} \\ \nu_{\dot{\psi},k} \end{bmatrix}$	$Q = \begin{bmatrix} \sigma_a^2 & 0 \\ 0 & \sigma_{\dot{\psi}}^2 \end{bmatrix}$
Measurement model	Radar measurement noise	Measurement noise covariance
$z_{k+1} = h(x_{k+1})$ see how precise the sensor is.		
$\begin{bmatrix} \omega_{\rho,k} \\ \omega_{\dot{\rho},k} \end{bmatrix}$ 查看其精确度		
$\begin{bmatrix} \sigma_{\rho}^2 & 0 & 0 \\ 0 & \sigma_{\varphi}^2 & 0 \\ 0 & 0 & \sigma_{\dot{\rho}}^2 \end{bmatrix}$		

https://blog.csdn.net/xiao_lxl



https://blog.csdn.net/xiao_lxl

怎么评估选择

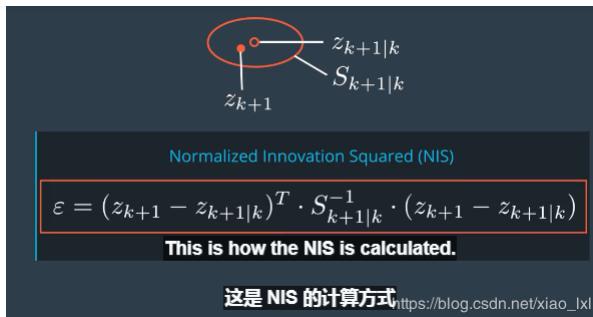
滤波器的一致性检查

NIS, Normalized Innovation Squared ,归一化新息平方

新息是预测测量值和实际测量值之间的差，

归一化是指相对于矩阵S的协方差而言，因此我们这里有矩阵S的逆。

NIS只是一个标量数字，计算非常简单，如下图公式。NIS的值分布符合卡方分布，



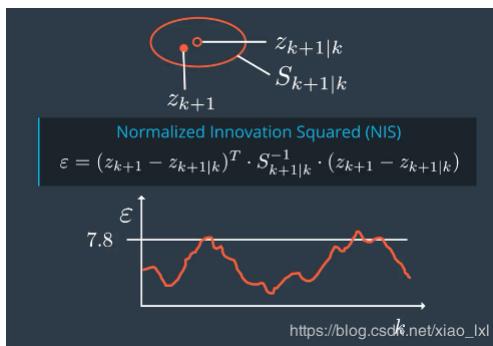
df指自由度，这里指测量空间的维度。

举个例子，假设 第三行

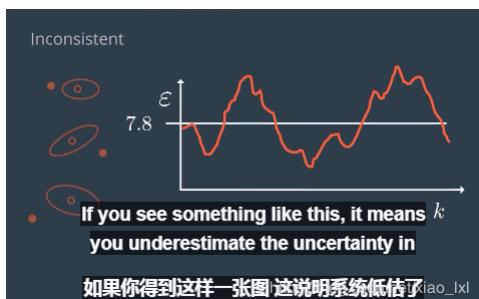


第一列0.95代表在统计意义上，所有情况下有95%的概率，你的NIS会超过0.352；
最后一列0.5代表在统计意义上，所有情况下有5%的概率，你的NIS会超过0.7815；

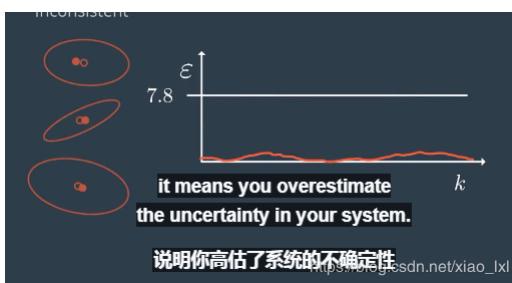
假设本例中，所有情况下有5%的概率，你的NIS会超过0.7815，若是如下图所示，则表明正常，



如果得到这样一张图，则说明系统低估了不确定性。



如果得到下面这样一张图，则说明系统高估了不确定性。你估算的精确度比你



想的要高。

不幸的是，NIS测试不会告诉你错误原因。但至少它提供了一些反馈信息。

例如，在本例中，你可以尝试降低过程噪音，然后再试。

点赞 收藏 分享 ...



xiao_lxl

发布了225篇原创文章 · 获赞318 · 访问量81万+

他的留言板

关注

无人驾驶汽车系统入门（三）——无损卡尔曼滤波，目标追踪，C++

阅读数 1万+

前面两篇文章我们了解了卡尔曼滤波以及扩展卡尔曼滤波在目标追踪的应用，我们在上一篇文章中还具体用Python实... [博文](#) 来自：[AdamShan的博客](#)



想对作者说点什么

春夜喜小雨 17小时前 #1楼

牛逼啊~



无迹卡尔曼滤波（UKF）详解

阅读数 7196

[博文](#) 来自：[qq_41011336的博客](#)

自动驾驶-用无损卡尔曼滤波算法定位

阅读数 283

前边章节介绍了卡尔曼滤波的原理，并用扩展卡尔曼滤波实现匀速直线运动模型的车辆定位。从上章的介绍了解到，... [博文](#) 来自：[m0_37862527的博客](#)

P5_扩展卡尔曼滤波器_udacity无人驾驶

阅读数 422

[博文](#) 来自：[weixin_40215443...的博客](#)

无人驾驶实战（三）——基于卡尔曼滤波（KF）的行人位置及速度的估算

阅读数 145

卡尔曼滤波的背景知识： 在物体跟踪或者预测过程中，我们需要对一些感兴趣的目标状态进行状态估计预测，但... [博文](#) 来自：[qq_40429562的博客](#)

分享靠写代码赚钱的一些门路

阅读数 2万+

作者 mezod，译者 josephchang10如今，通过自己的代码去赚钱变得越来越简单，不过对很多人来说依然还是很难... [博文](#) 来自：[qq_33570092的博客](#)

【安全】Web渗透测试（全流程）

阅读数 1万+

1 信息收集1.1域名、IP、端口域名信息查询：信息可用于后续渗透IP信息查询：确认域名对应IP，确认IP是否真实，... [博文](#) 来自：[qqchaozai的专栏](#)

无人驾驶技术——扩展Kalman滤波（EKF）

阅读数 109

文章目录基本原理运用范围以机器人移动为例，推导过程：融合高斯分布：案例分析：卡尔曼滤波器数据流直观图：... [博文](#) 来自：[xiao_lxl的专栏](#)

我花了一夜用数据结构给女朋友写个H5走迷宫游戏

阅读数 23万+

起因又到深夜了，我按照以往在csdn和公众号写着数据结构！这占用了我大量的时间！我的超越妹妹严重缺乏陪伴而... [博文](#) 来自：[biguai](#)

据说中台凉了？唔，真香

阅读数 1万+

全文长度:2200字阅读时间:8分钟TL;DR(toolongdon'tread)1、业务中台就是流程模板+扩展点2、没法很好抽象就别做... [博文](#) 来自：[u010459192的博客](#)

无人驾驶技术——扩展Kalman滤波(EKF) - xiao_lxl的专栏 - CSDN博客

11-11

无人驾驶技术——初探Kalman滤波器 - xiao_lxl的专栏 - CSDN博客

11-11

Linux文件操作高频使用命令

阅读数 5万+

文章目录0.新建操作：1.查看操作2.删除操作3.复制操作4.移动操作：5.重命名操作：6.解压缩操作0.新建操作：mk... [博文](#) 来自：[不能如期而至的专栏](#)

AdamShan
32篇文章
排名:千里之外

机尾云拉长
59篇文章
排名:千里之外

Howard_eng
15篇文章
排名:千里之外

默_存
34篇文章
排名:千里之外

无人驾驶汽车系统入门(一)——卡尔曼滤波与目标追踪 - ... CSDN博客

11-11

无人驾驶技术——CTRV模型 - xiao_lxl的专栏 - CSDN博客

12-2

为什么程序员在学习编程的时候什么都记不住？

阅读数 4万+

在程序员的职业生涯中，记住所有你接触过的代码是一件不可能的事情！那么我们该如何解决这一问题？作者 |Dylan... [博文](#) 来自：[CSDN资讯](#)

1简介无损卡尔曼滤波又称无迹卡尔曼滤波 (UnscentedKalmanFilter, UKF) , 是无损变换 (UnscentedTransform... 博文 来自: GHU

[无损卡尔曼滤波 - 天才樱木 - CSDN博客](#)

11-24

P6_无损卡尔曼滤波器_udacity无人驾驶 - weixin_402154..._CSDN博客

11-14

[无人驾驶汽车系统入门（一）——卡尔曼滤波与目标追踪](#)

阅读数 4万+

前言：随着深度学习近几年来的突破性进展，无人驾驶汽车也在这些年开始不断向商用化推进。很显然，无人驾驶汽... 博文 来自: AdamShan的博客

[对计算机专业来说学历真的重要吗？](#)

阅读数 18万+

我本科学校是渣渣二本，研究生学校是985，现在毕业五年，校招笔试、面试，社招面试参加了两年了，就我个人的... 博文 来自: 启航

[自动驾驶\(十三\)---无损卡尔曼滤波 - zhouyy858 - CSDN博客](#)

11-11

[无损卡尔曼滤波UKF与多传感器融合 - YoungGy的专栏 - CSDN博客](#)

11-15

[扛住阿里双十一高并发流量，Sentinel是怎么做到的？](#)

阅读数 1万+

Sentinel承接了阿里巴巴近10年的双十一大促流量的核心场景本文介绍阿里开源限流熔断方案Sentinel功能、原理、... 博文 来自: u014714618的专栏

[程序员成长的四个简单技巧，你 get 了吗？](#)

阅读数 2万+

最近拜读了“阿里工程师的自我修养”手册，12位技术专家分享生涯感悟来帮助我们这些菜鸟更好的成长，度过中年危... 博文 来自: 平头哥的技术博文

[自动驾驶-用无损卡尔曼滤波算法定位 - m0_37862527的博客 - CSDN博客](#)

11-11

[无损卡尔曼滤波学习笔记](#)

阅读数 458

作为学习自动驾驶的一部分，在这里把学习无损卡尔曼滤波的过程记录一下，以下内容会有很多都是参考别的博客，... 博文 来自: 申申的博客

[史上最详细的IDEA优雅整合Maven+SSM框架（详细思路+附带源码）](#)

阅读数 9万+

网上很多整合SSM博客文章并不能让初探ssm的同学思路完全的清晰，可以试着关掉整合教程，摇两下头骨，哈一大... 博文 来自: 程序员宜春的博客

[无损卡尔曼滤波UKF与多传感器融合](#)

阅读数 6804

非线性系统状态估计是一大难点。KF (Kalman Filter) 只适用于线性系统。EKF (Extended Kalman Filter) 利用泰... 博文 来自: YoungGy的专栏

[MySQL数据库—SQL汇总](#)

阅读数 4万+

一、准备下文整理常见SQL语句的用法，使用MySQL5.7测试，参考了尚硅谷MySQL教程及用例。用例sql：链接: ht... 博文 来自: Sirm23333

[无人驾驶技术——初探Kalman滤波器](#)

阅读数 81

文章目录高斯分布高斯公式将两个Gaussian相乘计算新的高斯的均值和方差卡尔曼滤波器预测函数一维kalman 实现... 博文 来自: xiao_lxl的专栏

[无人驾驶技术——CTRV模型](#)

阅读数 249

文章目录机器人运动与三角学CTRV模型CTRV差分方程CTRV积分角速度为0时CTRV过程噪声矢量CTRV过程噪声位... 博文 来自: xiao_lxl的专栏

[无人驾驶——2.定位之卡尔曼滤波](#)

阅读数 9

找遍全网，个人认为这篇讲的最好。参考：https://blog.csdn.net/young_gy/article/details/78177291 <http://bilgin.esme...> 博文 来自: weixin_34194379...

[nginx学习，看这一篇就够了：下载、安装。使用：正向代理、反向代理、负载均衡。常用命令和配置文件](#)

阅读数 3万+

文章目录前言一、nginx简介1. 什么是 nginx 和可以做什么事情2.Nginx 作为 web 服务器3. 正向代理4. 反向代理5. 动... 博文 来自: 冯安晨

[无迹卡尔曼滤波器（UKF）](#)

阅读数 255

参考 博文 来自: xiaohu的博客

[史上最全正则表达式语法，文末附常用表达式！](#)

阅读数 1万+

废话少说，直接开始学习！一、元字符元字符是构造正则表达式的一种基本元素。.：匹配除换行符以外的任意字符... 博文 来自: 藏经阁

[无人驾驶技术——雷达Clutter, CFAR,AoA](#)

阅读数 207

文章目录Clutter杂波阈值(Clutter Thresholding)动态阈值(Clutter Thresholding)知识问答CFARCA-CFARClutter雷达... 博文 来自: xiao_lxl的专栏

[大学四年，分享看过的优质书籍](#)

阅读数 3万+

数据结构与算法是在我大学里第一次接触到的，当时学了很多其他安卓、网页之类的，一开始感觉纳闷，数据结构... 博文 来自: 一个不甘平凡的码农

[一文搞懂什么是TCP/IP协议](#)

阅读数 3万+

什么是TCP/IP协议?计算机与网络设备之间如果要相互通信,双方就必须基于相同的方法,比如如何探测到通信目标,由... 博文 来自: petterp的博客

[学习 Java 应该关注哪些网站?](#)

阅读数 2万+

经常有一些读者问我：“二哥，学习 Java 应该关注哪些网站？”，我之前的态度一直是上知乎、上搜索引擎搜一下不... 博文 来自: 沉默王二

[项目中的if else太多了，该怎么重构？](#)

阅读数 4万+

介绍最近跟着公司的大佬开发了一款IM系统，类似QQ和微信哈，就是聊天软件。我们有一部分业务逻辑是这样的if (...

博文

[原创 | 最近程序员频繁被抓，如何避免面向监狱编程？！](#)

阅读数 4901

△Hollis,一个对Coding有着独特追求的人△这是Hollis的第233篇原创分享作者Hollis来源Hollis (ID: hollischuang) 博文 来自: HollisChuang's Blog

[有哪些让程序员受益终生的建议](#)

阅读数 8万+

从业五年多，辗转两个大厂，出过书，创过业，从技术小白成长为基层管理，联合几个业内大牛回答下这个问题，希... 博文 来自: 启航

[《吊打面试官》系列-Redis双写一致性、并发竞争、线程模型](#)

阅读数 7922

你知道的越多，你不知道的越多 点赞再看，养成习惯前言Redis在互联网技术存储方面使用如此广泛，几乎所有的后... 博文 来自: 敦丙

问天下男生，有谁想单身？又有谁想单身一辈子？虽然本人也是单身狗，但是也是一个远大的理想，哈哈，大白天... 博文 来自： weixin_44560813...

YouTube排名第一的励志英文演讲《Dream(梦想)》

I don't know what that dream is that you have, I don't care how disappointing it might have been as y...

阅读数 3万+

博文 来自： 乔治大哥的博客

一文看懂https如何保证数据传输的安全性的

通过漫画的形式由浅入深带你读懂https是如何保证一台主机把数据安全发给另一台主机的对称加密一禅：在每次发送... 博文 来自： weixin_30337157...

阅读数 975

程序员实用工具网站

目录1、搜索引擎2、PPT3、图片操作4、文件共享5、应届生招聘6、程序员面试题库7、办公、开发软件8、高清图... 博文 来自： 不脱发的程序员

阅读数 24万+

花了20分钟，给女朋友们写了一个web版群聊程序

参考博客 [1]https://www.byteslounge.com/tutorials/java-ee-html5-websocket-example

阅读数 7万+

博文

大学四年自学走来，这些私藏的实用工具/学习网站我贡献出来了

大学四年，看课本是不可能一直看课本的了，对于学习，特别是自学，善于搜索网上的一些资源来辅助，还是非常有...

阅读数 18万+

博文

linux系列之常用运维命令整理笔录

本博客记录工作中需要的linux运维命令，大学时候开始接触linux，会一些基本操作，可是都没有整理起来，加上是做...

阅读数 5万+

博文

大学四年，我把私藏的自学「学习网站/实用工具」都贡献出来了

在分享之前，先说说初学者如何学习编程，这个话题想必非常的重要，要学好编程，给你一些学习网站也好、实用工...

阅读数 11万+

博文

中国麻将：世界上最早的区块链项目

中国麻将：世界上最早的区块链项目 最近区块链这个玩意又被市场搞的很是火热，相信大部分人都不太清楚这玩意...

阅读数 7万+

博文

比特币原理详解

一、什么是比特币 比特币是一种电子货币，是一种基于密码学的货币，在2008年11月1日由中本聪发表比特币白皮书...

阅读数 7万+

博文

Python 基础（一）：入门必备知识

Python 入门必备知识，你都掌握了吗？

阅读数 4万+

博文

违法？猝死？你肯定不知道程序员还有这些“高危”操作

全文共2975字，预计学习时长9分钟图源：百度10月24日，一段“996程序员猝死在1024程序员节”的视频在各大IT群...

阅读数 4366

博文

兼职程序员一般可以从什么平台接私活？

这个问题我进行了系统性的总结，以下将进行言简意赅的说明和渠道提供，希望对各位小猿/小媛们有帮助~ 根据我们...

阅读数 11万+

博文

对《Java编程思想》读者的一点建议

《Java 编程思想》这本书在豆瓣的评分高达 9.1 分，但我总觉得有点虚高。记得刚上大学那会，就在某宝上买了一...

阅读数 1684

博文

程序员接私活怎样防止做完了不给钱？

首先跟大家说明一点，我们做 IT 类的外包开发，是非标品开发，所以很有可能在开发过程中会有这样那样的需求修...

阅读数 4万+

博文

《吊打面试官》系列-Redis基础知识

你知道的越多，你不知道的越多 点赞再看，养成习惯 前言 Redis在互联网技术存储方面使用如此广泛，几乎所有的...

阅读数 2万+

博文

Python十大装B语法

Python 是一种代表简单思想的语言，其语法相对简单，很容易上手。不过，如果就此小视 Python 语法的精妙和深邃...

阅读数 14万+

博文

数据库优化 - SQL优化

以实际SQL入手，带你一步一步走上SQL优化之路！

阅读数 3万+

博文

《吊打面试官》系列-缓存雪崩、击穿、穿透

你知道的越多，你不知道的越多 点赞再看，养成习惯 前言 Redis在互联网技术存储方面使用如此广泛，几乎所有的...

阅读数 1万+

博文

腾讯算法面试题：64匹马8个跑道需要多少轮才能选出最快的四匹？

昨天，有网友私信我，说去阿里面试，彻底的被打击到了。问了为什么网上大量使用ThreadLocal的源码都会加上priv...

阅读数 1万+

博文

面试官：你连RESTful都不知道我怎么敢要你？

干货，2019 RESTful最贱实践

阅读数 2万+

博文

为啥国人偏爱Mybatis，而老外喜欢Hibernate/JPA呢？

关于SQL和ORM的争论，永远都不会终止，我也一直在思考这个问题。昨天又跟群里的小伙伴进行了一番讨论，感...

阅读数 1万+

博文

Nginx 原理和架构

Nginx 是一个免费的，开源的，高性能的 HTTP 服务器和反向代理，以及 IMAP / POP3 代理服务器。Nginx 以其高...

阅读数 2万+

博文

致 Python 初学者

欢迎来到“Python进阶”专栏！来到这里的每一位同学，应该大致上学习了很多 Python 的基础知识，正在努力成长的...

阅读数 8万+

博文

springboot-集成WebSockets广播消息

我不喜欢你的大喇叭，天天都把咱们的事情说出去，这样秀，很容易出事情！ ...

阅读数 485

博文

“狗屁不通文章生成器”登顶GitHub热榜，分分钟写出万字形式主义大作

一、垃圾文字生成器介绍 最近在浏览GitHub的时候，发现了这样一个骨骼清奇的雷人项目，而且热度还特别高。项...

阅读数 6万+

博文

程序员：我终于知道post和get的区别

这是一个老生常谈的话题，然而随着不断的学习，对于以前的认识有很多误区，所以还是需要不断地总结的，学而时习...

阅读数 8万+

博文

你知道的越多，你不知道的越多 点赞再看，养成习惯GitHub上已经开源<https://github.com/JavaFamily>，有一线大厂...

博文

程序员把地府后台管理系统做出来了，还有3.0版本！12月7号最新消息：已在开发中有github地址

阅读数 8万+

第一幕：缘起 听说阎王爷要做个生死簿后台管理系统，我们派去了一个程序员..... 996程序员做的梦： 第一场：团...

博文

小白都能看得懂的java虚拟机内存模型

阅读数 1万+

目录 一、虚拟机 二、虚拟机组成 1.栈 栈帧 2.程序计数器 3.方法区 对象组成 4.本地方法栈 5.堆 GC GC案例 一、虚...

博文

面试官如何考察你的思维方式？

阅读数 1万+

1.两种思维方式在求职面试中，经常会考察这种问题：北京有多少量特斯拉汽车？某胡同口的煎饼摊一年能卖出多少...

博文

腾讯“疯狂”开源！

阅读数 2万+

作者 | 马超 责编 | 胡巍巍 出品 | CSDN (ID: CSDNnews) 近日，腾讯自研的万亿级分布式消息中间件TubeMQ正式...

博文

so easy！10行代码写个“狗屁不通”文章生成器

阅读数 4万+

前几天，GitHub 有个开源项目特别火，只要输入标题就可以生成一篇长长的文章。背后实现代码一定很复杂吧，里...

博文

MySQL数据库总结

阅读数 2万+

一、数据库简介 数据库(Database, DB)是按照数据结构来组织，存储和管理数据的仓库。典型特征：数据的结构化...

博文

记一次腾讯面试：进程之间究竟有哪些通信方式？如何通信？---- 告别死记硬背

阅读数 1万+

有一次面试的时候，被问到进程之间有哪些通信方式，不过由于之前没深入思考且整理过，说的并不好。想必大家也...

博文

20行Python代码爬取王者荣耀全英雄皮肤

阅读数 4万+

引言 王者荣耀大家都玩过吧，没玩过的也应该听说过，作为时下最火的手机MOBA游戏，咳咳，好像跑题了。我们...

博文

计算机考研，这样选学校才是正解

阅读数 1万+

写了一篇《启航：对计算机专业来说学历真的重要吗？》，一时间N多同学咨询自身情况要不要考研，眼看有点Hold...

博文

程序设计的5个底层逻辑，决定你能走多快

阅读数 2万+

阿里妹导读：肉眼看计算机是由CPU、内存、显示器这些硬件设备组成，但大部分人从事的是软件开发工作。计算机...

博文

张小龙-年薪近3亿的微信之父，他是如何做到的？

阅读数 3万+

张小龙生于湖南邵东魏家桥镇，家庭主要特点：穷。不仅自己穷，亲戚也都很穷，可以说穷以类聚。爷爷做过铜匠...

博文

西游记团队中如果需要裁掉一个人，会先裁掉谁？

阅读数 1万+

2019年互联网寒冬，大批企业开始裁员，下图是网上流传的一张截图：裁员不可避免，那如何才能做到不管大环境...

博文

iOS Bug 太多，苹果终于坐不住了！

阅读数 2万+

开源的 Android 和闭源的 iOS，作为用户的你，更偏向哪一个呢？整理 | 屠敏 出品 | CSDN (ID: CSDNnews) 内...

博文

程序员一般通过什么途径接私活？

阅读数 2万+

二哥，你好，我想知道一般程序员如何接私活，我也想接，能告诉我一些方法吗？上面是一个读者“烦不烦”问我的...

博文

2020年大前端发展趋势

阅读数 3万+

迅速发展的前端开发，在每一年，都为开发者带来了新的关键词。2019 年已步入尾声，2020 年前端发展的关键词又...

博文

Spring Alibaba Cloud使用Seata实现分布式事务(二)之原理分析

阅读数 659

前言 在上一篇文章Spring Alibaba Cloud使用Seata实现分布式事务,Nacos作为配置中心(一)进行实战演示,这篇文章...

博文

python json java mysql pycharm android linux json格式 c#免安装版反编译工具 c# 深度 递归 c#网页如何调试 c# 添加自定义的属性 c#中去除窗体边框 dll ida修改c# c#实现打印功能 c# 线程结束时执行 c# kb mb 图片 c# 替换第几位字符

©2019 CSDN 皮肤主题: 大白 设计师: blogdevteam



xiao_lxl 博客专家 TA的个人主页 >

原创 225 粉丝 691 获赞 318 评论 300 访问 81万+

等级: 博客 周排名: 898

积分: 9350 总排名: 3230

勋章:   

关注

私信

最新文章

机器学习——KNN

特征检测之特征提取（Detect）

机器学习——K折交叉验证

两个3*3的卷积核替代5*5(三个3*3卷积核替代7*7)分析

机器学习目录整理

分类专栏



归档

2019年12月	3篇
2019年11月	4篇
2019年10月	2篇
2019年9月	10篇
2019年8月	16篇
2019年7月	25篇
2019年6月	10篇
2019年5月	7篇

展开

热门文章

什么是协方差，怎么计算？为什么需要协方差？	阅读数 69846
python中调用 imread 报错： ImportError: cannot import name imread	阅读数 27545
VS2010封装DLL时报错：error LNK2019: 无法解析的外部符号 “_declspec(dllexport)	阅读数 21468
图像预处理——图像分割	阅读数 19424
Python股市数据分析教程——学会它，或可以实现半“智能”炒股 (Part 1)	阅读数 18659

最新评论

无人驾驶技术——无损卡尔曼滤波 (U...
chunyexiaoyu: 牛逼啊~
VS2010 无法定位程序输入点 ...
ffrost: 我的依然解决不了
OpenCV_目标跟踪学习笔记_1...
weixin_41957446: 感谢博主，学习了。
目标检测 配置RefineDe...
qq_44867109: [reply]xiao_lxl[/reply] 弄好了，谢谢
目标检测 Robust Lan...
beckhan001: 代码 https://github.com/qinnzou/Ro bust-Lane-Detection



CSDN学院



CSDN企业招聘

QQ客服 kefu@csdn.net
 客服论坛 [400-660-0108](tel:400-660-0108)

工作时间 8:30-22:00

关于我们 招聘 广告服务 网站地图

京ICP备19004658号 经营性网站备案信息

公安备案号 11010502030143

©1999-2019 北京创新乐知网络技术有限公司

网络110报警服务

北京互联网违法和不良信息举报中心

中国互联网举报中心 家长监护 版权申诉