OpenCV 칼만필터 사용법

- 예측, 추적에 활용되는 알고리즘

KalmanFilter

Mat statePre; //!< predicted state (x'(k)): x(k)=A*x(k-1)+B*u(k)

- 예측값, 수정 이전 상태값, 초기화

Mat statePost; //!< corrected state (x(k)): x(k)=x'(k)+K(k)*(z(k)-H*x'(k))

- 수정값, correct() 함수에 의해 계산됨

Mat transitionMatrix; //!< state transition matrix (A)

- 변환 행렬, 변환 메커니즘 적용, 중요

Mat controlMatrix; //!< control matrix (B) (not used if there is no control)

- 제어 행렬, 초기화 하지 않음

Mat measurementMatrix; //!< measurement matrix (H)

- 측정 행렬, correct() 함수에 측정값 입력시 자동 변화

Mat processNoiseCov; //!< process noise covariance matrix (Q)

- 프로세스 잡음 공분산, 클수록 수정값이 많이 변함, 1e-4

Mat measurementNoiseCov;//!< measurement noise covariance matrix (R)

- 측정 잡음 공분산, 작을수록 수정값이 많이 변함, 1e-1

- 100ms 정도의 지연 시간에서 둘다 1e-3 정도가 적당한 듯

Mat errorCovPre; //!< priori error estimate covariance matrix (P'(k)): P'(k)=A*P(k-1)*At + Q)*/

- 초기화 안함, 이전 오차 공분산

Mat gain; //!< Kalman gain matrix (K(k)): K(k)=P'(k)*Ht*inv(H*P'(k)*Ht+R)

- 초기화 안함

Mat errorCovPost; //!< posteriori error estimate covariance matrix (P(k)): P(k)=(I-K(k)*H)*P'(k)

- 이후 오차 공분산, 초기화 안함

predict();//예측 : 다음 시간단계에 대한 예측, 결과는 statePre에 저장

correct(Mat measurement);//교정 : 새로운 측정치와 통합, 결과는 statePost에 저장

```
//=========================================
// 칼만필터 를 이용한 마우스 추적(트래킹, 움직임 예측) 예제, 소스코드
/*
        // plot points
#define drawCross( center, color, d )                                 \
line( img, Point( center.x - d, center.y - d ),                \
Point( center.x + d, center.y + d ), color, 2, CV_AA, 0); \
line( img, Point( center.x + d, center.y - d ),                \
Point( center.x - d, center.y + d ), color, 2, CV_AA, 0 )
*/
struct mouse_info_struct { int x,y; };
struct mouse_info_struct mouse_info = {-1,-1}, last_mouse;

vector<Point> mousev,kalmanv;

void on_mouse(int event, int x, int y, int flags, void* param) {
   //if (event == CV_EVENT_LBUTTONUP)
   {
      last_mouse = mouse_info;
      mouse_info.x = x;
      mouse_info.y = y;

//       cout << "got mouse " << x <<","<< y <<endl;
   }
}

// 화살표 그리기
//http://tmjfzy.blog.163.com/blog/static/6644702520122251010177 94/
void drawArrow(cv::Mat& img, cv::Point pStart, cv::Point pEnd,  cv::Scalar color, int
thickness=1, int lineType=8)//, int len=10, int alpha=20)
{
   const double PI = 3.1415926;
   Point arrow;
   int alpha = 20;//화살촉 벌어지 각도
   int len = sqrt( pow((double)(pStart.y - pEnd.y),2) + pow((double)(pStart.x -
pEnd.x), 2) );
```

```
    int tip_len = len/10;//화살촉 길이

    //
    double angle = atan2((double)(pStart.y - pEnd.y), (double)(pStart.x - pEnd.x));
    line(img, pStart, pEnd, color, thickness, lineType);

    //
    arrow.x = pEnd.x + tip_len * cos(angle + PI * alpha / 180);
    arrow.y = pEnd.y + tip_len * sin(angle + PI * alpha / 180);
    line(img, pEnd, arrow, color, thickness, lineType);
    arrow.x = pEnd.x + tip_len * cos(angle - PI * alpha / 180);
    arrow.y = pEnd.y + tip_len * sin(angle - PI * alpha / 180);
    line(img, pEnd, arrow, color, thickness, lineType);
}

//칼만필터 테스트 - 마우스 따라가기
//http://www.morethantechnical.com/2011/06/17/simple-kalman-filter-for-tracking-using-opencv-2-2-w-code/
//http://morethantechnical.googlecode.com/svn/trunk/mouse_kalman/main.cpp
int KalmanTest_Mouse ()
{
    Mat img(500, 500, CV_8UC3);

    KalmanFilter KF(4, 2, 0);//상태4, 측정2

    Mat_<float> state(4, 1); //상태값 (x, y, Vx, Vy)

    Mat processNoise(4, 1, CV_32F);//노이즈 상태값4


    Mat_<float> measurement(2,1); //측정2
    measurement.setTo(Scalar(0));

    char code = (char)-1;

    namedWindow("mouse kalman");
    setMouseCallback("mouse kalman", on_mouse, 0);

    CString str;
    for(;;)
    {
        if (mouse_info.x < 0 || mouse_info.y < 0) {//이상치 거르기
            imshow("mouse kalman", img);
```

```
            waitKey(30);
            continue;
        }

        //초기화

        //초기 상태
        KF.statePre.at<float>(0) = mouse_info.x;
        KF.statePre.at<float>(1) = mouse_info.y;
        KF.statePre.at<float>(2) = 0;
        KF.statePre.at<float>(3) = 0;

        //전이 행렬
        KF.transitionMatrix = *(Mat_<float>(4, 4) << 1,0,0,0,   0,1,0,0,  0,0,1,0,  0,0,0,1);

        //단위행렬로 초기화


        //측정행렬
        setIdentity(KF.measurementMatrix);
        //프로세스 잡음 공분산, 클수록 수정값이 많이 변함
        setIdentity(KF.processNoiseCov, Scalar::all(1e-3));//1e-4, 따라오는 속도,클수록
빠름
        //측정 잡음 공분산, 작을수록 수정값이 많이 변함
        setIdentity(KF.measurementNoiseCov, Scalar::all(1e-2 ));//1e-1, 따라오는 속도,
작을수록 빠름
        //사후 에러 공분산
        setIdentity(KF.errorCovPost, Scalar::all( 0.1 ) );//.1

        str.Format(_T("(%.2f, %.2f, %.2f, %.2f)"), KF.processNoiseCov.at<float>(0),
KF.processNoiseCov.at<float>(1), KF.processNoiseCov.at<float>(2),
KF.processNoiseCov.at<float>(3)  );
        //PutText(img, str,  Point(1, 60), 0.7, CV_RGB(0,255,255), 1);

        mousev.clear();        kalmanv.clear();
        //randn(KF.statePost, Scalar::all(0), Scalar::all(0.1));

        for(;;)
        {
```

```
//          Point statePt(state(0),state(1));

        Mat prediction = KF.predict();//예측 : 다음 시간단계에 대한 예측
        Point predictPt(prediction.at<float>(0), prediction.at<float>(1));

        //측정값, 마우스의 실제 위치
        measurement(0) = mouse_info.x;
        measurement(1) = mouse_info.y;

        Point measPt(measurement(0), measurement(1));
        mousev.push_back(measPt);//마우스 배열에 추가
        // generate measurement
        //measurement += KF.measurementMatrix*state;

        Mat estimated = KF.correct(measurement);//교정 : 새로운 측정치와 통합
        Point statePt(estimated.at<float>(0),estimated.at<float>(1));
        kalmanv.push_back(statePt);//교정된 예측 값

        //그리기
        img = Scalar::all(0);//지우기
        //Scalar color = Scalar(255,0,255);


        drawCross( measPt, Scalar(0,0,255), 5 );
        drawCross( statePt, Scalar(255,255,255), 5 );
        //drawCross( predictPt, Scalar(0,255,0), 3 );
        //line( img, statePt, measPt, Scalar(0,0,255), 3, CV_AA, 0 );
        //line( img, statePt, predictPt, Scalar(0,255,255), 3, CV_AA, 0 );

        for (int i = 0; i < (int)mousev.size()-1; i++) {
            line(img, mousev[i], mousev[i+1], Scalar(255,255,0), 1);
        }
        for (int i = 0; i < (int)kalmanv.size()-1; i++) {
            line(img, kalmanv[i], kalmanv[i+1], Scalar(0,255,0), 1);
        }


        int cnt = mousev.size()-1;
        //변화 방향 화살표 그리기
```

```
        Point ptDiff = Point( (KF.statePost.at<float>(0)-KF.statePre.at<float>(0)),
KF.statePost.at<float>(1)-KF.statePre.at<float>(1)) * 3;//변화량, 작아서 3곱함
        if( cnt>1)drawArrow(img, mousev[cnt], mousev[cnt]+ptDiff, Scalar(255,0,255)
);

        //randn( processNoise, Scalar(0),
Scalar::all(sqrt(KF.processNoiseCov.at<float>(0, 0))));
        //state = KF.transitionMatrix*state + processNoise;

        str.Format(_T("(%.2f, %.2f, %.2f, %.2f)"), KF.statePost.at<float>(0)-
KF.statePre.at<float>(0), KF.statePost.at<float>(1)-KF.statePre.at<float>(1),
KF.statePost.at<float>(2)-KF.statePre.at<float>(2), KF.statePost.at<float>(3)-
KF.statePre.at<float>(3)  );
        PutText(img, str,  Point(1, 60), 0.7, CV_RGB(0,255,255), 1);

        if( cnt>1){
        str.Format(_T("(%d,%d)"),    kalmanv[cnt].x - kalmanv[cnt-1].x ,
kalmanv[cnt].y- kalmanv[cnt-1].y, 0, 0 );
        PutText(img, str,  Point(1, 80), 0.7, CV_RGB(0,255,255), 1);
        }

        imshow( "mouse kalman", img );
        code = (char)waitKey(100);
        if( code > 0 )
           break;
        if( cnt > 100) {mousev.clear();        kalmanv.clear();}

      }
      if( code == 27 || code == 'q' || code == 'Q' )
         break;
   }

   return 0;
}
```