객체 추적 하드웨어를 위한 Kalman-like 필터

*전다현, 박인철
한국과학기술원 전기 및 전자공학과
e-mail : dhjeon.ics@gmail.com

Hardware Friendly Kalman-like Filter for the Object Tracking

*Dahyun Jeon, In-Cheol Park
Department of Electrical Engineering
Korea Advanced Institute of Science and Technology

Abstract

By adding another register, physical model of a kalman filter can be changed to a linear approximation. Kalman filter is inappropriate for hardware implementation because of the complex matrix computation. In this paper, novel estimation algorithm for object tracking is proposed to simplify the computation complexity. By using extra delay cell, proposed algorithm has able to achieve high accuracy without any matrix inversion.

I. 서론

불안정한 측정 데이터를 보다 정확하게 보정하기 위해서 다양한 디지털 필터가 사용된다. 필터에는 Low pass filter, high pass filter, band pass filter 같이 일정한 환경 속에서 동작하는 필터가 있냐 하면 역동적인 환경에서 동작하는 adaptive filter가 있다.

Adaptive filter은 동작하는 환경에 따라 필터 계수가 바뀌며 필터 특성이 변화하게 된다. Adaptive filter의 종류로는 least mean squares와 recursive least squares 등이 있다. Kalman filter은 adaptive filter의 한 종류로서 1960년도에 칼만에 의해 발표 된 이후 칼만 필터는 물체 추적, 네비게이션, 자동 비행기 등 다양한 분야에서 사용 되고 있다.

Kalman filter의 응용에 대해서는 무수히 많은 연구가 진행 되었으며 현재도 많은 연구가 진행되고 있다. 하지만 Kalman filter는 많은 행렬 연산을 필요로 하기 때문에 하드웨어 설계에는 적합하지 않다. 특히 Kalman filter에는 복잡한 행렬의 역행렬을 구하는 작업을 필요로 한다.

본 논문에서는 하드웨어 설계에 적합한 새로운 추적 알고리즘을 제시하였다. 이는 Kalman filter에 비해 추가 레지스터를 필요로 하지만 훨씬 단순한 계산을 요구한다. 본 논문에서 제시한 알고리즘은 하드웨어 설계에 더욱 적합하며 작동 환경에 따른 다른 물리적 모델을 요구하는 Kalman filter에 반해 작동 환경과 독립적인 물리적 모델을 가지고 있다는 장점을 가지고 있다. 컴퓨터 시뮬레이션 결과 본 논문에서 제시한 알고리즘이 기존의 알고리즘에 비해 속도를 10배 이상 향상 시켰다.

II. 본론

2.1 Object Tracking

객체 추적은 비디오 데이터에서 실시간으로 원하는

객체를 추적하는 이미지 프로세싱 기술 중 하나이다. 객체 추적은 실시간으로 들어오는 frame data에서부터 대상 객체의 위치, 즉 (x, y)좌표를 추출해 내는 것이다. 객체 추적은 일반적으로 객체의 위치를 측정하는 단계와 객체의 위치를 예측하는 단계 그리고 최적 위치를 추적하는 단계로 나뉜다.

2.2 Kalman filter

칼만 필터는 1960년 칼만의 논문에서 제안 된 추적 알고리즘이다. 칼만 필터는 [표 1]과 같이 예측 단계와 수정 단계로 분류 되는 두 단계로 이루어져있다. 예측 단계에서는 물리적 모델을 통한 다음 위치의 예측 값, $\hat{x}_k^-$와 예측 결과의 분산행렬P_k^-를 추적 값 $\hat{x}_k$과 그의 분산 P_k를 통하여 계산하게 된다. 이 때 사용되는 물리적 모델은 예측 대상에 대한 선행 지식이 필요한 것으로 이를 통한 예측의 기댓값이 실제 값이 되도록 한다. 이 때 물리적 모델은 환경의 불완전성 때문에 process noise Q를 동반한다고 가정한다. 수정 단계에서는 앞 단계에서 구한 예측 결과 $\hat{x}_k^-$와 측정결과z_k를 Kalman gain, K_k,를 통하여 weighting하여 최종 추적 값을 출력한다. 이 때 K_k는 추적 결과의 분산이 최소화 되도록 하여 주는 값이다.

Kalman Filter	
Prediction stage	Correction stage
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$ $P_k^- = AP_{k-1}A^T + Q$	$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$ $\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$ $P_k = (I - K_k H)P_k^-$

표 1 Two stages of Kalman filter

Kalman filter의 각 단계에서는 많은 행렬 연산을 필요로 하며 수정 단계에서 K_k을 구하기 위해서 역행렬 연산을 수행하여야 한다. 이 때 K_k는 일반적으로 block diagonally dominant한 행렬이 된다고 가정 되어 역행렬을 구하기 힘들다 [2], [3]. 이런 행렬 연산으로 인하여 Kalman filter은 하드웨어 친화적이지 않다.

2.3 Proposed Algorithm

칼만 필터가 복잡한 행렬식을 필요로 하는 이유는 변수간의 correlation이 존재하기 때문이다. Correlation이 존재 하지 않는다면 행렬은 대각행렬이 되기 때문

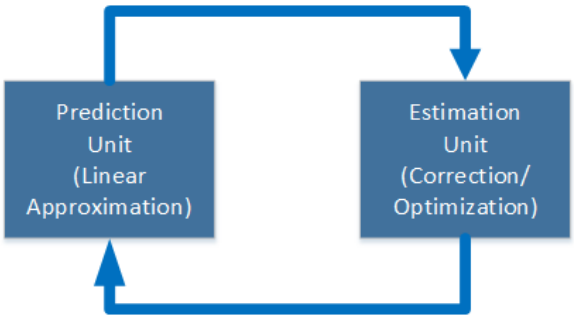

그림 1 Two stages of proposed algorithm

에 행렬 계산의 복잡도는 스칼라 계산과 동일 해 진다.

본 논문에서 제시 된 알고리즘은 그림 와 같은 예측 단계와 수정 단계로 이루어져 있다. 예측 단계는 객체의 경로의 2차 근사한 식으로 과거 2 프레임에서의 위치 정보를 저장하고 있어야 한다. 시간 k일 때의 변위 $\hat{x}_k^-$은 Taylor 근사로 $\hat{x}_k^- \approx \hat{x}_k + \dot{\hat{x}}_{k-1} + 0.5\ddot{\hat{x}}_{k-2}$로 근사된다. 여기서 $\dot{\hat{x}}_k$, $\ddot{\hat{x}}_k$은 각각 $\dot{\hat{x}}_k = x_k - x_{k-1}$ $\ddot{\hat{x}}_k = x_k - 2x_{k-1} + x_{k-2}$로 근사 될 수 있다. 여기서 이들의 분산은 다음 식으로 구할 수 있다.

$$Var(\dot{x}[n]) = Var(x[n]) + Var(x[n-1])$$
$$Var(\ddot{x}[n]) = Var(x[n]) + 2^2\,Var(x[n-1]) + Var(x[n-2])$$

두 번째 단계에서 첫 번째 단계의 예측 결과 $\hat{x}_k^-$와 측정 결과 m_k로 보다 정확한 추적 값 $\hat{x}_k$을 구한다. 최종 추적 값 $\hat{x}_k$은 $\hat{x}_k^-$와 m_k의 linear combination으로 다음과 같이 표현할 수 있다.

$$a\hat{x}_k^- + bm_k$$

여기서 $\hat{x}_k$의 기댓값 $E[\hat{x}_k]$은 다음과 같다.

$$E[a\hat{x}_k^- + bm_k] = aE[\hat{x}_k^-] + bE[m_k] \qquad (1)$$

이 때 $\hat{x}_k^-$와 m_k의 기댓값이 unbiased 됐다면 $\hat{x}_k$이 unbiased되기 위해서는 $a+b=1$이 되어야 한다. 이 때 $\hat{x}_k$의 분산은 아래와 같이 구할 수 있다.

$$Var(\hat{x}_k) = a^2\,Var(\hat{x}_k^-) + b^2\,Var(m_k)$$

$\hat{x}_k$의 분산이 가장 최소화 되게 하는 a와 b를 구하면 $\hat{x}_k$는 다음과 같이 표현 된다.

$$\hat{x}_k = \frac{Var(m_k)}{Var(\hat{x}_k^-) + Var(m_k)}\hat{x}_k^- + \frac{Var(\hat{x}_k^-)}{Var(\hat{x}_k^-) + Var(m_k)}m_k$$

이는 $\hat{x}_k^-$와 m_k를 분산의 크기에 따라 내분한 지점

이 $\hat{x}_k$가 되게 하는 지점이 된다.

예측 식을 경로의 2차 근사로 바꿈으로써 현재 변수와 과거 변수와의 correlation만 존재하고 변수간의 correlation은 존재하지 않게 된다. 이때 현재 변수의 variance는 식 로 인해서 계산 할 수 있다. 이 식에서 계산은 단순 scalar 계산이 되며 하드웨어 친화적이다.

Ⅲ. 구현

3.1 시뮬레이션 환경

모든 시뮬레이션은 2012 Mac Pro Server 상에서 진행 되었다. 3.2 GHz Quad Core Xeon W3565 (Nehalem) processor를 가지고 있으며 코어 당 256k의 level 2 cache를 가지고 있고 8 MB level 3 cache를 가지고 있다.

Kalman filter은 OpenCV에서 받은 open source를 수정하여 사용하였다. Measurement는 30으로 설정하였고 process noise는 실험적으로 수정하여서 최적의 값을 찾았으며 5로 설정하였다. Proposed algorithm도

실험은 어떤 지점에서 시작하여 직선, 2차 함수, V자 등 다양한 모양으로 움직이는 객체를 추적하여 그 때 각 measurement, Kalman filter, proposed algorithm으로 구한 estimate의 root mean square (RMS)와 Kalman filter와 proposed algorithm을 실행하는데 걸리는 시스템 속도를 측정하였다.

3.2 시뮬레이션 결과

정확성을 알아보기 위해서 측정 데이터, 칼만 필터, proposed algorith의 average RMS를 측정하여 보았다. [표 2]에 나와 있듯이 proposed algorithm은 Kalman filter보다 약 15% 향상 된 정확성을 나타내는 것을 확인 할 수 있다. 그림 2는 실제 값이 시간에 따라 위와 같이 바뀔 때 각각 Kalman filter로 구한 위치와 proposed algorithm으로 바꾼 위치를 나타내는 그림이다. 여기서 볼 수 있듯이 Kalman filter보다 true value에 가깝다는 것을 볼 수 있다.

제안 된 알고리즘은 속도 측면에서 기존의 Kalman filter보다 좋은 성능을 보여주었다. 속도는 sys/time.h library의 gettimeofday() 함수를 통하여서 측정하였다. gettimeofday()함수는 시간을 millisecond단위로 반환하는 함수로서 프로그램 시작과 끝의 시간을 측정함으로서 속도를 알 수 있다. 속도를 측정해본 결과 그림 3과 같이 Kalman filter에 비해 10배 정도의 속도 증가를 나타냄을 볼 수 있었다.

정확성과 속도 면에서 proposed algorithm이 최적의 알고리즘임을 볼 수 있다.

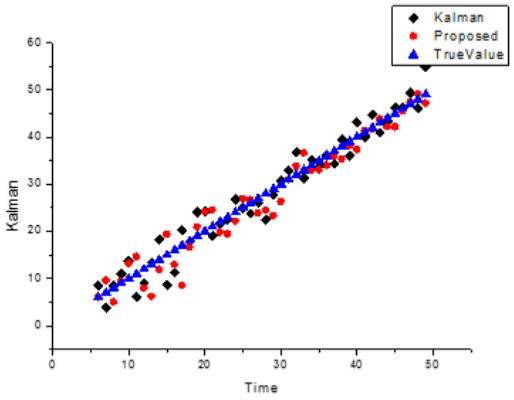

그림 2 Kalman filter와 Proposed algorithm

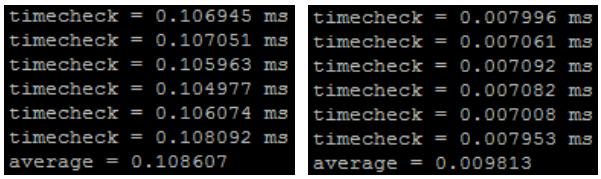

그림 3 Computing time of Kalman filter(left) and proposed algorithm(right)

Average RMS		
Measurement	Kalman filter	Proposed
15.48	8.02	6.81

표 4 Average root mean square of each algorithm

Ⅳ. 결론 및 향후 연구 방향

과거 예측 결과를 저장해두고 사용함으로 칼만 필터보다 효율적인 알고리즘을 제안하였다. 이는 RMS에서 칼만 필터보다 좋은 성능을 냈다. 또한 칼만 필터는 무거운 행렬 연산을 필요로 하는 반면 제안 된 알고리즘은 행렬 연산을 필요로 하지 않는다. 따라서 CPU simulation에서 더 좋은 속도를 낼 수 있었다. 뿐만 아니라 hardware로 설계하기에 더 적합한 구조이므로 low-power을 필요로 하는 환경에서 dedicated hardware을 제작함으로 더욱 효율적으로 객체 추적을 할 수 있을 것으로 기대된다.

앞으로 제안 된 알고리즘을 통하여서 하드웨어를 설계하여 실제 객체 추적 어플리케이션에 구현해 볼 것이다. 이를 통하여 low-power object tracking system을 구현할 계획이다. 또한 non-linearity가 높은 환경에서도 동작할 수 있는지 검증이 필요하며 이에 따른 연구가 필요하다.

Acknowledgement

이 논문은 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2010-0028680).

참고문헌

[1] Welch, G., Bishop, G., 2001. *"An introduction to the Kalman filter."* ACM Siggraph course 8. Annual Conference of. Computer, Graphics, and Interactive Techniques.

[2] Riedel, K.S., 1993: *"Block diagonally dominant positive denite approximate filters and smoothers."* Automatica, 29, 779-783.

[3] P. Barooah, W. M. Russell, and J. Hespanha, *"Approximate distributed Kalman filtering for cooperative multi-agent localization,"* in Proc. Int. Conf. Distrib. Comput. Sens. Netw. (DCOSS '10), Jun. 2010, pp.102 - 115.

[4] B. Farhang-Boroujeny. *"Adaptive Filters: Theory and Applications."* Willey, 1998.

[5] http://opencv.org