제33회 한국정보처리학회 춘계학술발표대회 논문집 제17권 제1호 (2010. 4)

CamShift 알고리즘을 사용한 레이저 포인터 추적

안호영*, 박종승*, 최순필**
*인천대학교 컴퓨터공학과
**(주)초이스테크놀로지
e-mail: tetronecl@incheon.ac.kr

Laser Pointer Tracking Using CamShift Algorithm

Ho-Young Ahn*, Jong-Seung Park*, Soon-Pil Choi**
*Dept. of Computer Science and Engineering, University of Incheon
**Chois Technology Co., Ltd

요 약

레이저 포인터를 검출하는 과정은 포인터의 위치를 검출하는 과정과 입력된 레이저 포인터의 좌표를 모니터의 좌표로 변환하는 과정으로 나눌 수 있다. 레이저 포인터의 추적에 있어서 다변하는 환경의 영향으로 강건성의 확보가 어렵다. 기존의 추적 방식인 Mean-Shift 알고리즘의 경우에는 계산량이 많아서 실시간으로 입력되는 동영상에는 부적합하다. 반면에 CamShift 알고리즘은 빠른 수행이 가능하여 비디오 영상 및 실시간 영상에 적용하기에 적합하고 배경 변화의 영향을 적게 받는다. 또한 검출하려는 색과 같은 색에 의해서 간섭 받는 현상을 방지할 수 있다. 배경이 복잡한 형태이거나 배경이 동적으로 움직일 때에도 강건한 결과를 얻을 수 있다. 제안된 알고리즘을 실환경에 적용한 결과 검출하고자 하는 물체가 예측 영역을 넘나들거나 또는 화면으로부터 지나치게 멀어지거나 가까워져서 상대적인 크기가 변화할 수 있는 불확실한 변화에도 안정적으로 반응함을 알 수 있었다.

1. 서론

최근 비디오 영상으로부터 실시간으로 물체를 추적하는 기술에 대한 필요성이 증가하고 있고 많은 분야에서 관련 연구가 진행되고 있다. 물체를 추적하는데 있어서 카메라가 움직이거나 잡음이 생길 수 있고 또한 물체의 추적이 실패하는 등의 다양한 상황에 노출될 수 있다. 또한 실시간으로 처리할 수 있을 정도의 적은 계산량으로 처리될 수 있어야 한다. 이러한 요구사항을 동시에 만족하는 알고리즘을 제시하는 일은 매우 어려운 것으로 이를 해결하기 위한 많은 노력들이 있었다[1][2].

특히 무선으로 PC를 제어하는 기술은 많은 사람들의 관심이 되었다. 레이저 포인터로 마우스 기능을 대신하려는 노력이 있었다. 그러나 레이저를 이용한 추적은 주변 화면에 영향을 많이 받기 때문에 사용하기에 어려움이 존재하였고 레이저를 비추는 순간과 컴퓨터가 레이저 포인터를 인식하는 순간의 동기화가 맞지 않아서 어려움이 있었다[3].

최근에 흔히 사용되는 카메라 영상으로부터의 포인터 추적 방식으로는 Mean-Shift 알고리즘[4], CamShift 알고리즘[5], ABCShift 알고리즘[6] 등이 있다. 이 알고리즘 들은 단순한 포인트뿐만 아니라 얼굴 추적, 손 추적, 물체 추적 등 다양한 용도로 사용된다. 본 논문에서는 레이저 포인터의 추적 방식을 CamShift 알고리즘을 기반으로 추적하는 방법을 사용하였다. 제안한 레이저 포인터 추적 방법을 구현하고 다양한 환경에서 테스트하였다.

2. 기존의 연구

기존의 포인트 추적에 있어서 주된 관심사는 포인트를 인식하는 방법과 입력된 레이저 포인터의 좌표에서 모니터의 좌표로 변환하는 방법이다. 전체적인 과정을 확인해 보면 다음의 4단계로 생각해볼 수 있다.

첫 번째 단계는 모니터의 좌표와 입력받은 영상의 좌표 사이의 관계를 측정하는 Calibration 단계로 시스템을 시작하기 전에 한번만 수행하면 된다. 두 번째 단계는 포인트를 추적하는 단계이다. 입력된 빛의 빈도를 바탕으로 포인트를 추적한다. 세 번째 단계는 얻어진 포인트의 위치를 모니터의 위치로 변환하는 단계이다. 마지막 단계는 얻어진 모니터의 위치를 마우스의 입력 값으로 전송한다[3].

포인트를 추적하는 방법은 일반적으로 주어진 문턱치(threshold)보다 강한 색상의 픽셀들 중에서 포인트 위치를 조사하는 접근을 사용한다. 이 경우에는 모든 픽셀에 대해서 비교연산을 수행하여야 하고 비교 연산이 많아질수록 연산 속도가 증가하게 되었다. 또한 영상을 얻는 과정에서 잡음(noise)이 발생하므로 밝기값이 비정상적으로 높은 픽셀들이 다수 존재하게 된다.

Mean-Shift 알고리즘의 경우에는 주어진 영역의 전체를 탐색하고 물체에 대한 존재할 확률이 높은 위치에 대한 수렴 여부를 반복적으로 확인하여 결과를 얻는 방법이다[7]. Mean Shift 벡터는 Bhattacharrya 계수가 최대가 되는 지점을 가리키게 되고 반복적인 계산으로 확률밀도 함수의 경사도가 0이 되는 국지 최대점으로 수렴하게 된다. Bhattacharrya 계수는 추적하려는 물체가 존재하는 확

률을 나타내며 물체를 추적하는 과정은 Bhattacharrya 계수가 최대가 되는 위치를 추적하는 것과 동일하다.

$$\hat{y_1} = \frac{\sum_{i=1}^{n} x_i w_i g\left(\left\| \frac{\hat{y_0} - x_i}{h} \right\|^2 \right)}{\sum_{i=1}^{n} w_i g\left(\left\| \frac{\hat{y_0} - x_i}{h} \right\|^2 \right)} \quad (1)$$

수식 (1)에서 y 는 추적하는 동적 물체의 다음 위치, n은 픽셀 수, x는 픽셀의 좌표, w는 색상 가중치이다. 이는 계산 량이 많아서 실시간으로 입력되는 영상의 경우에는 처리하는데 부적합하다.

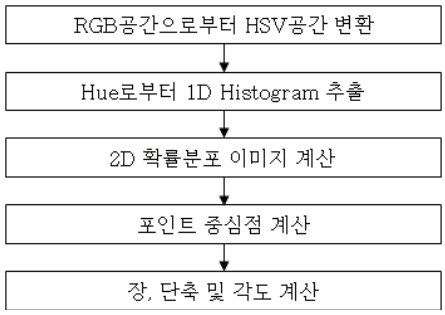
3. 레이저 포인터의 추적

검출된 객체의 영역의 데이터로부터 Hue 값의 분포를 이용하여 변환될 위치를 예측하고 탐지한 후 중심을 찾아 객체를 추적하게 된다. 또한 색 기반 추적 알고리즘으로 조명이나 주위 얼굴색에 민감하게 반응한다. 프레임에서 영역을 예측하여 검색 영역을 제시함으로서 전체를 스캔하던 기존 방법을 효율적으로 개선하였다.

Mean-Shift의 정지영상 색 분할 알고리즘은 비디오 영상에서 처리할 수 있도록 확장된 알고리즘이다[7]. 통계적인 방식을 도입하여 평균점을 구하여 포인트를 추적하였다.

CamShift 알고리즘은 2D 색상 확률 분포 영상을 이용하고 분포의 변화에 동적으로 반응한다[8]. 진행 과정에서 윈도우의 사이즈도 조정한다. 현재의 윈도우 크기나 검출된 물체의 위치는 내부적으로 저장 및 출력되고 이 정보는 다음 영상과의 연산에 사용된다. 이러한 연산의 반복에 의해서 오브젝트의 검출은 이루어진다.

중심점의 값을 x축, y축의 1차 모멘트에서 0차 모멘트를 나누는 방법으로 해결할 수 있었고 물체의 각도와 장축, 단축의 길이는 물체의 영역을 타원으로 일반화하여 2차 모멘트로부터 값을 얻었다.



(그림 1) CamShift 처리 과정

3.1 CamShift 처리 과정

CamShift 알고리즘[9]을 처리하기 위해서는 일단 RGB 공간으로 이루어진 입력 영상 정보를 HSV 공간으로 변환하여 Hue 색상 정보만 사용할 수 있도록 한다. 얻어진 Hue 정보로부터 1D 히스토그램을 계산하고 2D 확률 분포를 얻는다. 얻어진 분포에서 포인트의 중심점, 장축, 단축 및 각도를 계산한다. (그림 1)에서 처리 과정을 가시화하였다.

3.2 CamShift를 이용한 확률 분포 영상 계산

HSV 공간을 사용하면 각각의 요소가 서로 독립적인 장점이 있다. CamShift에서는 RGB 공간을 사용하지 않고 HSV 공간을 사용한다. RGB 공간에서 HSV 공간으로 변경 후 Hue를 사용한다. Hue를 사용하면 조명의 영향에 덜 민감하여 Hue값에서 아주 작은 값이나 큰 값을 알아볼 수 없으므로 조명의 영향을 배제할 수 있다.

Hue값으로부터 1D 히스토그램을 구한다. 비가중치 히스토그램을 기준으로 수식 (2)을 통해서 1D 히스토그램을 구할 수 있다. 수식 (2)에서 n을 영상 픽셀 위치로 정의하고 $c(x_i^*)$를 x_i^*의 위치로 정의한다.

수식 (3)을 통해서 2D 확률 분포 영상을 얻는다. 영상은 2D 확률 분포의 최대 영상을 기준으로 크기를 늘린다. 범위는 0부터 255사이로 제한시켜야 한다. 이는 차후 포인트의 위치 및 정보를 얻는데 사용된다[8].

$$\hat{q_u} = \sum_{i=1}^{n} \delta[c(x_i^*) - u] \quad (2)$$

$$\hat{p_u} = \min\left(\frac{255}{\max(\hat{q})} \hat{q_u},\ 255 \right)_{u=1\ldots m} \quad (3)$$

3.3 CamShift를 이용한 계산

CamShift의 연산은 0, 1, 2차 모멘트를 이용해서 타원으로 해당 물체를 일반화해서 값을 얻는다. CamShift 연산의 절차는 4단계로 설명할 수 있다[10]. 첫 번째 단계에서 수식 (4)로부터 0차 모멘트를, 수식 (5)와 수식 (6)을 이용해서 1차 모멘트를 계산한다.

$$M_{00} = \sum_x \sum_y I(x,y) \quad (4)$$

$$M_{10} = \sum_x \sum_y x I(x,y) \quad (5)$$

$$M_{01} = \sum_x \sum_y y I(x,y) \quad (6)$$

두 번째 단계에서는 수식 (7)과 수식 (8)을 이용해서 포인트의 중심점을 계산한다. 두 1차 모멘트를 각각 0차 모멘트로 나누면 무게중심의 좌표를 얻을 수 있다.

$$x_c = \frac{M_{10}}{M_{00}} \quad (7)$$

$$y_c = \frac{M_{01}}{M_{00}} \quad (8)$$

세 번째 단계에서는 수식 (9), (10), (11)을 이용해서 2차 모멘트를 계산한다.

$$M_{20} = \sum_x \sum_y x^2 I(x,y) \quad (9)$$

$$M_{02} = \sum_x \sum_y y^2 I(x,y) \quad (10)$$

$$M_{11} = \sum_x \sum_y xy I(x,y) \quad (11)$$

네 번째 단계에서는 수식 (12), (13), (14)을 이용해서 값을 구하기 위한 매개 변수 a, b, c를 계산한다.

$$a = \frac{M_{20}}{M_{00}} - x_c^2 \quad (12)$$

$$b = 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right) \quad (13)$$

$$c = \frac{M_{02}}{M_{00}} - y_c^2 \quad (14)$$

다섯 번째 단계에서는 수식 (15), (16), (17)을 이용해서 각도 θ와 해당 타원 영역의 장지름 L과 단지름 S를 계산한다.

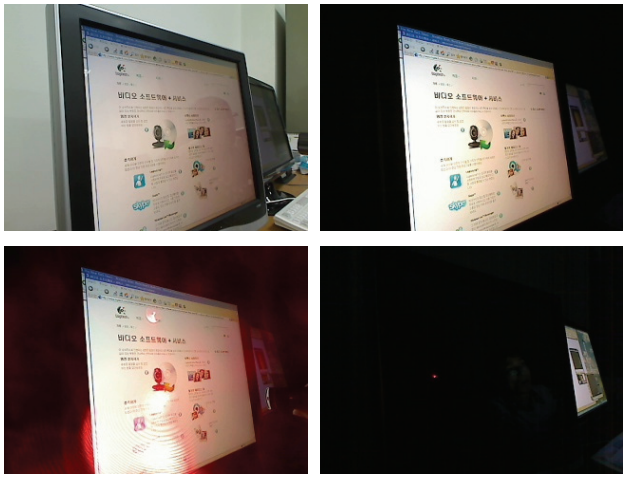
$$\theta = \frac{1}{2} arctan\left(\frac{b}{(a-c)}\right) \quad (15)$$

$$S = \sqrt{\frac{(a+c) - \sqrt{b_2 + (a-c)^2}}{2}} \quad (16)$$

$$L = \sqrt{\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2}} \quad (17)$$

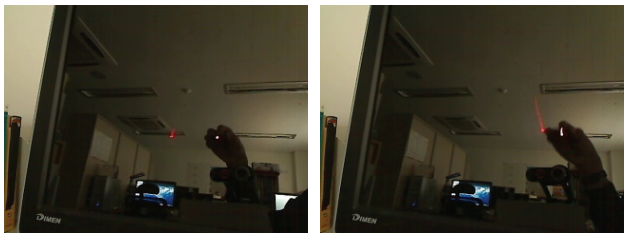
4. 실험 결과

주변의 조명이 다양한 경우와 사용자의 움직임의 패턴 및 속도가 다양한 경우에 대해서 레이저 포인터 추적 실험을 수행하였다.

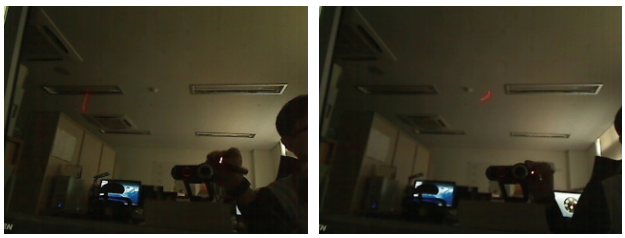


(그림 2) 조리개의 개방 정도에 따른 포인터 상의 변화: (a),(b),(c),(d)

(그림 2)에서 (a)는 외부 조명이 비교적 강한 환경에서 레이저 포인터를 비춘 모습이다. 사람의 눈으로도 인식이 매우 어려운 수준이다. (b)는 외부 조명이 거의 없는 환경에서의 레이저 포인터를 비춘 모습이다. (a)와 마찬가지로 식별이 어려움을 알 수 있다. (c)는 레이저 포인터가 카메라 방향과 일치하여 눈부심이 심하게 발생하는 경우이다. 정확한 레이저 포인터의 위치를 정의하기 어렵다. (d)는 디스플레이에서 발광원이 없는 경우에서의 레이저 포인터를 비춘 모습이다. 레이저 포인터의 모습이 선명하게 확인된다. 전체적으로 (d)와 같이 이상적인 상황에서만 레이저 포인터의 위치를 올바르게 인식할 수 있다.



(그림 3) 사용자의 움직임 속도에 따른 포인터 추적 실험



(그림 4) 사용자의 패턴에 따른 포인터 추적 실험

(그림 3)에서 사용자의 움직임 속도에 따라서 어떤 차이점이 발생하는지 확인하였다. 왼쪽 화면은 사용자의 움직임이 많지 않을 때에 해당하고 오른쪽 화면은 사용자가 위에서 아래로 빠르게 움직일 때에 해당한다. 왼쪽 화면에서는 포인트가 좁고 강하게 나타났고 오른쪽 화면은 레이저광의 빛번짐(glare)이 나타나서 포인트의 위치가 불명확하게 된다. 포인트의 중심점의 위치를 계산하는 과정에서 오차 범위가 증가하게 된다.

(그림 4)에서는 사용자 움직임의 패턴을 보여준다. 왼쪽 화면은 사용자가 포인터를 위에서 아래로 움직인 경우이고 화면에는 직선 형태로 나타난다. 오른쪽 화면은 포인터를 원 궤도를 돌면서 움직인 경우이고 화면에는 곡선을 그리면서 나타난다.

사용자의 움직임 패턴 및 속도는 포인트의 추적에 큰 영향을 미친다. 따라서 추적은 사용자가 레이저 포인터를 사용하는 환경에 따라서 적응적으로 동작되어야 한다. 또한 주변의 조명이나 사용자의 움직임에 대해 일정한 제약을 가해서 추적의 안정성을 높일 수 있다.

레이저 포인트를 추적하는데 사용되는 CamShift 알고리즘의 이점들은 다음과 같다[5].

- 계산량이 적어서 실시간 비디오 처리에서 적합하다.
- 배경의 변화에 영향을 적게 받는다.
- 포인트가 관심 영역을 벗어난 후 다시 영역 안으로 들어왔을 때 효과적으로 추적할 수 있다.
- 추적하고자 하는 레이저 포인터 색과 주변의 색이 유사한 경우의 간섭 현상을 줄일 수 있다.

CamShift 알고리즘은 탐색 윈도우의 크기를 스스로 조절한다는 점에서 Mean-Shift 알고리즘과 다르다. 해당 물체가 화면상의 거리가 가깝거나 멀어지면 그러한 변화를 인식해서 윈도우 크기를 변화시켜서 처리할 수 있다.

기존의 방법을 개선하여서 동적으로 변하거나 복잡하게 구성된 배경에 대해서도 처리가 가능하고 물체의 움직임이 많을 때도 물체의 실제 위치를 효과적으로 추적할 수 있다. 이러한 CamShift 알고리즘은 레이저 포인터를 추적하는데 효과적으로 사용될 수 있다.

5. 결론

레이저의 포인터 추적의 경우에는 주변의 조명 환경과 사용자의 움직임 형태에 따라서 추적의 안정성이 달라진다. 사용자의 움직임 패턴이나 조명 환경에 일정한 제약을 두는 경우에는 안정성을 높일 수 있으나 제약 자체는 활용성을 떨어뜨린다는 점에서 바람직하지 않다. 제약과 무관하게 안정성을 높이는 방법으로 CamShift 알고리즘을 사용할 수 있으며 실시간 레이저 포인터 추적에 적용하였다.

레이저 포인트를 추적하기 위한 알고리즘인 CamShift 알고리즘은 계산량이 적기 때문에 실시간 영상에 대한 처리에 적합하다. 그러나 심한 조도 변화 및 잡음에는 여전히 불안정성이 나타난다. 이에 대한 문제점을 보완한다면 포인트 추적에 있어서 더욱 강건한 레이저 포인터 추적 시스템이 될 것이다.

참고문헌

[1] C. Kirstein and H. Müller, "Interaction with a Projection Screen Using a Camera-tracked Laser Pointer", Proceedings of The International Conference on Multimedia Modeling, pp.191-192, 1998.

[2] J. Lapointe and G. Godin, "On-screen Laser Spot Detection for Large Display Interaction", HAVE'05, pp. 72-76, 2005.

[3] Toshiharu Wada, Masanobu Takahashi Keiichiro Kagawa, Jun Ohta "Laser Pointer as a Mouse", SICE Annual Conference, pp. 17-20, 2007.

[4] Cheng, Yizong, "Mean Shift, Mode Seeking, and Clustering", IEEE T-PAMI, Vol. 17(8), pp.790-799, 1995.

[5] R.Stolkin, I .Florescu, G. Kamberov, "An Adaptive Background Model for CamShift Tracking with a Moving Camera", ICAPR'07, pp. 261-265, 2007.

[6] R. Stolkin, I. Florescu, M. Baron, C. Harrier and B. Kocherov, "Efficient visual servoing with the ABCshift tracking algorithm", ICRA'08, pp. 3219-3224, 2008.

[7] Xia Liu, Hongxia Chu, Pingjun Li, "Research of the Improved Camshift Tracking Algorithm", ICMA'07, pp. 968-972, 2007.

[8] John G. Allen, Richard Y. D. Xu, Jesse S.Jin, "Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces", VIP'2004, pp. 3-7, 2004.

[9] Jie Xia, Jian Wu, Haitao Zhai, Zhiming Cui, "Moving Vehicle Tracking Based on Double Difference and CamShift", International Symposium on Information Processing, pp. 29-32, 2009.

[10] Lexiao Ye, Yigang Wang, "Real-time Tracking of the shoot Point from Light Pen Based on CamShift", ICINIS'08, pp. 560-564, 2008.