모바일 인터페이스 제어를 위한 움직임 추정 기법

*이철우, **김창수

고려대학교

*wiserain@korea.ac.kr, **changsukim@korea.ac.kr

Motion Activity Estimation for Mobile Interface Control

*Chul-Woo Lee **Chang-Su Kim

Korea University

요약

본 논문에서는 휴대폰이나 UMPC 등의 모바일 기기에 내장된 카메라를 이용하여 입력 영상을 통해 전역적인 움직임 벡터를 취득하고 이를 이용해서 모바일 인터페이스를 제어하는 기법을 제안한다. 카메라로부터 입력되는 영상에서 특징점을 추출하고 광흐름을 기반으로 각각의 특징점에 대한 움직임을 추정한다. 그 과정을 통해서 생성된 움직임 벡터의 집합으로부터 affine 행렬을 계산하여 전체 화상의 움직임을 표현하는 파라미터를 도출할 수 있다. 움직임 파라미터 값은 다시 인터페이스를 제어하는 신호를 생성하며 이 움직임 신호는 메뉴 네비게이션, 슬라이드 쇼 및 문서 스크롤과 같은 모바일 인터페이스의 제어에 이용될 수 있다. 모의 실험을 통하여 인터페이스 제어를 위한 화상의 움직임 정보가 적절히 획득됨을 확인한다.

1. 서론

소니의 워크맨(Walkman)이 모바일 기기의 시초가 된 이후로부터 어디서나 가지고 다닐 수 있는 포터블 기기는 우리의 생활의 일부가 되었다. 사람들은 언제 어디서나 모바일 기기와 상호작용을 통해서 지시를 내리고 정보를 얻는 작업을 하루에도 수십 번 이상을 행한다. 이런 상황 하에서 효과적인 인터페이스의 제어는 작업 효율을 증대시키는 중요한 역할을 한다.

고전적인 인터페이스 조작 방법은 키를 이용한 방법(Key-based)이다. 어떤 상황 하에서 특정키를 누름으로써 할당된 신호가 전송되고 해석되어 정해진 작업을 수행하는 방식이며 정확도가 높아 여전히 주요한 인터페이스의 조작 방법으로 활용되고 있다. 하지만 키를 누르는 제한적인 행동으로만 입력을 가할 수 있어, 기기와 소통하기 위한 새로운 제어방법을 찾는 연구가 행해지고 그 결과로 실제로 많은 영역에서 적용이 이루어지고 있다. 터치 기반(Touch-based)의 인터페이스 제어는 키 기반의 기법에서 조금 더 진보된 방법이다. 키 기반의 제어 기법은 사용자가 얻는 화면의 정보와 입력하는 장소가 상이한 반면 터치 기반의 인터페이스에서는 보이는 그대로 입력함으로 더욱 유기적인 사용자와 기기의 연계성을 보여준다. 이 방식은 현재 시중의 MP3 플레이어나 휴대폰에 적용되어 있다. 하지만 이 방법 역시 키 입력 방식이 가지는 한계를 그대로 물려받아 물리적인 접촉을 가해야만 된다는 한계를 가지고 있다. 다른 방법으로는 현재 많이 채용되고 있는 기법인 가속 센서를 이용한 방법이 있다. 적은 비용으로 기기의 위치와 움직임 정보를 얻을 수 있으며, 이를 이용하여 어플리케이션에서 활용되고 있다. 하지만 실제로 이용되고 있는 기기를 살펴보면 가속 센서는 다른 인터페이스 컨트롤 기술과는 달리 그 자체가 주요한 역할을 하기 보다는 보조적으로 쓰여 다른 기술과 결합하였을 때 의미가 있다. 결국 그 활용 범위가 제한적이 될 수밖에 없다.

반면 카메라를 활용한 기술은 많은 주목을 받고 있다. 영상이 주는 정보는 다른 감각기관을 통해서 얻는 것보다 수십 배의 정보를 가지고 있다. 카메라를 통해서 얻어지는 밝기 정보나 거리 정보의 추출, 객체 추적 등의 정보는 다른 인터페이스 제어 기법에서 줄 수 있는 제한적인 정보와는 달리 직관적이고 다양한 정보를 전달 할 수 있다.

따라서 본 논문에서는 현재 모바일 기기에서 카메라가 기본으로 채용되고 있다는 점에 착안하여 카메라를 통한 입력 영상을 분석하여 전역 움직임을 추정하고 그 정보를 모바일 인터페이스 제어에 적합한 신호로 변환하는 방법을 제안한다. 2장에서는 제안하는 알고리듬에 대해 설명하며 3장에서는 실험 결과를 제시하고 마지막으로 4장에서 결론과 향후 연구 과제를 도출한다.

2. 본론

카메라가 켜지면 그 순간부터 연속적인 일련의 영상신호를 받는다. 이전 시간 $t-1$에서의 이미지를 I_{t-1}라고 한다면 제안하는 알고리듬은 이전 시간과 현재시간, t에서 얻어지는 두 장의 이미지를 기반으로 움직임 정보를 추출한다. 우선 이전 시간의 영상, I_{t-1}에서 특징점을 추출하여 움직임을 추적할 점을 찾고 광흐름을 이용하여 다음 영상인 I_t에서의 위치를 추적하여 대응점의 집합을 찾는다. 점들의 집합을 통해서 전역적인 움직임을 표현하는 Affine 행렬을 도출하여 움직임을 정량화한 값을 얻는다. 이 값을 이용하여 동작을 판별하고 특정 신호를 생성한다. 다음에 이어지는 내용은 각 단계를 자세히 설명하고 있다.

그림 1 특징점의 편중 현상

그림 2. 특징점 추출 결과

가. 특징점 추출과 갱신

제안하는 알고리듬에서 특징점을 추출하는 것은 매우 중요한 단계이다. 효과적인 특징점의 추출은 차후 이루어지는 움직임 정보의 검출과 그 결과의 신뢰도에 큰 영향을 미친다. 특징점을 추출하기 위해서 입력되는 영상의 모든 화소, i에 대해서 다음과 같은 식을 통해서 해당 화소 주변의 정보에 기반한 자기상관행렬을 계산한다.

$$M(i) = \sum_{j \in W(i)} \begin{bmatrix} I_x^2(j) & I_x(j)I_y(j) \\ I_x(j)I_y(j) & I_x^2(j) \end{bmatrix}$$

$W(i)$는 화소 i를 중심으로 한 사각 윈도우이며 $I_x(j)$와 $I_y(j)$는 각각 x, y 방향으로의 그래디언트이다.

자기상관행렬 $M(i)$를 통해서 우리는 행렬의 고유치(Eigenvalue)를 구할 수 있다. 이 고유치가 충분히 크다면 이 화소는 움직임을 추적하기 용이한 화소로 선택 된다. 해당 프레임의 영상에서 모든 화소에 대해 자기상관행렬과 고유치를 구하고 고유치가 큰 순서대로 정렬을 한 뒤, 상위 n개의 화소를 움직임 추정을 위한 초기 특징점으로 할당한다.

위의 방법은 Shi and Tomasi에 의해서 소개 되었다[1]. 그들은 자기상관행렬의 고유치가 큰 화소가 중요한 특징점이 되며 추적에 효율적이라는 점을 증명하였다.

이미지 프레임의 모든 영역이 특징점의 후보로서 적절한 것은 아니다. 이미지의 경계부분은 물체나 특징점이 지나가면서 추적점을 잃어버리게 된다. 예를 들어, 이전 영상에서 경계 주변에서 특징점이 검출 되었다고 가정하자. 만약 해당 화소가 화면의 중심으로 이동한다면 그 점은 현재 영상에서 이동한 화소를 찾는 과정에 아무런 문제가 없다. 하지만 화면의 밖으로 이동을 하게 되면 추적은 실패하게 된다. 이 문제를 사전에 방지하기 위해 특징점은 외곽 경계 부분은 제외한 영역을 대상으로 추출하게 된다.

추출된 특징점은 다음 장에서 소개되는 단계를 거치면서 다시 특징점 추적의 입력이 된다. 하지만 잡음으로 인해 처음에 얻어진 n개의 특징점은 과정을 반복할 때마다 계속해서 줄어든다. 일정 수의 신뢰도 높은 특징점의 확보가 알고리듬의 중요한 요소이므로 이 문제를 해결

그림 3. 광흐름을 이용한 추적으로 발생된 움직임 흐름

하기 위해서 실패된 k개의 화소를 다음 반복되는 과정을 위해 추가해 준다. 또한 추적에 실패한 점의 수, k와 특징점의 개수, n의비율이 절반을 넘는다면 기존의 성공한 점들도 모두 포기하고 전체 이미지 영역을 대상으로 다시 n개의 특징점을 설정한다.

위와 같은 영상의 움직임을 효율적으로 반영하기 위한 특징점 선정 기준에도 불구하고 특징점은 추적을 반복할수록 고유치가 높은 점에 편중되는 현상이 발생한다.(그림 1) 따라서 일정수의 프레임이 지나면 다시 초기화 과정을 거쳐 특징점을 다시 설정하도록 한다. 위와 같은 방법을 통해서 특징점과 추적점의 집합은 일정한 수를 유지하게 된다. 그림 2는 특징점 추출의 결과로 화면의 전 영역에 걸쳐 고르게 분포하고 있는 특징점들이 빨간색으로 표시되어 있는 모습을 확인할 수 있다.

나. 광흐름(Optical Flow)을 이용한 특징점 추적

추출한 n개의 특징점을 통해서 현재 영상에 대응하는 점들의 집합을 도출한다. 특징점 추적에는 광흐름의 방법을 사용한다. 광흐름의 방법에는 몇 가지 방법[2]이 있지만 제안하는 알고리듬에서는 Lucas-Kanade 광흐름 추적 방법[3]을 사용한다. Lucas-Kanade 기법은 1981년 처음 이론이 소개된 이후로 많은 파생 연구가 진행되었다. 그 중에서도 이미지 피라미드를 이용하여 반복적인 방법으로 다음 특징점을 추적하는 방법을 적용한다[4]. 그림 3은 Lucas-Kanade Pyramid 방법에 의해서 이전 영상에서 찾은 점들의 집합을 통해 움직임 흐름을 보여준다.

광흐름은 그 이론의 배경이 빛의 흐름을 추적하는 알고리듬이다. 따라서 실제 화면에서는 객체가 움직이지 않음에도 불구하고 빛의 변화가 심한 환경이나 카메라의 자동 밝기 조절 기능에 의해서 의도하지 않은 움직임이 감지되는 경우가 있다. 또한 광흐름은 입력되는 영상의 시간 차이가 아주 짧다는 가정에 기반하여 도출된 이론이므로 매우 빠른 움직임의 경우에는 오류가 발생하기도 한다.

이에 의해서 발생하는 오류를 보정하기 위해 이전 영상에서의 점과 다음 영상에서 대응하는 점 사이의 기하학적인 거리를 계산하여 그 값이 다른 값의 평균보다 크다면 오류로 판별하여 다음 업데이트에서 제외한다. 이렇게 신뢰도가 낮은 점 k개는 손실되어 $n-k$개의 특징점과 추적점의 쌍이 다음 단계로 전해진다.

다. Affine 변환

앞선 과정에서 시간차를 두고 얻은 2장의 영상을 I_{t-1}, I_t라고 한다면 광흐름 추적 과정을 거쳐 얻어진 각 영상에 대응되는 점의 집합은 아래와 같다.

$$P_t = \{p_1, p_2, p_3, \cdots, p_{n-k}\}$$

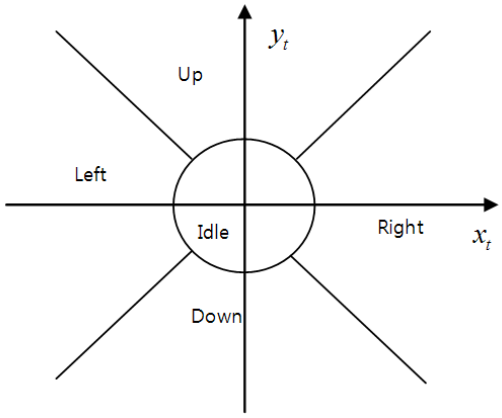

그림 4. 움직임 결정 기준

전체 n개의 화소 중 k개의 화소는 오류로 인해 제거 되고 최종 $n-k$개의 화소만이 반영되었다. 이 점들은 서로 대응하는 쌍을 이루며 최소자승법을 이용하여 다음 식을 최소화 하는 Affine 변환 행렬, A_t를 구할 수 있다.

$$\sum_{p_t \in P_t} (p_{t-1} A_t - p_t)^2$$

여기서 p_t는 Homogeneous 좌표로써 $p_t = [p_x, p_y, 1]^T$이다. 얻어진 Affine 변환 행렬, A_t는 두 좌표, p_x, p_y로 표현되는 벡터 공간의 관계를 나타내는데 행렬 내의 특정 원소나 그 조합을 통해서 병진(Translation), 축척(Scaling), 회전(Rotation), 층밀리기(Shearing)와 같은 선형 좌표 변환을 감지할 수 있다. 그 중에서 병진 이동을 이용하여 2차원의 가로, 세로 방향의 움직임 벡터는 주어진 A_t에 대해서 다음과 같이 각각 얻을 수 있다.

$$x_t = g$$
$$y_t = h$$
$$A_t = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

라. 움직임 판별

실시간으로 갱신되는 움직임 벡터는 미세한 잡음에 의해 그 안정도가 떨어진다. 따라서 민감도의 조절이 필요하며, 그것을 위해 이동 평균(Moving Average)을 이용한다. 이동 평균은 실시간으로 심한 변화가 있는 값을 장기적인 관점에서 분석하고자 할 때 주로 쓰이는 방법이며 경제 지표 분석에 사용되는 기본적인 도구 중에 하나이다. 아래와 같이 현재부터 과거 특정 시점까지의 평균을 통해 값을 안정화 시킨다.

$$MA = \frac{1}{m} \sum_{k=t-m}^{t} x_k$$

x_k는 시간 k에서의 움직임 벡터이며 (다)에서 구한 x_t나 y_t가 될 수 있다. m은 평균을 구할 시간 범위이다. 여기서 m이 작다면 매 프레임 업데이트 되는 결과는 여전히 불안정할 것이고 반대로 m이 크다면 안정된 결과를 보여주지만 실시간으로 변화하는 값을 즉각적으로 보여주지 못하게 된다. 따라서 적절한 m의 선택이 중요하다.

이동 평균을 통해서 안정화된 값들은 그림 4에 나타난 기준을 통해서 미리 정해진 5개의 동작으로 판별된다. 5개의 동작은 2차원의 움직임과 정지 상태를 포함하며 다음 단계를 위해 각 동작을 상징하는 하나의 문자로 변환된다. 그 값은 I(정지 상태, Idle), R(오른쪽, Right), L(왼쪽, Left), U(위쪽, Up), D(아래쪽, Down)이다.

마. 제스쳐 인식

이제 각 영상의 움직임은 이전 시간의 영상에 대해 가지는 움직임의 정보를 뜻하는 한 글자의 문자 코드로 표현된다. 그러나 이 정보 자체는 사용자의 입력을 적절하게 시스템에 반영하기 어렵다. 인터페이스 제어를 위한 동작의 인식은 시작점과 끝점의 정확한 구분이 필요하다. 예를 들어 오른쪽으로 3번의 신호를 전달하고 싶다고 하자. 사용자는 카메라를 조작하여 3번의 오른쪽, R에 해당하는 신호를 발생시키지만 컴퓨터 입장에서는 정확히 몇 번의 동작이 입력되었는지 가려내는 것은 쉬운 문제가 아니다. 따라서 동작의 시작과 끝을 구분하여 그 구간 내에 입력되는 동작만을 해석하도록 하여야 한다. 또한 단순히 5개의 동작 신호, IRLUD만을 이용한다면 똑같이 5개의 명령밖에 할당할 수 없을 것이다.

문제를 해결하기 위해서 1자리의 코드를 r 자리의 코드로 확장한다. 시스템은 순차적으로 입력되는 1개의 동작 코드를 버퍼에 기록하여 FIFO(First In First Out) 방식으로 처리하다 미리 지정된 제스쳐 코드와 일치하는 조합이 발생하게 되면 할당된 기능을 수행하게도록 한다. 이와 같은 방법을 통해 기존에는 제한적인 5개의 신호에서 이론적으로 5^r개로 확장할 수 있을 뿐만 아니라 민감도 문제도 해결할 수 있다.

3. 실험 결과

카메라의 조작을 통해 입력되는 움직임이 효과적으로 해석되어 원하는 신호를 생성하는지 확인하기 위하여 아래와 같은 실험을 구성하였다.

우선 실험에 사용된 설정값들은 다음과 같다. 초당 24 프레임을 처리하는 PC용 웹캠을 이용하여 영상정보를 입력받고 2 장에 제안된 알고리듬을 이용해 움직임을 판별한다. 초기 특징점의 개수, n은 80개로 설정하였으며 경계부분의 20 픽셀은 특징점 추출에서 제외하였다. 또한 광흐름을 통해 도출되는 움직임 벡터의 길이가 20 이상인 경우는 오류로 판별하도록 하였으며, 이동 평균을 구할 때 적용하는 시간 영역의 크기, m은 8로 하였다. 또한 제스쳐 코드의 길이, r은 3자리로 고정하였다.

미리 신호를 정해 놓고 카메라에 움직임의 조합을 입력하였을 때 원하는 신호가 호출 되는지를 관찰한다. 표 4는 실험 결과를 보여준다. 각각의 제스쳐를 25번 행하고 그 뒤의 의도한 결과가 나온 횟수를 표시하였다. 표에서 첫 4개 열의 결과는 각각의 제스쳐 코드가 I (정지 상태)에서 출발하여 1차원의 값들로만 이루어져 있다. 예를 들면 IRL이나 ILR의 경우는 횡 축으로만 움직임이 조합되어 있는 제스쳐 코드

제스쳐 코드	IRL	ILR	IUD	IDU	IRU	ILD	IUR	IDL
시도 횟수	25	25	25	25	25	25	25	25
일치 횟수	22	19	20	23	17	18	18	17
정확도 (%)	88	76	80	92	68	72	72	68

표 1 제스쳐 인식 결과

이며, IUD와 IDU는 종 축의 움직임으로만 조합되어 있다. 반면에 나중 4개 열의 결과는 횡, 종 축의 움직임이 섞여 있다. 결과를 분석하면 단 방향의 움직임 조합의 제스쳐 코드가 좀 더 높은 정확도를 보여주고 있다. 이는 사용자가 제스쳐를 취할 때 단 방향 관성이 높아서 실제 결과에도 반영되는 모습을 보여주고 있다.

4. 결론

본 논문은 카메라를 사용하여 입력 영상의 움직임 정보를 모바일 인터페이스 제어의 신호로 활용하는 방법을 제안하였다. 제안하는 방법은 특징점을 추출하고 그 움직임을 광흐름 기법으로 추측한다. 그리고 Affine 변환 이론에 근거하여 움직임 벡터를 도출한다. 움직임 벡터의 적절한 가공을 통해서 동작과 그 조합인 제스쳐 코드를 얻을 수 있었다. 실험 결과를 통해 움직임 정보는 효과적으로 해석되어 사용자가 원하는 신호를 생성해 낼 수 있음을 확인하였다.

추후 임계치 기준의 움직임 결정 방법을 개선하여 역동적인 영상 정보에서 신뢰도 높은 움직임을 검출하도록 할 것이다. 또한 거리 정보의 추출 적용하여 카메라를 통해서 인터페이스를 조작할 수 있는 신호를 다양화 할 것이다.

5. 참고 문헌

[1] Shi. J. and Tomasi. C., "Good features to track," IEEE Conference on Computer Vision and Pattern Recognition 2004, Jun. 2004.

[2] B. K. P. Horn and B. G. Schunck "Determining optical flow," Artificial Intelligence, 17, pp. 185-203, 1981.

[3] Lucas. B. D. and Kanade. T., "An iterative image registration technique with an application to stereo vision," In proceedings of the 7th International Joint Conference on Artificial Intelligence, Apr. 1981.

[4] J.-Y. Bouguet., "Pyramidal implementation of the lucas kanade feature tracker," Included in OpenCV library, 2001.

[5] J. Mooser, S. You, and U. Neumann, "Large document, small screen: A camera driven scroll and zoom control for mobile devices," in Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D), pp. 27-34, Feb. 2008.