모바일 환경에서 모션 추정을 위한 빠르고 정확한 알고리즘

Fast and Accurate Algorithm for Motion Estimation in Mobile Environments

김준호, 오일석
전북대학교 컴퓨터공학부

Junho Kim(juno4u@chonbuk.ac.kr), Il-Seok Oh(isoh@chonbuk.ac.kr)

요약

본 논문에서는 모바일 환경에서 모션을 추정하기 위하여 특징점을 코너로 선택하고, 기존 코너 검출 알고리즘인 Rosten 알고리즘 보다 모션 추정의 정확성을 높이는 방법을 제안한다. Rosten 알고리즘은 간단한 덧셈과 뺄셈만으로 코너를 검출함에 따라 처리 속도가 빠른 장점이 있지만, 코너를 매칭하면서 모션 추정의 성능이 떨어지는 단점을 가지고 있다. 우리는 Normalized Cross Correlation(NCC) 계수를 사용하여 코너를 매칭하고, 두 단계에 걸쳐 부정확한 매칭에 의한 외톨이를 제거한다. 기존의 Rosten 알고리즘과 제안하는 알고리즘을 실제 영상에서 실험을 통하여 성능을 비교하고, 실제 모바일 장치의 응용 프로그램에 적용한다. 제안한 알고리즘이 Rosten 알고리즘 보다 더 정확한 모션을 추정해 냄을 실험 결과 확인하였고, 실제 모바일 환경에서도 실시간으로 동작함을 확인하였다.

■ 중심어 : | 모바일 상호작용 | 사용자 인터페이스 | 모션 추정 |

Abstract

In this paper, we propose a new method of improving accuracy of motion estimation in mobile environments, compared with Rosten's algorithm. The present method selects corners as feature points. The Rosten's algorithm uses simple addition and subtraction to detect the corners. Although it has the advantage of faster processing speed, Rosten's algorithm has a drawback of low performance in motion estimation. We use the NCC(Normalized Cross Correlation) coefficients to match the corners, and remove in two steps the outliers of inaccurate matching corners. We compare the proposed algorithm with Rosten's algorithm by applying both to the real images. We find that the proposed method shows better performance than Rosten's algorithm in motion estimation. In addition, we implement the present method on mobile devices and confirm that it works in mobile environments in real time.

■ keyword : | Mobile Interaction Technique | User Interface | Motion Estimation |

I. 서 론

휴대 전화를 포함한 모바일 장치는 언제 어디서나 가지고 다닐 수 있는 생활의 일부분이 되었다. 모바일 장치의 기능은 단순히 전화를 걸고 받는 기능뿐만 아니라 문자 메시지를 주고받으며, 사진을 찍거나 인터넷에 접

접수번호 : #091110-004
접수일자 : 2009년 11월 10일
심사완료일 : 2009년 11월 26일
교신저자 : 오일석, e-mail : isoh@chonbuk.ac.kr

속하여 메일을 보내고, 전자상거래, 교통 정보와 길 안내, 음악 및 동영상 감상 등 다양한 기능을 하는 멀티미디어 IT기기로 발전하고 있다. 이처럼 모바일 장치는 단순한 기계에서 벗어나 '손안의 작은 컴퓨터'라 불리며 이를 사용하는 인구는 더욱 늘어나는 추세이다. 이에 따라 사용자와 모바일 장치 사이에는 하루에도 수십 번 이상의 상호작용이 이루어지며, 이를 통하여 명령을 내리고 정보를 얻는다. 이런 상황에서 모바일 장치와 사용자간의 효과적인 상호작용 방법에 대한 새로운 요구가 증가하고 있다.

현재 모바일 장치와 상호작용을 하는 방법 중 가장 기본적이면서 대표적인 것이 바로 12버튼의 키패드를 사용하는 것이다. 그러나 키패드를 사용하는 방법은 매우 단순하고 제한적인 콘텐츠에서 사용이 가능하고, 다양한 콘텐츠 출현을 저해하고 있다.

최근 모바일 장치를 위하여 다양한 방식의 상호작용 방법들이 제안되고 있다. 그 중 대표적인 것이 바로 모션 정보를 사용하여 모바일 장치와 상호작용을 하는 것이며, 모바일 게임과 여러 응용 프로그램에 적용된 사례가 발표되고 있다. 모바일 장치에서 모션 정보를 획득하는 대표적인 방법으로는 가속도 센서(accelerometer)나 각속도 센서(gyroscope)와 같은 하드웨어 센서를 사용하는 것이다. 하지만 현재 출시된 대부분의 모바일 장치에는 이러한 센서가 탑재되어 있지 않고, 가격이 비싸다는 단점이 있다.

모션 정보를 획득하는 다른 방법으로는 모바일 장치에 탑재된 카메라를 이용하는 것이다. 최근 출시되는 모바일 장치는 대부분 고성능의 카메라를 내장하고 있으며, CPU의 계산 능력뿐만 아니라 메모리도 상당한 발전을 이루었다. 이러한 환경은 카메라를 모바일 장치의 인터페이스로 사용이 가능하게 한다. 하지만 카메라를 통하여 모션을 추정하는 것은 많은 연산을 필요로 하며, 현재의 모바일 장치는 이를 위해 충분한 연산 능력을 제공하지 못하고 있다. 그렇기에 모바일 환경에서 모션 추정을 하기 위하여 더욱 빠르고 정확한 알고리즘이 필요하다.

모바일 장치의 카메라를 사용하여 인터페이스로 사용하는 방법은 모바일 장치의 위치나 움직임을 추정하는 것이다. 모션을 추정하는 방법 중 하나는 마커를 사용하여 움직임 정보를 추정하는 것이다. Bucolo[1]와 Hachet[2]은 미리 정의된 마커를 모바일 장치의 모션을 추정하는 기준으로 사용하며, 모바일 장치의 카메라를 통하여 얻어진 모션 정보는 게임의 인터페이스로 사용한다. 하지만 마커를 기반으로 하여 모션을 추정하는 방법은 항상 마커를 휴대해야하는 커다란 단점이 있다. Mosquito Hunt[3]와 Marble Revolution[4]은 마커를 사용하지 않고 간단한 광흐름(optical flow) 기술을 사용하여 모바일 장치의 움직임을 추정하여 인터페이스로 사용한다. Wigdor[5]는 기존의 모바일 장치에서 키패드를 통하여 입력되던 글자 입력 방식을 모바일 장치의 기울임 정보를 통하여 글자를 입력한다. 모바일 장치의 기울임 정보는 내장된 카메라를 이용하여 입력된 모션 플로우가 어느 방향으로 기울어지는지 가장 우세한 쪽으로 선택하여 해당되는 글자를 입력한다. Drab[6]은 Projection Shift Analysis라는 간단한 모션 추정 알고리즘을 제안하고, 입력받은 영상에서 x, y축 방향으로 투영한 히스토그램 정보를 인접한 영상 프레임 간에 비교하여 모션을 추정한다. Haro[7]와 Wang[8]은 블록 매칭에 기반을 둔 모션 추정 알고리즘을 제안하였다. Hannuksela[9]는 모바일 환경에서 마커를 사용하지 않고 실시간으로 모션을 추정하는 블록 매칭기반의 뛰어난 알고리즘을 제안하였고, Kim[10]은 시간 일관성에 기반한 모션 추정 알고리즘을 제안하였다.

이처럼 모바일 장치의 카메라를 이용하여 움직임을 추정하고, 이를 통하여 모바일 장치와 상호작용을 하기 위한 새로운 인터페이스의 연구는 활발히 진행되고 있다.

우리는 코너 기반의 매칭 알고리즘을 사용하여 모바일 환경에서 실시간으로 빠르고 정확한 모션을 추정하는 방법을 제안하고자 한다. 모바일 환경에서 기존의 코너 검출 알고리즘을 적용하면 계산량이 커지며, 제한된 시간 안에 코너를 검출하지 못한다. 이에 따라 Rosten[11] 코너 매칭 알고리즘의 문제점을 개선하고, 영상의 밝기 값에 따라 영향을 적게 받으면서 모션 추정이 가능한 빠르고 정확한 알고리즘을 제안한다.

Rosten의 코너 매칭 알고리즘은 성능이 떨어지는 두 가지 문제점이 존재한다. 첫 번째 문제점은 두 영상 프

레임 F_{t-1}과 F_t사이에서 찾아진 코너들을 매칭할 때 Sum of Squared Differences(SSD)를 사용함으로써 성능이 떨어지는 문제이고, 두 번째 문제점은 F_{t-1}에서 찾아진 코너가 F_t에서는 검출되지 않고 사라지는 경우 성능이 떨어지는 문제이다.

우리가 제안하는 알고리즘은 연속된 F_{t-1}과 F_t사이에서 찾아진 코너를 매칭하기 위하여 해당 코너의 주변에 위치한 픽셀들의 밝기 값에 따른 상관관계를 비교하여 코너를 매칭하는 것이다. 또한, 상관관계를 비교할 때 임의의 임계값에 따라 부정확한 outlier를 제거하고, 모션 추정시 F_{t-1}과 F_t사이에서 각각의 코너들이 이동한 거리 차를 정렬하여 확률이 높은 모션 정보만 취한다. 실험 결과를 통하여 제안하는 방법이 빠른 시간 내에 보다 더 정확한 모션 추정이 가능한 것을 보이며, 실제 모바일 장치의 여러 응용프로그램에 제안 알고리즘을 적용한 모습을 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 Rosten의 빠른 코너 검출 및 매칭 알고리즘에 대한 관련연구를 설명하고, 3장에서는 Rosten 알고리즘의 문제점 제시와 이를 해결하기 위한 알고리즘을 제안한다. 4장에서는 제안한 알고리즘의 실험 및 평가를 하고, 실제 모바일 장치에 제안한 알고리즘을 적용한 응용 프로그램을 소개한다. 마지막으로 5장에서 결론을 내린다.

II. 관련 연구

다양한 분야에서 물체를 찾거나 움직임을 추적하기 위하여 코너 정보는 중요한 특징으로 사용되며, 정지영상 뿐만 아니라 동영상 내의 복잡한 배경 영상에서 제한된 시간 안에 코너를 빠르고 정확하게 찾는 것이 매우 중요하다. 이때 빠른 속도로 코너를 검출하기위한 기존 알고리즘으로는 Rosten과 Trajkovic[12] 등이 있으며, 이 논문에서는 Rosten의 FAST(Features form Accelerated Segment Test) 코너 검출 알고리즘을 사용하였다.

1. Rosten의 코너 검출 알고리즘

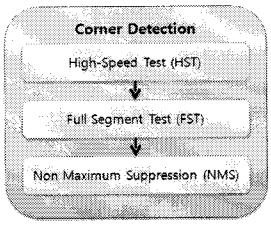


그림 1. Rosten의 FAST 알고리즘 개요

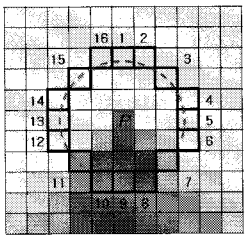


그림 2. 코너 후보 픽셀 P보다 밝은 12개의 인접하는 픽셀

FAST 알고리즘은 [그림 1]와 같은 과정으로 코너를 검출한다. HST 단계에서는 [그림 2]와 같이 P를 코너 후보 픽셀이라고 하고, I_P를 P의 밝기 값이라 했을 때, P 주위의 원에 위치한 1, 5, 9, 13번 픽셀의 밝기 값(I_1, I_5, I_9, I_{13})을 I_P와 임계값 t를 더하거나 뺀 값과 비교한다. 만약 P가 코너라면 $I_P + t$ 보다 밝거나 $I_p - t$ 보다 어두운 것이 적어도 3개 이상일 것이며, 그렇지 않다면 P는 코너가 아니다.

$$\begin{cases} I_i \le I_P - t & (\text{Darker}) \\ I_P - t < I_i < I_P + t & (\text{Similar}) \\ I_P + t \le I_i & (\text{Brighter}) \end{cases} \quad (1)$$

$f(I_i) = \begin{cases} 1, I_i\text{가 } D \text{ or } B \\ 0, I_i\text{가 } S \end{cases}$ 일 때,

$f(I_1) + f(I_5) + f(I_9) + f(I_{13}) \ge 3$라면 P는 코너이다.

HST에서 코너일 것이라고 선택된 후보 코너는 다음 단계인 FST에서 원에 위치한 모든 픽셀의 밝기 값을 $I_P + t$ 나 $I_p - t$ 와 비교하게 되고, 유사하지 않다고 선택된 픽셀의 총 개수 n이 $n \ge 12$ 라면 코너로 선택한다. [그림 2]의 점선은 픽셀 P보다 밝은 12개의 인접하는 픽셀을 나타낸다. HST와 FST를 통해 찾아진 코너들은 Non Maximum Suppression(NMS)을 통하여 국

부적으로 최대값을 가지는 코너를 최종적인 코너로 선택하게 된다.

2. Rosten의 코너 매칭 알고리즘

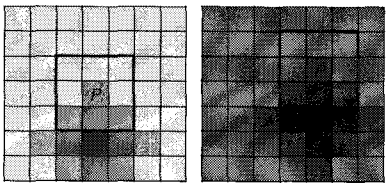
FAST 알고리즘을 이용하여 코너를 검출한 후, 연속된 F_{t-1}과 F_t에서 찾아진 코너들은 코너 매칭을 위하여 SSD 알고리즘을 사용한다. SSD는 F_{t-1}와 F_t에서 매칭하려는 코너들의 해당 픽셀 주위에 위치한 마스크 내 모든 픽셀들의 밝기 차이를 모두 합친 값을 사용한다. $f(x,y)$와 $g(x,y)$는 F_{t-1}과 F_t에서 해당 픽셀의 밝기 값을 의미하며, F_{t-1}과 F_t의 매칭 후보 코너 중 가장 작은 SSD 값을 가진 것을 서로 매치되는 것으로 선택하게 된다.

$$\sum_{x=-n}^{n}\sum_{y=-n}^{n}(f(x,y)-g(x,y))^2 \qquad (2)$$

III. 제안 알고리즘

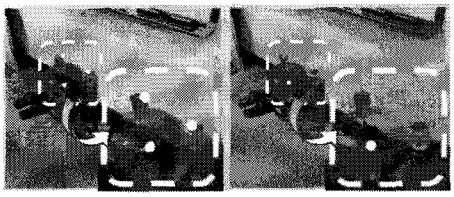
1. Rosten 알고리즘의 문제점

Rosten의 알고리즘은 단순한 덧셈과 뺄셈으로만 이루어져 코너를 검출하기에 기존의 코너 검출 알고리즘들 보다 속도가 빠른 장점이 있다. 하지만 두 프레임 F_{t-1}과 F_t사이의 영상에서 찾아진 코너들을 매칭할 때 SSD를 사용함으로써 성능이 떨어지는 문제를 안고 있다. 인접한 영상 프레임의 밝기 값이 달라지는 경우로, [그림 3]과 같이 F_t이 F_{t-1} 보다 전체적으로 밝기 값이 달라지는 것을 말한다. Rosten의 경우 SSD를 사용하여 코너 매칭을 하고 있기 때문에 영상의 밝기 값에 커다란 영향을 받게 되고, 매칭시 전혀 다른 값을 가지게 되어 유사도는 떨어지게 된다.



(a) F_{t-1} (b) F_t

그림 3. 연속된 영상 프레임에서 밝기 값이 달라진 경우



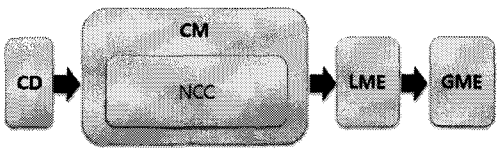
(a) F_{t-1} (b) F_t

그림 4. 연속된 영상 프레임에서 코너가 사라진 경우

또 다른 이유로는 [그림 4]의 (a)와 (b)처럼 F_{t-1}에서 찾아진 코너가 F_t에서는 검출되지 않고 사라진 경우이다. 이런 경우 코너 매칭할 때 F_t에서 찾아진 코너 후보들 중 F_{t-1}에서 찾아진 코너와 유사하지 않은 전혀 다른 코너를 매칭하게 되고, 다음 단계에서 모션 추정할 때 실제와는 전혀 다른 값을 출력하기 때문에 정확성이 떨어지게 된다.

2. 제안 알고리즘

이런 모든 문제를 해결하기위해 총 3개의 알고리즘을 제안한다.



CD(Corner Detection)
CM(Corner Matching)
LME(Local Motion Estimation)
GME(Global Motion Estimation)

그림 5. 제안 알고리즘 1

문제를 해결하기 위한 첫 번째 방법은 [그림 5]와 같이 Trajkovic의 Normalized Cross Correlation(NCC) 계수를 코너 매칭에 사용하는 것이다. 기존의 SSD 알고리즘은 영상의 밝기 값에 민감하므로, NCC 계수를 사용하여 영상의 밝기 값에 크게 영향을 받지 않고 코너들 사이에 상관관계 기반으로 매칭하게 되어, 보다 나은 매칭 정확성을 기대할 수 있다.

$$c(f,g)=\frac{\sum_{x=-n}^{n}\sum_{y=-n}^{n}(f(x,y)-\bar{f})(g(x,y)-\bar{g})}{\sqrt{\sum_{x=-n}^{n}\sum_{y=-n}^{n}(f(x,y)-\bar{f})^2}\sqrt{\sum_{x=-n}^{n}\sum_{y=-n}^{n}(g(x,y)-\bar{g})^2}} \qquad (3)$$

$f(x,y)$와 $g(x,y)$는 F_{t-1}과 F_t에서 해당 픽셀의 밝기 값을 의미하며, $\bar{f}$와 $\bar{g}$는 주위 픽셀들의 평균 밝기 값을 나타낸다. 이때 나온 NCC 계수 값은 -1과 1 사이의 값을 가지며, 1일 때 두 픽셀은 완벽한 유사도를 가진다.

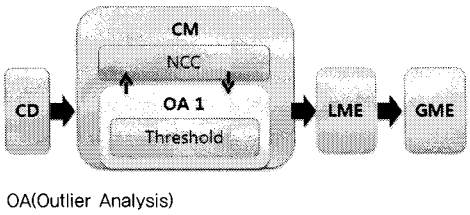


그림 6. 제안 알고리즘 2

하지만 NCC를 사용한 코너 매칭 또한 SSD와 마찬가지로, [그림 4]와 같이 F_{t-1}에서 찾아진 코너가 F_t에서 사라지는 경우에는 전혀 다른 코너를 선택하는 같은 문제점에 봉착하게 된다.

이런 문제를 해결하기 위해서 코너를 매칭할 때와 모션 추정할 때 두 단계의 외톨이 분석(outlier analysis)을 통하여 외톨이를 제거한다.

외톨이를 제거하는 첫 번째 방법은 [그림 6]에 제안한 알고리즘과 같이 NCC를 사용하여 코너를 매칭할 때, NCC 계수 c와 임의의 임계값 t를 비교하는 것이다. 만약 c가 t보다 큰 경우에는 두 개의 코너가 서로 유사할 확률이 높은 것으로 보고 매칭 정보 a를 취하고, 작은 경우에는 취하지 않는 것이다. 이때의 t값은 NCC 계수 c의 범위인 $-1 \le t \le 1$ 사이에 존재하는 임의의 값이며, 매칭 정보 a의 개수를 n이라 할 때 매칭 정보들의 집합 $S_a = \{ a_1, a_2, \ldots, a_n \}$로 나타낼 수 있다.

알고리즘 1. 첫 번째 외톨이 분석

입력 : NCC 계수 c, 매칭 정보 집합 S_a
　　임의의 임계값 t
출력 : Outlier가 제거된 매칭 정보 집합 S_β
알고리즘 :
1. // c를 구한 후
2. if ($c > t$)
3. 　　$S_\beta = a$;
　　// 서로 유사할 확률이 높은 매칭 정보

이렇게 얻어진 매칭 정보 집합 S_β는 이후 단계인 Local Motion Estimation(LME)에서 사용된다. LME는 각각의 매칭된 코너 간의 정보를 가지고 모션이 어떻게 움직였는지 추정하는 것이다. 이는 F_{t-1}의 코너 위치에서 F_t의 코너 위치의 차를 통하여 쉽게 구할 수 있다.

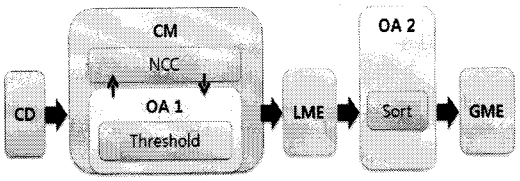


그림 7. 제안 알고리즘 3

두 번째 방법으로는 [그림 7]과 같이 LME를 통하여 얻어진 F_{t-1}과 F_t의 매칭된 각각의 코너들 사이에 이동한 거리 차의 크기를 정렬하여 매칭 정보 집합 S_β를 새롭게 얻을 수 있다. 만약 LME를 통하여 얻어진 매칭 정보의 개수 m이 임의의 수 l보다 크다면 m을 l로 나눈 수 k일 때 정렬된 S_β의 집합에서 k부터 $m - k$까지의 인덱스에 위치한 매칭 정보 집합 $S_\theta = \{\beta_k \ldots \beta_{m-k}\}$을 옳게 매칭된 코너로서 인정하는 것이다.

알고리즘 2. 두 번째 외톨이 분석

입력 : LME를 통하여 얻어진 매칭 정보 집합 S_β,
　　S_β의 원소 개수 m, 임의의 수 l
출력 : Outlier 제거된 매칭 정보 집합 S_θ
알고리즘 :
1. $k = 0$;
2. $S_{\beta'} = \mathrm{Sort}\ (\ S_\beta\ )$
3. if ($m > l$)
4. 　　$k = m / l$;
5. $S_\theta = \{\ \beta'_k \ldots \beta'_{m-k}\ \}$

이렇게 얻어진 매칭 정보 집합 S_θ는 이후 단계인 Global Motion Estimation(GME)에서 최종적으로 F_{t-1}에서 F_t으로 이동되는 방향을 추정할 수 있다.

Ⅳ. 실험 및 평가

(a) 영상1 (b) 영상2 (c) 영상3

그림 8. 실험에 사용된 영상 데이터

실험은 Intel Core2 Duo 2.33GHz CPU가 장착된 PC의 환경에서 이루어졌으며, 운영체제는 Windows XP sp3와 컴파일러는 Visual Studio 2008 C++을 사용하였다. 실험에 사용된 데이터로는 일반 웹캠으로부터 획득한 194개의 프레임과 139개의 프레임, 130개의 프레임으로 구성된 연속된 영상 데이터 3개를 사용하였다. 코너 검출에 사용된 알고리즘은 Rosten의 Open Sources를 사용하였고, Rosten의 SSD 코너 매칭과 3개의 제안한 알고리즘을 비교 분석하였다.

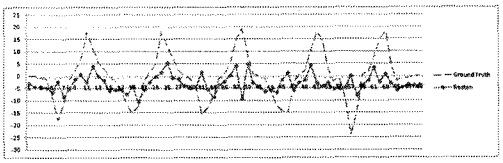
1. 성능 평가

실험 영상 데이터로부터 얻은 ground truth와 각 알고리즘의 모션 추정 결과를 비교하여 성능을 평가하였다.

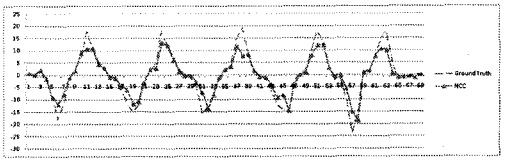
표 1. 에러량 비교(단위 : 픽셀)

		Rosten	NCC	NCC Threshold	NCC Sort
영상1	최대	9.0000	5.2000	5.2000	1.7500
	최소	0.0714	0.0000	0.0000	0.0000
	평균	2.4013	1.1171	0.6611	0.3467
영상2	최대	28.5000	11.3000	9.6250	6.1667
	최소	0.0385	0.0000	0.0000	0.0000
	평균	6.8650	2.1278	1.1061	0.5379
영상3	최대	25.0000	10.8696	7.3333	5.2500
	최소	0.0000	0.0000	0.0588	0.0000
	평균	8.2959	2.0542	1.1289	0.6770
전체 영상 (평균)		5.8541	1.7664	0.9654	0.5205

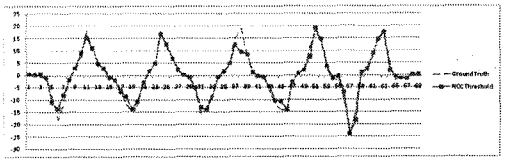
[표 1]에 나타난 것처럼 전체 영상의 평균 에러량은 Rosten이 약 5.8픽셀, NCC가 약 1.7픽셀, NCC Threshold가 약 0.9픽셀, NCC Sort가 약 0.5픽셀을 보였다. 위의 결과는 Rosten 알고리즘이 모션을 추정함에 있어서 비교적 커다란 오차를 보이는 반면, 제안한 알고리즘들은 순차적으로 평균 50%이상의 에러량이 감소하는 것을 보인다. 또한 [그림 9]에 나온 것처럼 각 알고리즘을 적용할 때마다 ground truth와 유사해지며 성능이 향상되는 것을 확인하였다.



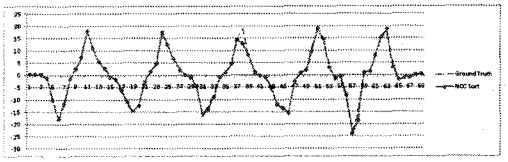
(a) Rosten



(b) NCC



(c) NCC Threshold



(d) NCC Sort

그림 9. 영상2 데이터의 ground truth와 각 알고리즘 비교

2. 속도 평가

Rosten 알고리즘과 제안한 알고리즘이 모션을 추정하는데 걸리는 수행 시간을 비교하기 위하여 각 영상에서 모든 프레임을 처리하는 전체 수행 시간과 프레임당 처리하는 수행 시간으로 구분하였다. 또한 각각의 영상 데이터별로 하나의 알고리즘당 총 10번씩의 반복 실험을 통하여 전체 수행 시간의 최대값과 최소값, 평균값을 구했고, 구해진 평균값에서 해당 영상의 프레임 수를 나누어 프레임당 수행 시간을 구했다.

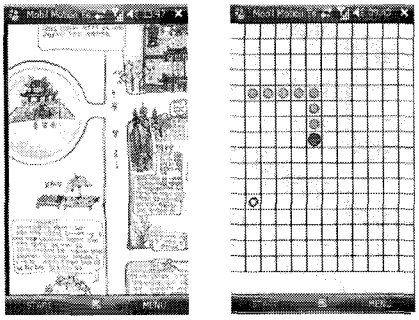
표 2. 속도 비교(단위 : 초)

			Rosten	NCC	NCC Threshold	NCC Sort
영상1	전체 수행 시간	최대	0.3440	0.4220	0.4070	0.4070
		최소	0.3120	0.4060	0.3900	0.4060
		평균	0.3296	0.4078	0.3969	0.4061
	프레임당 수행 시간		0.0034	0.0042	0.0041	0.0042
영상2	전체 수행 시간	최대	0.2650	0.4840	0.4840	0.4850
		최소	0.2500	0.4680	0.4680	0.4680
		평균	0.2530	0.4734	0.4703	0.4751
	프레임당 수행 시간		0.0036	0.0068	0.0067	0.0068
영상3	전체 수행 시간	최대	0.2190	0.2190	0.2190	0.2350
		최소	0.2030	0.2030	0.2030	0.2030
		평균	0.2171	0.2170	0.2170	0.2217
	프레임당 수행 시간		0.0033	0.0033	0.0033	0.0034
전체 영상	전체 수행 시간(평균)		**0.2666**	**0.3661**	**0.3614**	**0.3676**
	프레임당 수행 시간(평균)		**0.0035**	**0.0048**	**0.0047**	**0.0048**

[표 2]에 나타난 것처럼 Rosten의 알고리즘은 모든 영상 데이터에서 평균적으로 전체 수행 시간은 약 0.2666초, 프레임당 수행 시간은 약 0.0035초의 속도로 수행되었다. NCC만 사용한 알고리즘은 평균적으로 전체 수행 시간은 약 0.3661초, 프레임당 수행 시간은 약 0.0048초의 시간이 소요되었다. 이것은 Rosten 알고리즘 보다 전체 수행 시간은 약 0.995초, 프레임당 수행 시간은 약 0.0013 이내의 시간이 조금 더 소요되었다. NCC를 진행한 후 Threshold를 통하여 한 번의 외톨이 분석을 진행한 알고리즘은 NCC만 진행한 알고리즘보다 평균적으로 전체 수행 시간은 약 0.0047초 이내, 프레임당 수행 시간은 약 0.0001초 이내의 시간이 줄어들었다. NCC를 진행한 후 Threshold와 Sort를 통하여 두 번의 외톨이 분석을 적용한 알고리즘은 NCC만 사용한 알고리즘과 비슷한 시간의 수행 시간이 소요되었다.

3. 모바일 응용 프로그램에 적용

우리는 제안한 알고리즘을 검증하기 위하여 모바일 장치의 여러 응용 프로그램을 구현하고, 입력 인터페이스로 제안한 알고리즘을 적용하였다. 모바일 장치는 운영체제가 Windows Mobile 6.1 기반인 SCH-M490을 사용했고, 컴파일러는 Visual Studio 2008 C++을 사용하였다.

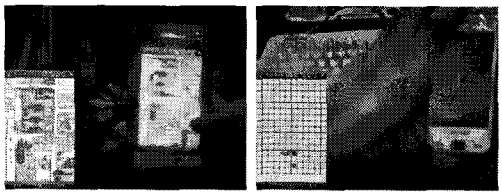


(a) 지도 브라우저 (b) 뱀 게임

그림 10. 응용 프로그램 UI

3.1 지도 브라우저

모바일 장치의 활용 범위가 점점 증가함에 따라 모바일 장치에서 지도를 통하여 현재 위치를 검색하거나 주변의 정보를 얻을 수 있다. 하지만 모바일 장치의 화면 크기는 한정되어있으며, 단순히 키패드만으로 커다란 크기의 지도를 조작하는 것이 불편할 수도 있다. 우리가 구현한 지도 브라우저는 모바일 장치를 상하 또는 좌우로 움직임으로써 해당 모션을 추정하고 이를 인터페이스로 사용하여, 커다란 크기의 지도 정보를 상하좌우로 스크롤하면서 사용자가 원하는 정보를 얻을 수 있도록 했다.



(a) 지도 브라우저 (b) 뱀 게임

그림 11. 응용 프로그램 사용 모습

3.2 뱀 게임

모바일 장치가 보편화되고 하드웨어가 발전함에 따라 모바일 장치를 통하여 게임을 즐기는 사람들이 급격히 증가하고 있다. 또한 이동통신사 및 개발사들은 이들을 위해 고성능의 3D 게임폰과 매우 많은 종류의 게임을 출시하고 있다. 이와 더불어 Wii[13]와 같이 사용자가 직접 모션을 취하면서 조작하는 체감형 게임이 흥행을 이끌고 있다. 이처럼 모션을 통한 인터페이스를 쉽고 재미있게 접근할 수 있는 분야가 바로 게임이다. 우리는 제안한 알고리즘을 통하여 사용자의 모션을 추정하고 얻어진 모션 정보를 입력 인터페이스로 사용하는 간단한 모바일 게임을 구현하였다. 게임은 많이 알려진 뱀 게임으로 뱀을 상하좌우로 제어하여 먹이를 먹으면 꼬리가 길어지는 게임이다.

V. 결론

Rosten의 알고리즘은 실시간에 사용할 수 있는 속도로서 매우 빠른 처리가 가능한 우수한 알고리즘이다. 하지만, 모션 추정을 하기 위한 코너 매칭 과정에서 연속된 프레임의 밝기 값이 변하거나, 코너가 사라지는 경우 영향을 많이 받아 정확성이 떨어지는 커다란 약점을 가지고 있다. 그렇지만, 본 논문에서 제안하는 알고리즘은 Rosten 알고리즘의 두 가지 약점을 극복하고, 모션 추정 성능이 매우 크게 향상됨을 실험 결과를 통하여 확인하였고, 실제 모바일 장치에 응용 프로그램을 구현하고 제안한 알고리즘을 적용함으로써 모바일 환경에서도 실시간으로 동작이 가능함을 보였다.

그러나 모션이 급격히 변하거나, 특징점인 코너를 찾기 어려운 환경에서는 모션을 추정하는데 영향을 미치며, 추후 이러한 문제에 대해 깊은 연구가 필요하다.

참 고 문 헌

[1] S. Bucolo, M. Billinghurst, and D. Sickinger, "User Experiences with Mobile Phone Camera Game Interfaces," Proceedings of the 4th international conference on Mobile and ubiquitous multimedia, pp.87-94, 2005.

[2] M. Hachet, J. Pouderoux, and P. Guitton, "A Camera-Based Interface for Interaction with Mobile Handheld Computers," Proceedings of the 2005 symposium on Interactive 3D graphics and games, pp.65-72, 2005.

[3] https://www.icg.tugraz.at/~daniel/HistoryOfMobileAR/

[4] http://handheld.softpedia.com/get/Games/Arcade/Marble-Revolution-21182.shtml

[5] D. Wigdor and R. Balakrishnan, "TiltText: Using Tilt for Text Input to Mobile Phones," Proceedings of the 16th annual ACM symposium on User interface software and technology, pp.81-90, 2003.

[6] S. A. Drab and N. M. Artner, "Motion Detection as Interaction Technique for Games & Applications on Mobile Devices," Proceedings of Workshop on Pervasive Mobile Interaction Devices (PERMID 2005), 2005.

[7] A. Haro, K. Mori, T. Capin, and S. Wilkinson, "Mobile Camera-based User Interaction," Proceedings of IEEE International Conference on Computer Vision Workshop on Human-Computer Interaction, pp.79-89, 2005.

[8] J. Wang, S. Zhai, and J. Canny, "Camera Phone Based Motion Sensing: Interaction Techniques, Applications and Performance Study," Proceedings

of the 19th annual ACM symposium on User interface software and technology, pp.101-110, 2006.

[9] J. Hannuksela, P. Sangi, and J. Heikkil, "Vision-based motion estimation for interaction with mobile devices," Computer Vision and Image Understanding, Vol.108, pp.188-195, 2007.

[10] M. W. Kim and I. S. Oh, "Estimation of Rapid-motion for Mobile Devices Using Temporal Coherence," Image and Vision Computing New Zealand, pp.1-6, 2008.

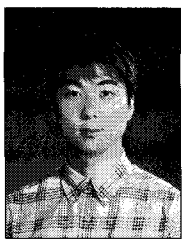
[11] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," In European Conference on Computer Vision, pp.430-443, 2006.

[12] M. Trajkovic and M. Hedley, "Fast Corner Detection," Image and Vision Computing, Vol.16, pp.75-87, 1998.

[13] http://www.nintendo.co.kr/Wii/main.php

저 자 소 개

김 준 호(Junho Kim) 준회원



- 2008년 2월 : 전북대학교 컴퓨터공학과(공학사)
- 2008년 3월 ~ 현재 : 전북대학교 컴퓨터공학전공 석사과정

<관심분야> : 패턴인식, 컴퓨터비전, 모바일

오 일 석(Il-Seok Oh) 정회원

- 1984년 2월 : 서울대학교컴퓨터공학과(공학사)
- 1992년 2월 : KAIST 전산학과(공학박사)
- 1992년 2월 ~ 현재 : 전북대학교 컴퓨터공학부 교수

<관심분야> : 문서영상처리, 패턴인식, 컴퓨터비전