제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003. 5)

ARM7 CPU를 위한 효율적인 지문인식 알고리즘

김준석*, 설승진**
*대민전자(주) 연구원
**대민전자(주) 연구소장
e-mail : juns16@freetouch.com

An Efficient Fingerprint Recognition Algorithm for ARM7 CPU

Joon-Seok Kim*, Seung-Jin Sul**
R&D Center of Daemin Electronics Co.,Ltd.

요 약

생체인식 분야 중에서 지문인식 시스템이 가장 대중화되었음에도 불구하고 아직까지 지문인식 장비는 고가인 실정이다. 이는 지문인식 시스템이 많은 이미지 처리와 수학 연산을 위해 고속의 CPU와 많은 기억장치를 요구하기 때문이다. 본 논문에서는 PC기반에서 사용되는 지문인식 알고리즘을 ARM7 CPU에 맞게 최적화하는 방법에 대해 알아보고, ARM7 CPU를 탑재한 지문인식 모듈에 적용하여 실행시간을 측정하여 성능을 평가하였다.

1. 서론

현재 생체인식 시장은 홍채인식, 지문인식, 얼굴인식, 장문인식, 음성인식 등이 주류를 이루고 있으며 [5], 특히 지문인식 시스템이 가장 대중화되어 PC보안, 도어락, 출입통제, 출결관리, 근태관리 등에 응용되고 있다.

그러나 지문인식 장비는 입력지문을 향상시키는 전처리나, 이진화, 세선화 등 많은 이미지 처리를 해야하고, 이 과정에서 많은 수학적인 연산이 필요하다[1][2][3][4]. 이런 많은 연산량에도 불구하고 빠른 실행속도를 갖게 하려면 고속의 CPU와 많은 기억장치를 탑재해야 하는데 이것은 곧 지문인식 장비의 가격 상승 요인이 된다.

본 논문에서는 저가의 지문인식 시스템을 구축할 수 있는 ARM7 CPU를 탑재한 지문인식 모듈을 소개하고, PC기반에서 사용되는 지문인식 알고리즘을 ARM7 CPU에서 빠르게 동작하도록 최적화하는 방법에 대해 알아본다. 그리고, ARM7 CPU를 탑재한 모듈에 최적화 된 알고리즘을 적용하였고, 150개의 지문을 등록한 후 특징 추출 시간과 인증 시간을 측정하여 성능을 평가하였다.

2. 연구 내용

센서로부터 입력받은 지문 영상으로부터 특징정보를 추출하는 과정은 크게 전처리, 이진화, 세선화, 특징추출, 잡음특징 제거로 나눌 수 있으며 각 단계에는 영상처리를 위한 마스크 연산과 sin, cos, atan2, sqrt 등 많은 수학 함수가 사용된다 [1][2][3][4]. 본 연구에서는 이미 개발된 자체 보유의 PC기반 지문인식 알고리즘을 ARM7 프로세서를 탑재한 지문인식 모듈을 위해 최적화하였다. PC기반 지문인식 알고리즘에서 사용된 많은 연산의 횟수를 가능한 적게 하였으며, 부동 소수점 연산을 등가의 정수 연산으로 대체하였다. 컴파일러에서 제공하는 느린 수학함수는 참조 테이블(Lookup Table)로 바꾸거나 더 빠르게 동작하는 함수를 자체 제작하여

사용하였고, 테스트 결과 전체 성능에 크게 기여하지 못하는 루틴은 삭제하여 속도를 향상시켰다.

속도뿐만 아니라 적은 메모리 사용을 위해 "버퍼 재사용" 기법을 사용했다. "버퍼 재사용"이란 이미 할당된 버퍼가 현재 과정에서 쓰이지 않을 때 그 버퍼를 다른 용도로 할당해서 쓰는 것을 말한다. 특징 추출 과정뿐만 아니라 특징 매치 과정에서도 방향제한, 종류 비교 등을 통해서 특징 비교 시간을 빠르게 했다.

3. 실행 속도 개선

지문인식 알고리즘에서 가장 속도를 많이 차지하는 부분은 sin, cos, atan2, sqrt와 같은 수학 함수이다. 따라서 속도 개선을 위해 가능한 위 함수 사용을 제한하였고 꼭 필요한 부분에서는 가장 빨리 동작할 수 있는 방법을 택하였다.

sin과 cos함수는 참조 테이블로 교체하였다. 함수의 호출에는 기본적인 문맥 교환 시간(context switching time)이 소요되고, 삼각함수는 내부적으로 많은 부동소수점 연산을 포함하기 때문에 ARM7 프로세서에서 매우 느리게 동작한다. 삼각함수를 참조 테이블로 교체하면 문맥 교환 시간이 없어지고, 내부의 부동소수점 연산도 필요 없으므로 많은 속도 개선을 이룰 수 있다.

atan2함수는 입력되는 인자의 다양함으로 인해 참조 테이블로 변환하기가 어렵다. 그래서 atan2함수는 함수의 정의에 따라 직접 함수를 작성하였다. 이 또한 컴파일러에서 제공하는 내장 수학 함수에 비해 빠른 성능을 보였다.

sqrt함수는 주로 두 점간의 거리를 계산할 때 사용되는데 문맥을 적절히 변형하면 사용을 피할 수 있다.

```
for(i=0; i < index2; i++)
{
 for(k= -10; k <11; k++)
 {
   for(l=-10; l <11; l++)
   {
     distance = l * l + k * k;
     fdistance = sqrt(distance);
     if(fdistance < 10) Buffer[(y+k) * N + x + l]=
                        (BYTE)(i+1);
   }
 }
}
```

[그림1] sqrt함수를 사용한 경우

```
for(i=0; i < index2; i++)
{
 for(k= -10; k <11; k++)
 {
   for(l=-10; l <11; l++)
   {
     distance = l * l + k * k;
     if(distance < 100) Buffer[(y+k) * N + x + l]=
                        (BYTE)(i+1);
   }
 }
}
```

[그림2] sqrt함수를 사용하지 않은 경우

영상의 향상이나 방향성 라플라시안 필터를 사용한 이진화 등의 연산을 할 경우 각 픽셀에 특정 크기의 마스크를 적용하게된다. 이것은 전체 이미지의 픽셀 수와 마스크의 크기에 따라 매우 많은 곱셈과 덧셈 연산을 수반하게 되는데 여기서 연산의 양을 줄이는 것이 속도 향상에 크게 도움이 된다. 연산의 양을 줄이는 방법으로는 마스크의 가중치 값이 0 인 것을 곱셈에서 제외하는 방법과 수식을 변형하는 두 가지 방법을 사용하였다. 가중치가 0인 것을 제외하는 방법으로는 곱셈 연산을 for 루프를 통하지 않고 직접 전개해서 수행할 수 있다. 다음은 수식을 변형하여 속도를 개선한 예이다.

```
val += 10 * Buffer[pos - 2];
val += 3  * Buffer[pos - 1];
val += 10 * Buffer[pos    ];
val += 3  * Buffer[pos + 1];
val += 10 * Buffer[pos + 2];
```

[그림3] 수식을 변형하기 전

```
val +=10*(Buffer[pos-2]+Buffer[pos]+Buffer[pos+2]);
val +=3*(Buffer[pos-1] + Buffer[pos+1]);
```

[그림4] 수식을 변형한 후

수식을 변형할 때에는 수식의 의미가 변하지 않도록 주의해야 한다.

특징 템플릿을 서로 비교해서 같은 지문인지를 판별하는 매치 함수도 빠른 속도를 필요로 한다. 매치

함수의 수행 속도를 빠르게 하기 위해서 방향 제한 방법을 사용했다. 도어락과 같은 실제 응용에서 지문은 외부 구조물에 의해 일정 각도 이상으로 회전하기가 힘들다. 따라서 알고리즘에서는 두 템플릿을 비교할 때 기준 특징점의 각도 제한을 두어 두 기준 특징점의 각도 차가 특정 값 이상이면 비교를 하지 않는 방법을 사용하였다.

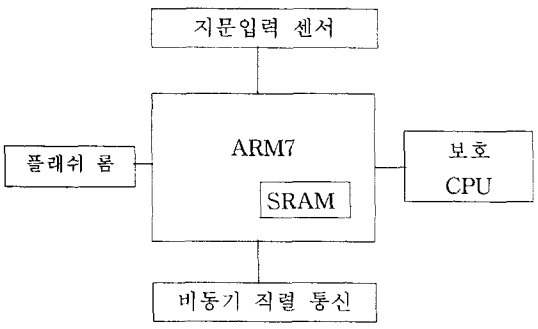
4. 실험

실험에는 자체 개발한 DFZ-2002S 광학식 모듈을 사용하였다.

CPU	32 Bit RISC CPU(66MHz)
Interface	Serial(TTL Level)
동작 전압	DC +3.3V ±0.3V
동작 전류	< 120mA
동작 온도 범위	-20℃ < T < +60℃
SRAM / Flash ROM	256KB / 256KB

[표1] 실험에 사용된 모듈의 사양

DFZ-2002S 모듈은 ARM7 CPU를 탑재하였고, 적은 용량의 SRAM을 사용하여 속도를 향상시켰다. 또 별도의 보호 CPU를 탑재하여 내부 데이터를 보호하도록 했고, 경통에 PCB를 부착해서 크기를 소형화했다. 일반적인 지문인식 모듈은 장치를 사용하지 않을 경우 어느 정도의 전력을 소모하는 반면, DFZ-2002S는 일정시간 입력이 없으면 전원공급을 완전히 차단해서 전력을 소모하지 않는다. 그리고 사용자의 입력이 들어오면 매우 빠른 속도로 부팅해서 입력에 응답하게 된다.



[그림5] DFZ-2002S Block Diagram

실험은 모두 150개의 지문을 모듈에 등록한 다음 지문입력에서부터 인증 결과를 출력할 때까지의 시간을 측정하였다. 모두 5번에 걸쳐서 측정했으며 결과값으로 이들의 평균값을 선택했다. 시간 측정은 다음과 같이 수행했다. PC의 시간 측정 프로그램에서 모듈에 특징 비교 명령을 전송한다. 모듈이 이미지를 읽은 후 [신호1]을 PC에 전송한다. PC에서는 [신호1을] 받으면 타이머를 가동한다. 모듈에서 특징점 추출이 끝나면 [신호2]를 PC에 전송한다. PC는 [신호2]를 받으면 타이머를 중지하고 현재 시간을 특징 추출 시간으로 저장하고, 새로운 타이머를 가동한다. 모듈이 150개의 특징점 비교를 모두 끝내면 PC에 [신호3]을 전송한다. [신호3]을 받은 PC는 타이머를 중지하고 그 값을 150개의 특징 비교 시간으로 저장한다.

시도 횟수	특징 추출 시간	인증시간(전체 / 개당)
1	1.2초	2.25초 / 0.0150초
2	1.1초	1.87초 / 0.0124초
3	1.2초	2.31초 / 0.0154초
4	1.4초	2.27초 / 0.0151초
5	1.2초	1.95초 / 0.0130초
결과(평균)	1.22초	2.13초 / 0.01418초

[표2] 수행시간 결과

실험에서 특징 데이터 1개당 수행시간은 150개 특징 비교시간을 구한 후 150으로 나눈 값으로 정했다. 그 이유는 개당 수행시간을 측정하기 위해 모듈에서 PC로 신호를 보낼 때 소비되는 시간이 오차로 작용할 수 있으므로 이를 제거하기 위함이다.

4. 결론 및 향후 연구과제

PC기반 지문인식 알고리즘은 고속의 CPU와 메모리 사용에 제한이 없는 상태에서 동작한다. 하지만 저가의 독립형 제품을 만들기 위해서는 저속 CPU와 적은 메모리를 탑재해야만 하는데 이런 낮은 하드웨어 사양에서 빠르게 동작하도록 하려면 알고리즘을 최적화하여 CPU 사용률을 낮추어야 한다. 본 연구에서는 지문인식 알고리즘에 사용되는 삼각함수를 참조 테이블로 변환하였고, 느린 내장함수를 자체 개발 함수로 대체, 그리고 연산 수식 최적화 기법과

부동 소수점 연산 제거를 통해서 ARM7 CPU를 채택한 지문인식 모듈에서 빠르게 동작하는 알고리즘 개발에 성공하였다. 특징 추출 시간이 평균 1.22초, 개당 인증 시간이 평균 0.01418초로 모듈에 지문이 150개 등록 되어있을 때 약 3.3초 정도에 인증이 완료 될 수 있다.

향후 연구 과제는 현재 특징점 추출 시간이 평균 1.22초인데 이를 0.8초대로 낮추는 것이다. 이를 위해서는 현재 지문의 융선과 골을 분리하여 이진화를 수행하는 라플라시안 필터의 효율적이 개선이 필요하다.

[참고문헌]

[1] N. K. Ratha, K. Karu, S. Chen, and A. K. Jain, "A Real-Time Matching System for Large Fingerprint Databases", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol.18, No.8, pp.799-813, 1996.

[2] A. K. Jain, L. Hong, and R. Bolle, "On-Line Fingerprint Verification", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol.19, No.4, pp.302-314, 1997.

[3] A. K. Jain, L. Hong, S. Pankanti, and R. Bolle, "An Identity Authentication System Using Fingerprints", Proc. IEEE, Vol.85, No.9, pp.1365-1388, 1997.

[4] A. K. Jain, L. Hong, S. Pankanti, and R. Bolle, "An Identity Authentication System Using Fingerprints", Proc. IEEE, Vol.85, No.9, pp.1365-1388, 1997.

[5]http://www.kisa.or.kr/K_trend/KisaNews/200112/special_report_04.html, 2003. 3