고속 저전력 지문인식 알고리즘 처리용 회로

유민희*, 정승민*

*한신대학교

High Speed and Low Power Scheme for a Fingerprint Identification Algorithm

Min-Hee Yoo*, Seung-Min Jung*

* Hanshin University

E-mail : jasmin@hs.ac.kr

요 약

본 논문에서는 특징점 기반의 지문인식 알고리즘의 각 단계에 있어서 32-bit CPU 사이클 점유율을 분석하고 그 중 전체 80%를 차지하고 있는 가보필터링과 세선화 단계를 처리하기 위한 전용 하드웨어 구조를 제안한다. 특징점 기반의 지문인식 알고리즘을 개발하였으며 소스 코드를 분석하여 가보필터링과 세선화 단계의 처리를 마이크로프로세서가 처리하지 않고 바이패싱하기 위한 전용 하드웨어를 위한 선행연구를 ARM 에뮬레이터 환경에서 실시하였다.

ABSTRACT

This paper proposes an effective hardware scheme for gabor filter and thinning stage processing of a fingerprint identification algorithm based on minutiae with 80% cycle occupation of 32-bit RISC microprocessor. The algorithm was developed based on minutiae with bifurcation and ending point. The analysis of an algorithm source code was performed using ARM emulator.

키워드

지문인식알고리즘, 세선화, 가보필터, 로직설계, ARM 프로세서

Ⅰ. 서 론

지문은 인간의 손가락 끝에 있는 융선(ridge)과 골(valley) 패턴들의 집합이며 개인을 타인으로부터 구별할 수 있게 하는 고유의 특성을 가진다. 융선 특징들을 특징점이라 부르며 단점과 분기점이 있다. 지문 인식의 가장 중요한 단계는 시스템에 입력된 지문 이미지로부터 알고리즘이 특징점 들을 신뢰성 있게 추출하는 것이다.

지문인식 시스템은 그림1과 같이 32 bit 이상의 고성능 마이크로프로세서를 내장[1][6] 혹은 외장[2] 탑재하여 알고리즘을 처리하도록 구현되어 있다. 일반적으로 적용되는 알고리즘은 특징점(minutiae) 추출방식으로서 전형적 가보필터 기반 (TGF: Traditional Gabor Filter)의 가우스 저역 통과 필터를 사용하여 지문 이미지의 잡음을 줄인 후에 융선의 방향과 주파수를 정확히 추출하는 방법이 적용된다[3][4].

가보필터 기반 알고리즘은 그림 2에서처럼 먼저 가우스 저역 통과 필터를 사용하여 지문 이미지의 잡음을 줄인 후에 융선의 방향과 주파수를 정확히 추출하는 방법이 적용된다[3]. 가보 필터는 선택적 주파수와 방향의 성질을 갖는다. 앞서 구한 방향 정보와 주파수 정보를 아래의 식 1,2를 이용하여 가보 필터 계수를 구한 후, 가보 필터링을 수행한다. 여기서 Φ은 가보 필터의 방향이고 f은 코싸인파의 주파수로, 각각은 현재 중심 픽셀이 속해있는 블록의 방향 값과 주파수 값과 같다. δ_x^2와 δ_y^2은 각각 x축과 y축으로 가우시안 포물선의 공간상수이다.

$$h(x,y:\Phi,f)=\exp\left\{-\frac{1}{2}\left[\frac{x_\phi^2}{\delta_x^2}+\frac{y_\phi^2}{\delta_y^2}\right]\right\}\cos\left(2\pi f x_\phi\right), \quad (1)$$

$$x_\phi = x\cos\phi + y\sin\phi\,,\quad y_\phi = -x\sin\phi + y\cos\phi \quad (2)$$

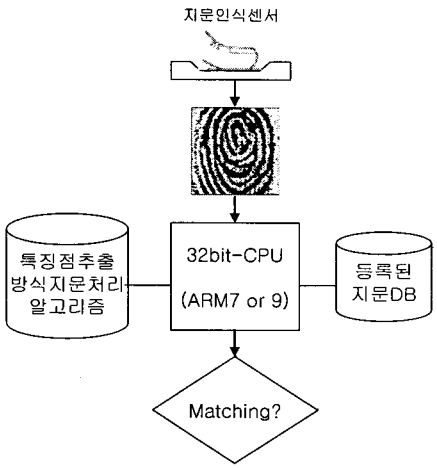

그림 1. 기존 지문인식 시스템 구성도

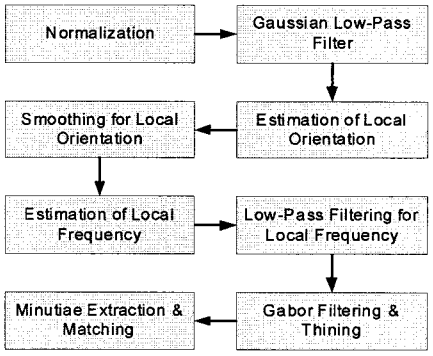

그림 2. 특징점기반 지문 인식의 알고리즘의 흐름도

가보필터는 대역통과 필터로, 융선과 골을 보호하면서 noise를 제거하는 융선 추출 방법으로, 그림 3의 원영상 처럼 주름, 땀샘, 상처로 떨어져 있는 융선을 연결시켜 주고 있는 것을 볼 수 있다. 세선화(Thinning) 단계에서는 그림 3의 융선 추출 영상과 같이 두꺼운 영상을 한 픽셀 두께를 갖도록 융선을 세선화 한다.

II. 특징점 기반 알고리즘 분석

원영상 => 융선 추출 영상 => 세선화 영상

그림 3. 가보필터 및 세선화 결과

알고리즘의 각 단계별 마이크로프로세서 점유율 분석을 위하여 ARM 에뮬레이터의 내장 ARM7TDMI 코어[5]를 적용하여 얻은 각 단계별 프로그램의 통과 여부 및 특징점 수 등에 대한 결과를 비교해 나가면서 분석 하였다. 입력지문 이미지는 IEEE 표준 공인 데이터인 FVC2002의 DB3를 이용하였다. 알고리즘의 각 단계에서 사용되는 명령어 분포와 소요되는 사이클 수는 다음과 같다.

<> 알고리즘의 마이크로프로세서 수행 명령 사이클

S-cycles : Sequential cycles. 마이크로프로세서는 선행 번지이후 워드 혹은 반워드가 있는 번지로 혹은 동일 번지로 전송요청

N-cycles : Non-sequential cycles. 마이크로프로세서는 선행 사이클에서 사용된 번지와 무관한 번지로 전송요청

I-cycles : Internal cycles. 마이크로프로세서는 내부 명령을 수행하고 있으므로 전송을 요청안함

C-cycles : Coprocessor cycles

<> 알고리즘의 처리단계별 명칭

QC : Quality Check, 이미지 질 체크

BO : Block Orientation, 블록 방향계산

SBO : Smooth Block Orientation

RF : Ridge Frequency, 융선 주파수 계산

FGF : Frequency Gaussian Filter, 가우시안 필터링

FLPF : Frequency LowPass Filter

GF : Gabor Filter

T : Thinning, 세선화

MD : Minutia Detection, 특징점 추출

DFM : Detect False Minutia, 의사특징점 제거

RM : Read Minutiae from a file, 등록지문읽기

MP : Matching Processing, 일치성 검사

표 1과 그림 4는 ARM 에뮬레이터를 이용하여 수행한 알고리즘 처리결과를 나타낸다. 알고리즘의 단계별 마이크로프로세서 사이클 수를 비교해 보면 가보필터(GB)와 세선화(T)가 점유하는 비율이 전체 80%이다. 클럭 속도를 30MHz에서 40MHz까지 변화시켰을때 전체 처리속도는 1.5초에서 1.125초로 나타났다.

표 1. 알고리즘단계별 소요사이클 수 및 속도

처리단계	QC	BO	SBO	RF	FGF	FLPF	GF	T	MD	DFM	MP	전체클럭수
명령어	215059	1,375,079	253,535	2,918,970	13,029	107,652	12,496,058	9,548,256	552,487	443,587	81,920	27,790,573
S-cycles	245781	1,414,353	304,618	2,873,557	14,461	114,166	12,838,764	9,602,337	667,456	489,315	112,700	28,431,727
N-cycles	61446	366,037	99,830	847,425	3,174	14,407	3,213,533	5,635,665	197,734	200,131	3,797	10,581,733
I-cycles	30721	429,333	34,733	487,788	2,365	5,023	2,250,996	2,597,056	71,698	100,584	7,972	5,987,548
C-cycles	0	0	0	0	0	0	0	0	0	0	0	0
Total	337,948	2,209,723	439,181	4,208,770	20,000	133,596	18,303,293	17,835,058	936,888	790,030	124,469	45,001,008
%	0.75%	4.91%	0.98%	9.35%	0.04%	0.30%	40.67%	39.63%	2.08%	1.76%	0.28%	100.00%
입력클럭	처리속도(초)											
40Mhz	0.008	0.055	0.011	0.105	0.001	0.003	0.458	0.446	0.023	0.020	0.003	1.125
35Mhz	0.010	0.063	0.013	0.120	0.001	0.004	0.523	0.510	0.027	0.023	0.004	1.286
30Mhz	0.011	0.074	0.015	0.140	0.001	0.004	0.610	0.595	0.031	0.026	0.004	1.500

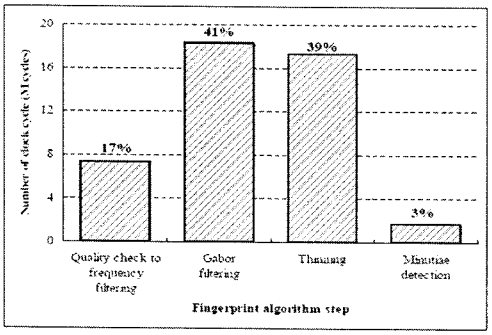

그림 4. 알고리즘 각 단계별 마이크로프로세서 사용 점유율 선행연구결과 (ARM7TDMI기반 ARM emulator 수행결과).

```
UINT8* GaborFilter(UINT8 *img,UINT8 *Oblocks, INT8 *Fblocks)
{

    INT8 m,n,fre;
    INT16 temp,temp1,temp2;
    for( i=0; i<No_Brows; i+=1){
        for(j=0; j<No_Bcols; j+=1){

            degree =(INT16) Oblocks[i*No_Bcols+j];
            fre =Fblocks[i*No_Bcols+j];
            cosv=cos_fix(degree);
            sinv=sin_fix(degree);

            kv=(-7)*cosv;
            ku=-(-7)*sinv;
            v[0][0] = kv+(-7)*sinv;
            u[0][0] = ku+(-7)*cosv;
            degree = (2*fre*(-7)*180) >> 7;
            degree = degree;
            g[0][0] = (h[0][0]*cos_fix(degree)) >> 12;
            v[0][1] = kv+(-5)*sinv;
            u[0][1] = ku+(-5)*cosv;
```

그림 5. 가보필터의 소스코드 패턴

III. 제안된 지문인식 시스템

본 연구에서 제안하고자 하는 부분은 전체사이클의 80%이상 마이크로프로세서를 점유하고 있는 가보필터 및 세선화 단계가 전체 이미지 픽셀을 스캔하면서 지극히 단순한 루틴을 반복적으로 수행하고 있다는 점에 착안 이 두 과정을 소프트웨어 적으로 32-bit 마이크로프로세서에서 수행하지 않고 하드웨어 적으로 처리하기 위한 전용 연산칩을 개발하고자 하는 데 있다.

코드구현의 관점에서 보면 가보필터링은 지문센서로부터 얻어진 원 지문이미지를 연속성과 비연속성으로 구분하고 지문의 골(ridge)의 밀도(frequency)를 고려하여 전체적으로 유연 화하는 단계로서 그림 5와 같이 각 픽셀이 소속된 윈도우에 대하여 방대한 양의 행렬 계산을 반복적으로 수행한다. 세선화는 특징점 추출을 위하여 두꺼운 지문패턴의 껍질을 단계적으로 벗겨내는 과정으로 가보필터링 보다도 더 단순하여 그림 6와 같이 for 문과 if 문의반복 수행을 역시 방대하게

```
void Thinning(UINT8 * img)
{
        UINT8 i,j,OK=1,turn=0;
        ..........,
        if( img[temp2+i]==1) {

        np1=img[temp1+(i-1)]+img[temp1+i]+img[temp1+(i+1)]+
        img[temp2+(i-1)]+img[temp2+(i+1)]+img[temp3+(i-1)]+
                img[temp3+i]+img[temp3+(i+1)];

                if(np1 > 2 && np1 < 8){
                y[0]= img[temp1+(i-1)];
                y[1]= img[temp1+i];
                if(y[0]==0 && y[1]==1) sp1++;
                if(y[1]==0 && y[2]==1) sp1++;
,,,,,,,,,,,,,,,,,,
                                if(sp1==1) {
                if(turn==0 && (y[3]==0 || y[5]==0||(y[1]==0
                && y[7]==0))){
                dst_Image[temp2+i]=0; OK=0;}
                else if(turn==1 && (y[1]==0 || y[7]==0
                ||(y[3]==0 && y[5]==0))){

        dst_Image[temp2+i]=0;OK=0; }

        }
```

그림 6. 세선화의 소스코드 패턴

수행한다. 가보필터 및 세선화 단계의 연산과정을 하드웨어기술언어(HDL)을 이용하여 RTL 수준에서 구현하고 로직 합성을 통하여 전용 연산

칩을 개발함으로써 현재 전체 알고리즘 단계에 소요되는 마이크로프로세서 총 사이클 수 대비 약 40% 까지 줄여 인증속도를 크게 향상시키고자 한다. 또한 전체 사이클 수가 대폭 줄어들게 됨으로써 더 이상 많은 게이트를 차지할 뿐만아니라 전력소모가 많은 고성능의 32-bit 마이크로프로세서 대신 16-bit 이하의 저전력 소형 마이크로프로세서를 적용할 수 있게 된다. 이럴 경우 본 연구를 통하여 개발된 지문인식 전용 연산 칩과 8-bit 혹은 16-bit 내장형 마이크로프로세서를 집적화하여 칩 사이즈를 줄이고 전체 지문인증시스템을 단순화 하는 효과도 기대된다. 전용칩의 개발을 통하여 지문인증처리속도의 향상, 전력소모의 감소, 칩 사이즈의 감소 및 시스템의 소형화가 가능하다. 그림 7은 본 연구에서 제안하고 있는 지문인식 시스템이다.

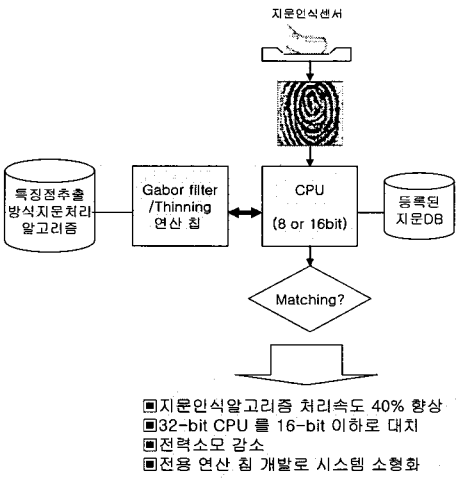

그림 7. 제안된 지문인식 시스템 구성도

IV. 결 론

본 논문에서는 특징점기반의 지문인식 알고리즘의 각 단계별 처리 과정에서 32비트 명령어 축약형 마이크로프로세서의 전체 사이클의 분포를 분석하였다. 분석결과 전체 80%이상을 융선 이미지 추출단계인 가보필터와 세선화 단계가 점유하고 있으며 소스 코드상에서 이들의 구현이 이미지 어레이의 단순계산으로 이루어져 있음을 확인하였다. 따라서 이를 RTL 수준의 HDL 코드로 기술하여 로직을 구현함으로써 하드웨어적으로 처리한다면 저성능 저전력 마이크로프로세서를 적용하면서도 처리속도와 전력소모에 있어서 향상을 기대할 수 있을 것으로 보인다. 본 연구는 향후 하드웨어와 알고리즘간의 인터페이스관련 연구를 통하여 시스템으로 구현 검증될 예정이다.

참고문헌

[1]Seung-Min Jung, Jin-Moon Nam, Dong-Hoon Yang and M. K. Lee, "A CMOS Integrated Capacitive Fingerprint Sensor with 32-bit RISC Microcontroller," *IEEE Journal of Solid-state Circuit*, Vol. 40, No. 8, pp. 1745-1750, 2005.

[2]슈프리마(주) 홈페이지, http://www.supremainc.com/korean/

[3]Lin Hong, Yifei Wan, and Anil Jain, "Fingerprint Image Enhancement: Algorithm and Performance Evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 8, pp. 777-789, August 1998.

[4]Chul-Heui Lee, Min-Seob Lee, "An Efficient Fingerprint Image Classification using Singular Points and Gabor filter," *Journal of Telecommunications and Information*, vol. 7, 2003.

[5]""ARM7TDMI," Technical Reference Manual, Rev. 4, ARM Limited, 2001.

[6]정성익, 외, "이미지 처리 및 인증을 위한 지문센서 개발,"정보통신부 정보통신산업기술 개발사업 최종보고서, 2004년 6월.