The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 17, No. 5, pp.165-171, Oct. 31, 2017. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2017.17.5.165
JIIBC 2017-5-23

펜학습기의 도트 패턴 인식을 위한 새로운 알고리즘

A Novel Algorithm for Dot Pattern Recognition with Reading Pen

박혜빈*, 정진우**

Hye-bin Park*, Jinoo Joung**

요 약 펜학습기는 교육 분야에서 주로 사용되며 급격히 성장하고 있는 기기로 종이 위에 인쇄된 미세한 도트패턴을 식별하여 코드로 변환한 후 코드와 매칭되는 음원을 실시간으로 재생한다. 입력된 도트패턴 영상을 코드로 변환하는 것이 펜학습기에서 가장 중요한 기능이므로 본 논문에서는 이 부분에 초점을 맞추어 알고리즘을 고안하였다. 해당 기술에 대한 알고리즘은 기존에 전혀 공개된 바 없으며 본 논문은 이러한 문제를 해결하려는 최초의 공개적인 시도이다. 실제 환경에서 성능을 평가한 결과 85% 패턴 인식률을 확인하였다.

Abstract Reading Pen is used mostly in educational purpose, yet its possibility is great as an input device for various IT devices. Reading Pen recognizes tiny dot patterns printed in papers, converts the patterns to corresponding codes, and plays audio contents that match the codes. Recognizing dot patterns and converting to the codes are the main function of Reading Pens, on which we focus and propose a novel algorithm. Such technology is never open to public and this paper is the first open attempt in this area. We test the algorithm and obtain 85% recognition ratio.

Key Words : Reading Pen, Dot Pattern, Image Recognition

Ⅰ. 서 론

교육용 펜학습기는 보이스펜, reading pen, 혹은 재능펜으로도 불리는 교육용 보조기구로 끝부분에 렌즈가 달려있어 눈으로는 판별되지 않는 0.05mm 내외의 직경을 가지는 작은 도트 이미지를 판독한다. 펜학습기는 종이 위에 인쇄된 미세한 디지털패턴을 광학렌즈를 통해 식별·인식하여 특정 데이터("코드")로 변환한 후 이 코드와 매칭 되는 음원을 실시간으로 재생한다[1]. 이때 교재가 낙서, 물, 기름 등에 의해 오염된 경우 혹은 저가 인쇄지를 사용하여 도트패턴에 번짐이나 소실이 발생한 경우, 펜학습기의 인식률이 70%이하로 낮아진다.

국내시장에서 사용되고 있는 도트패턴은 아래 도시된 그림과 같이 14번으로 명시된 블록의 도트들(7개)로 격자(프레임)구조와 방향성(16'번 도트가 주요 역할)을 표시한다[2]. 따라서 하나의 프레임에 있는 16개 도트 중 7개는 헤더이고 나머지 9개가 유효한 도트이다.

*상명대학교 컴퓨터과학과
**교신저자, 상명대학교 휴먼지능정보공학과
접수일자: 2017년 7월 15일, 수정완료: 2017년 9월 15일
게재확정일자: 2017년 10월 13일

Received: 15 July, 2017 / Revised: 15 September, 2017 / Accepted: 13 October, 2017
*Corresponding Author: jjoung@smu.ac.kr
Dept of Human Intelligence and Information Engineering, Sangmyung University, Korea

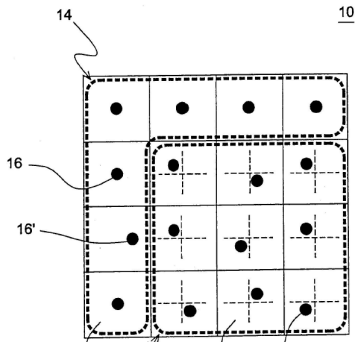

그림 1. 국내 시장에서 사용되는 도트패턴
Fig. 1. Dot pattern used in Korean market

본 논문에서는 위의 도트패턴을 인식하여 적절한 코드를 출력하는 영상처리 알고리즘을 고안하였다. 해당 분야의 기술은 기존에 공개된 바 없으며 본 논문은 이러한 문제를 해결하려는 최초의 공개적인 시도이다. 2장에서 제안하는 알고리즘을 설명하고, 3장에서는 제안한 알고리즘의 인식 정확도를 측정했으며 마지막으로 4장에서 결론을 맺는다.

II. 영상처리 알고리즘

본 논문에서는 그림 1의 상용 도트패턴을 인식하여 대응되는 코드를 생성하는 알고리즘을 제안한다. 다음은 영상처리 알고리즘의 흐름을 나타낸 그림이다.

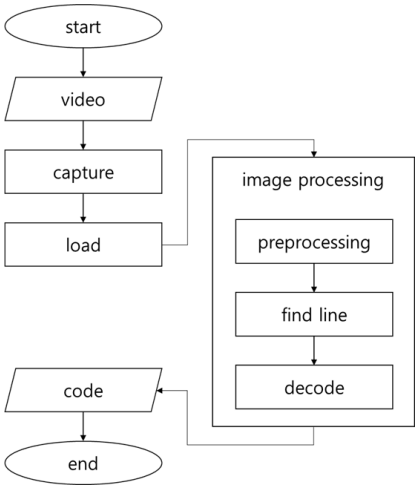

그림 2. 영상처리 알고리즘 흐름도
Fig. 2. Image processing algorithm flowchart

영상 입력장치로부터 영상이 들어오면 이것을 buffer에 저장(capture)한 후 저장된 이미지 데이터를 가져와 배열에 저장하고(load), 여러 영상처리 알고리즘을 적용하여 이미지에서 도트패턴을 찾고, 이에 대응되는 코드를 생성한다(image processing). Image processing 단계의 경우 입력 이미지를 가공하는 전처리과정(preprocessing), 전체 이미지에서 직선을 찾는 과정(find line), 찾은 직선을 바탕으로 한 frame의 도트패턴을 찾아 이를 코드화하는 과정(decode)으로 이루어진다.

1. preprocessing

전처리과정은 입력 이미지에서 도트와 배경을 구분하기 위해 시행되며, 본 연구에서는 색상 정보를 이용하지 않기 때문에 먼저 RGB 포맷인 이미지를 grayscale 영상으로 변환한다. 변환된 이미지는 자동으로 임계치를 결정해주는 Otsu 알고리즘[3]을 사용하여 배경은 255, 도트가 존재하는 부분은 0으로 이진화한다. Otsu 알고리즘으로 얻은 이진영상은 단일 임계치를 사용하기 때문에, 영상 내 노이즈가 존재하고 도트 영역이 불완전할 수 있다. 따라서 열기와 닫기연산을 차례로 적용하는 모폴로지연산을 수행하여 노이즈를 제거하고 도트 영역에 존재하는 빈틈을 메꾸었다[4][5]. 모폴로지연산을 적용한 결과영상에서 도트영역으로 판별된 부분을 구분하기 위해서 labeling 기법을 적용하여 도트영역별로 label을 매긴다[6]. 다음은 전처리과정에 따른 원본 영상 이미지의 변화를 나타낸 그림이다.

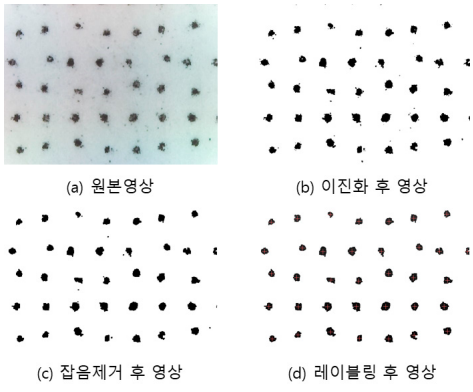

그림 3. 전처리 과정
Fig. 3. Preprocessing process

2. find line

기존의 도트패턴은 그림 1에서 볼 수 있듯이 헤더부분과 콘텐츠부분으로 나뉘는 데[5], frame의 맨 위 도트들이 일직선상에 배치되어 있다는 것을 확인 할 수 있다. 따라서 영상에서 일직선(frame line)을 이루는 도트들(frame dots)을 찾으며 한 frame의 도트패턴을 찾을 수 있다.

허프 변환(Hough transform)은 영상처리에서 직선을 찾기 위해 널리 사용되는 방법으로 2차원 영상좌표에서의 직선의 방정식을 파라미터공간으로 변환하여 직선을 찾는다[7]. 허프 변환을 사용하면 임계치 이상인 직선을 모두 찾기 때문에 다음과 같이 여러 개의 직선이 검출된다는 문제가 있다. 여러 개의 직선 중에서 실제로 필요한 것은 frame line 하나이기 때문에 후처리를 통해 나머지 직선을 제거할 필요가 있다.

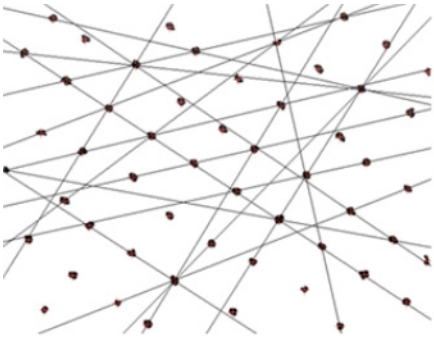

그림 4. 허프 변환 결과
Fig. 4. Hough transfrom result

(1) 중복 직선 제거

3개 이상의 도트가 일직선상에 배치되면, 이를 직선으로 판별하도록 허프 변환을 구현했기 때문에, 일직선상에 4개 이상의 도트가 존재할 경우 이를 1개가 아닌 2개 이상의 직선으로 판별할 수 있다.

(2) 모든 직선에 대해 직선 상 모든 도트 추가

3개 이상의 도트가 포함되면 직선으로 판별하므로, 여러 환경요인으로 인해 계산에서 오차가 발생하여 실제로는 직선 위에 있으나 포함되지 못한 도트가 존재할 수 있으므로 허용오차를 고려하여 포함시켜 준다.

(3) 직선에 속한 도트가 4개 미만이면 제거

한 frame의 도트패턴을 찾기 위해서는 frame line내 frame dots가 최소 4개 이상 있어야 한다.

(4) 도트 간 길이로 필터링

frame dots는 일정한 간격으로 배치되어 있기 때문에 이를 이용하여 다른 직선을 제거할 수 있다. 모든 직선에 대해 직선에 속한 도트들 간 길이를 구하고, 길이의 평균분산(avgVar)을 계산한 후 이 값이 MAXAVGVAR 보다 작은 직선만 남긴다. 필터링 후 남은 직선이 2개 이상인 경우 avgVar가 최소인 직선을 선택한다.

(5) 직선의 방정식 업데이트

선택된 직선은 전체 도트가 아니라 3개 이상의 도트의 좌표로 식을 구한 것이기 때문에 전체 k개의 도트에 대해 모든 경우의 수를 대입하여 파라미터를 수정할 필요가 있다. 이때 파라미터 수정을 위해 사용되는 식은 아래와 같다.

$$\theta = \tan^{-1}\left(\frac{y_1 - y_2}{x_2 - x_1}\right) \quad (1)$$

다음은 최종 선택된 직선을 영상에 나타낸 그림이다.

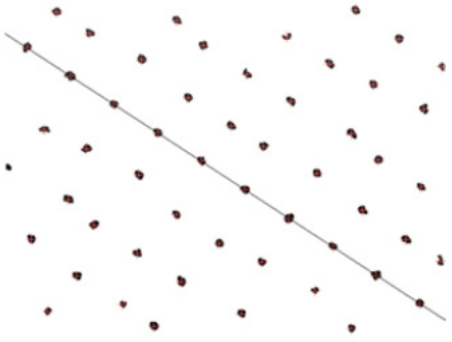

그림 5. 허프 변환 후처리 결과
Fig. 5. Hough transform postprocessing result

3. decode

decode 부분은 가상의 격자점(standP)을 구하는 단계(calStandP), 구한 standP를 실제 도트영역(realP)과 매칭하는 단계(calRealP), standP를 기준으로 realP의 방위를 계산하고 이를 코드화하는 단계(calCode)로 이루어진다.

(1) calStandP

기존의 정보도트(16)은 가상의 standP를 기준으로 1~4

분면 중 하나에 위치한다. 허프 변환을 이용해 찾은 frame dots로부터 standP를 구하기 위해 아래와 같이 standP가 정사각형으로 배치되어 있다는 사실을 이용한다.

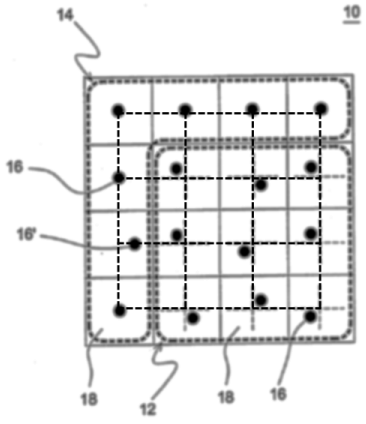

그림 6. 도트패턴의 특징
Fig. 6. Dot pattern feature

그러면 다음과 같이 standP를 구할 수 있다.

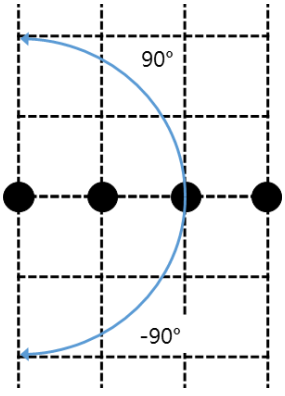

그림 7. standP 계산하는 과정
Fig. 7. Calculating standP process

I번째 frame dot를 기준으로 I+j번째 frame dot를 90도 회전하면 위로 j만큼 떨어진 standP가 계산되고, -90도 회전하면 아래로 j만큼 떨어진 standP가 계산된다. 충분한 데이터를 확보하기 위해 frame line을 포함하여 8줄의 격자점을 구한다.

frame line이 영상의 어느 부분에 위치할지 사전에 알 수 없기 때문에, 자동으로 frame line의 위치를 파악할 필요가 있다. 이에 대한 수식은 아래와 같으며, (x_f,y_f)은 frame line의 처음 도트좌표를 (x_l,y_l)은 마지막 도트좌표를 의미한다.

$$\begin{cases} \text{if}|x_l-x_f|<|y_l-y_f|, & row\times\dfrac{x_f+x_l}{2}\times width \\ otherwise, & row\times\dfrac{y_f+y_l}{2}\times height \end{cases} \quad (2)$$

기준이 되는 frame dots는 정확히 frame line 위에 있는 것이 아니라 허용오차 내에 존재한다. 따라서 frame dots가 frame line 상에서 원래 있어야 할 위치를 계산하고 이를 기준 standP로 삼을 필요가 있다. frame line의 기울기를 고려하여 기준 standP의 좌표를 계산하는 식은 아래와 같다.

$$\begin{cases} \text{if}\theta>1.22 \ or \ \theta<-1.22, & ((\rho-y\cos\theta)\sin\theta,y) \\ otherwise & (x,(\rho-x\sin\theta)\cos\theta) \end{cases} \quad (3)$$

계산된 기준 standP와 원래 frame dots 간에 오차가 발생하게 되는 데 이때 최대 오차를 찾아 이 값을 standP=realP인 0 사분면을 판단하기 위한 기준(maxDiffRho)으로 삼을 수 있다. frame dots의 오차는 아래와 같이 구한다.

$$\begin{aligned} &\max DiffRho \\ &=\max_i|x'\sin\theta+y'\cos\theta-(x\sin\theta+y\cos\theta)| \quad (4) \\ &=\max_i|x'\sin\theta+y'\cos\theta-\rho| \end{aligned}$$

(2) calRealP

calStandP 단계에서 구한 standP를 기준으로 해당 격자점과 가장 가까운 realP를 찾아내야 한다. 각 standP마다 그 점을 지나는 2개의 직선이 존재하는 데, 이 두 직선에서 허용오차 내에 존재하는 도트가 해당 standP에 대한 realP가 된다.

$$x\sin\theta_{stand}+y\cos\theta_{stand}=\rho_{stand} \quad (5)$$
$$x\sin\theta_{inv}+y\cos\theta_{inv}=\rho_{inv} \quad (6)$$

realP를 찾는 알고리즘을 정리하면 다음과 같다. 먼저 모든 realP에 대하여 각 realP와 가장 가까운 frame dots 간 길이(minLen)를 구한다. minLen을 frame dots 간 간

격(len)으로 나누어 frame line과 몇 line 떨어져 있는 지를 계산한 후 search할 row를 추정하고, 가장 가까운 frame dot의 index를 col로 삼는다. standP[startIdx+row, col∓1], standP[startIdx-row, col∓1]을 search하여 조건을 만족하는 인덱스를 찾아 해당 standP를 realP에 binding한다.

(3) calCode

이번 단계에서는 standP를 기준으로 realP의 방위를 계산하고 이를 코드화한다. standP를 지나는 두 직선에 realP의 좌표를 대입하여 나온 값을 rhoStandP와 rhoInv라고 할 때, 아래 그림과 같이 방위를 계산할 수 있다.

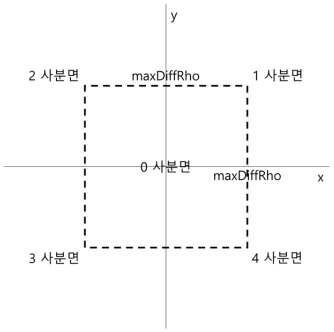

그림 8. realP의 방위
Fig. 8. realP bearing

이때, x는 rhoInv-ρ_{inv}을, y는 rhoStand-ρ_{stand}을 치환한 값이다.

기존 도트패턴의 방위에 따른 코드를 바탕으로 도트패턴을 아래와 같이 변환할 수 있다.

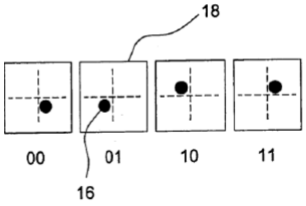

그림 9. 도트패턴의 코드 비트
Fig. 9. Dot pattern code bit

위와 같이 변환된 값을 정해진 순서에 따라 읽으면 도트패턴에 대한 코드를 얻을 수 있다.

Ⅲ. 실험 및 결과

본 논문에서는 도트패턴을 인식하여 적절한 코드를 출력하는 알고리즘을 고안하였다. 이에 대한 성능을 평가하기 위해서 패턴 인식률을 측정했으며, 이는 각 도트패턴을 입력으로 넣었을 때 의도한 데이터가 검출되는지 여부로 결정된다.

총 28개의 도트패턴을 인쇄하였으며, 각 도트패턴을 USB 현미경[8]으로 확대한 후 본 논문에서 고안한 알고리즘을 적용하여 인식결과를 확인하였다. 다음은 패턴 인식률을 평가하기 위한 실험환경에 대한 그림과 각 실험환경요소에 대한 세부사항이다.

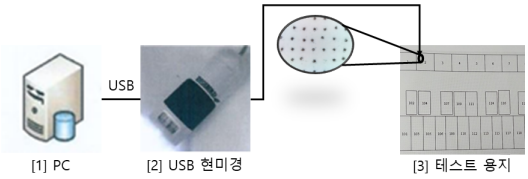

그림 11. 실험환경
Fig. 11. Simulation environment

표 1. 실험환경 세부사항
Table 1. Simulation environment details

번호	요소	세부사항
1	PC	OS: Windows 10 Pro (64bit) CPU: Intel i7-6700 RAM: 8 GB
2	USB 현미경	Model: OPT-230 Magnitude: 140x Resolution: 640x480
3	테스트 용지	28개의 도트패턴이 인쇄된 종이

각 도트패턴에 대해 임의로 5곳을 선정하여 총 140 군데를 실험했으며, 다음은 이에 대한 실험결과를 정리한 표다.

표 2. 실험결과
Table 2. Simulation result

	Correct	Incorrect	Fail
No./Total	119/140	10/140	11/140
	85 %	7.1 %	7.9 %

이때, Correct는 출력 코드와 정답 코드가 일치하는 경우, Incorrect는 출력 코드와 정답 코드가 불일치하는 경우, Fail은 find line 단계에서 직선이 검출되지 않아 코드를 추출하지 못한 경우를 말한다.

Ⅳ. 결 론

본 논문에서는 상용 도트패턴을 인식하여 적절한 코드를 출력하는 알고리즘을 개발하기 위해 여러 영상처리 기술을 적용하였다. 먼저 grayscale, 이진화, 잡음 제거 등을 이용해 배경과 도트를 구분했으며, labeling으로 도트마다 label을 매기고, 허프 변환을 사용해 헤더 부분에 해당하는 직선 frame line과 그 직선에 존재하는 도트들 frame dots를 찾았다. 찾은 정보를 바탕으로 영상 내 존재하는 격자점 standP를 계산하고 구한 standP와 실제 도트 realP를 binding 한 후 realP와 standP 간의 상대위치를 이용해 한 frame 정도 도트의 방위를 계산하고, 해당 방위를 대응하는 코드로 치환했다. 치환한 코드를 적절한 순서에 따라 decode하여 각 도트패턴에 대한 실제 코드열을 얻었다. 해당 분야의 기술은 기존에 공개된 바 없으며 본 논문은 이러한 문제를 해결하려는 최초의 공개적인 시도이다.

고안한 알고리즘의 패턴 인식률을 평가하기 위해 총 28개의 패턴에 대해 5번씩 반복 실험하였으며, 그 결과 85 %의 정확도를 보이는 것을 확인했다.

차후 연구에서는 조명이 균일하지 않거나 영상에 이동이나 회전으로 인한 왜곡이 존재하는 경우에도 동작하도록 개선한다.

References

[1] Ngoc Son Han and Seong-Whan Kim, "Design of pattern recognition scheme for smart pen for IPTV story book game interaction", Korean Society For Computer Game, No. 13, pp. 117-122, June 2008.

[2] Tsai, Yao-Hung, and Li-Ching Chen, "Graphical indicator", U.S. Patent, No. 7549597, 23 Jun 2009.

[3] Mehmet Sezgin, "Survey over image thresholding techniques and quantitative performance evaluation", Journal of Electronic imaging, Vol. 13, No. 1, pp. 146-168, 2004.
DOI: http://dx.doi.org/10.1117/1.1631315.

[4] Eui Chul Lee, "A Method for Improving Vein Recognition Performance by Illumination Normalization", Journal of the Korea Institute of Information and Communication Engineering, Vol. 17, No. 2, pp. 423-430, 2013.
DOI: http://dx.doi..org/10.6109/jkiice.2013.17.2.423.

[5] Tae-Hee Lee, et al, "A Road Extraction Algorithm using Mean-Shift Segmentation and Connected -Component", Journal of Digital Convergence, Vol. 12, No. 1, pp. 359-364, 2014.
DOI: http://dx.doi.org/10.14400/JDPM.2014.12.1.359.

[6] A. Rosenfeld and J. L. Pfaltz, "Sequential Operations in Digital Picture Processing", Journal of the Association for Computer Machinery, Vol. 13, pp. 471-494, 1966.

[7] Hak-Chul Shin, et al, "Performance Improvement of Eye Tracking System using Reinforcement Learning", The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 13, No. 2, pp. 171-179, 2013.
DOI: http://dx.doi.org/10.7236/JIIBC.2013.13.2.171.

[8] http://www.digibird.co.kr/

※ 본 연구는 상명대학교 교내연구비를 지원받아 수행하였음. This research was supported by a Research Grant from Sangmyung University.

저자 소개

박 혜 빈(Hye-bin Park)

• 2012년~현재 : 상명대학교 컴퓨터과학과 재학 중

<관심분야: 유무선 네트워크, 음성인식, 인공지능>

정 진 우(Jinoo Joung)

• 1992 KAIST 전자공학과(학사)
• 1997 NYU School of Engineering (Ph.D in EE)
• 1997~2005 : 삼성종합기술원
• 2005~현재 : 상명대학교 컴퓨터과학과 교수

<관심분야: 유무선 네트워크, SoC design, Embedded system, 인공지능, 음성인식>