Proceedings of the 12th
KACC, October 1997

신경 회로망을 이용한 납땜 검사 FOV의 최적화 알고리즘

Optimal Algorithm of FOV for Solder Joint Inspection Using Neural Network

오 제 휘*, 차 영 엽**

*원광 대학교 대학원 (Tel : (0653) 50-6693 , Email : ggypcha@wonnms.wonkwang.ac.kr)
**원광 대학교 기계공학과 (Tel : (0653) 50-6693 , Email : ggypcha@wonnms.wonkwang.ac.kr)

Abstracts In this paper, a optimal algorithm that can produce the FOV is proposed in terms of using the Kohonen's Self-Organizing Map(KSOM). A FOV, that stands for "Field Of View", means maximum area where a camera could be wholly seen and influences the total time of inspection of vision system. Therefore, we draw algorithm with a KSOM which aims to map an input space of N-dimensions into a one- or two-dimensional lattice of output layer neurons in order to optimize the number and location of FOV, instead of former sequential method. Then, we show demonstration through computer simulation using the real PCB data.

Keywords Vision system, Nueral network, Solder joint inspection, Self-Organizing Map

1. 서론

현재의 전자 제품이 소형화와 다기능화, 그리고 고성능화를 요구함에 따라, 전자 제품 제작에 이용되는 SMD 실장기술은 제작 기술의 급격한 발전에 따라 소형화와 고밀도화 되어 가는 경향을 가지고 있다. 그런데 이러한 부품제작과 실장기술의 급속한 발전에 반하여 검사기술은 기존의 인간에 의한 목시검사에 의존하는 실정이다. 기존의 목시검사는 소형화, 고밀도화 된 SMD 검사시 작업자 눈의 피로 가중, 집중력 저하, 작업자에 따른 판정기준의 상이, 작업자의 숙련도와 컨디션에 따른 품질의 산포 다양, 검사 결과의 빠른 Feedback 불가, 품질의 신뢰도의 하락, 인건비의 상승 등의 문제점을 가지게 되었다. 따라서 이러한 단점을 극복하기 위하여 납땜 외관을 자동으로 검사하는 시스템들이 나타나기 시작했으며, 그 중에서 비젼 시스템(Vision System)을 이용한 검사 장비는 좋은 결과를 나타내고 있다[2].

일반적인 비젼 시스템의 구성은 Host PC부, 영상 획득부, 구동부로 이루어져 있으며, 그 간략한 검사 흐름도는 그림. 1과 같이 나타낼 수 있다. 그림. 1에서 보는 것과 같이 하드웨어(구동부, 영상 획득부)와 함께 가장 중요한 부분은 검사 알고리즘이라고 할 수가 있다. 그래서 검사 알고리즘에 관한 연구는 폭넓게 연구되어져 왔으며[2], 최근에는 신경 회로망을 이용한 검사 알고리즘도 나와 있는 상태이다[4].

하지만 검사 알고리즘 만큼 중요한 FOV에 관한 연구는 상대적으로 적다. FOV란 Field of View의 약자로써 임의의 구동부 좌표에서 영상 획득부 즉 Camera가 볼 수 있는 최대한의 범위를 나타내는 용어이다. 실제 검사 시스템에서는 특정 PCB의 FOV의 크기는 일정하게 정한다. 그러므로, 그 PCB에서 FOV의 중첩을 최소화 하여 FOV의 개수와 거리를 최적화 한다면, 총 부품 검사 시간이 줄어들게 할 수 있을 것이다.

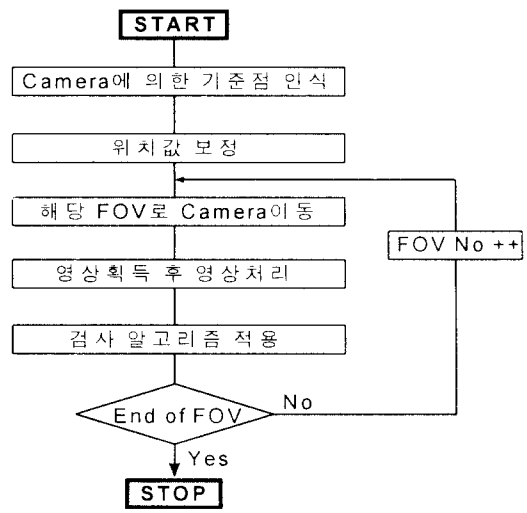

그림. 1 비젼 시스템의 흐름도
Fig. 1 Flowchart of Vision System

본 연구는 신경 회로망 중 코호넨(Kohonen)의 Self-Organizing Map을 이용하여 새로운 FOV 생성 알고리즘을 제안 하겠다. 우선, 기존의 순차적인 FOV생성 알고리즘과 취약점을 설명 하고자 한다. 그리고 새로이 제안되는 알고리즘에 대한 이론적 설명과 시뮬레이션, 그리고 결과를 보이겠다.

2. 순차적인 FOV 생성 알고리즘

기존의 FOV 생성 알고리즘 중에 대표적인 예는 순차적인 방법이다[5]. 알고리즘은 부품 좌표가 최소인 부분을 FOV의 최소 좌표가 되어 FOV의 최대 크기만큼 임의의 FOV를 만든 후, 이 FOV에 포함된 부품을 삭제한다. 그리고 삭제된 부품을 제외한 최소 좌표를 얻어 위의 방법과 같은 작업을 남은 부품이 없을 때까지 한다. 이러한 검사 순서도는 그림. 2와 같다.

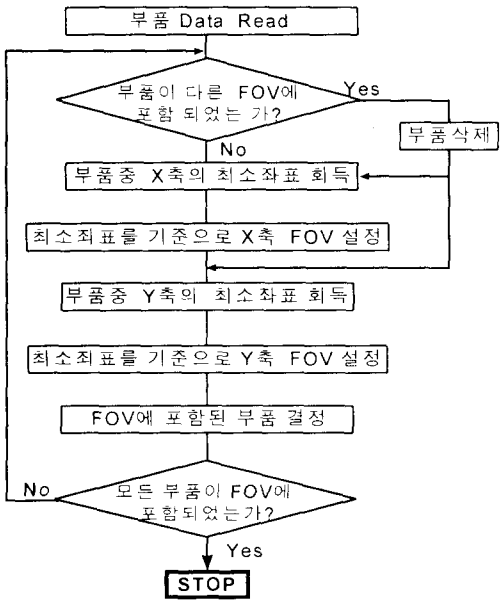

그림. 2 순차적인 FOV 알고리즘
Fig. 2 Algorithm of Sequential FOV

이러한 방법은 비록 원만하게 모든 부품을 FOV에 포함시킬 수는 있겠지만, FOV 수의 과다성과 부품이 많을수록 생성시간의 과중, 프로그래밍의 어려움 등의 단점을 가지고 있다.

특히 FOV수의 과다성은 그림. 3 에서와 같이 각 FOV간의 많은 중첩이 발생하면서 총 검사 시간이 길어지는 결과를 낳게 한다. 그림 .3(a)는 순차적인 FOV를 나타내며, 그림.3(b)의 최적화된 FOV보다 각 FOV간의 중첩에 의한 FOV 수가 많아서 안 좋은 상태이다.

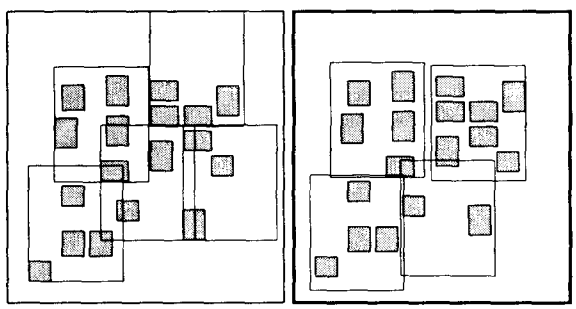

그림. 3 순차적 FOV(a) 와 최적화된 FOV(b) 의 비교
Fig. 3 Comparison of Sequential FOV(a) with Optimal FOV(b)

3. 신경 회로망을 이용한 FOV 생성 알고리즘

네트워크 구조 및 신호 처리 방법에 의해 신경 회로망을 크게 3가지로 나눌 수 가 있다. 첫 번째는 입력 신호들을 임의의 출력 신호로 변형시키는 Feedforward Network이며, 두 번째는 초기의 입력값이 Feedback 시스템에 의하여 점근적으로 최종 상태에 접근하면서 출력값을 찾아내는 Feedback Network이다. 그리고, 마지막으로 측면의 영향력에 의해 스스로 활동을 완성 시키거나 서로 다른 신호들의 분류가 가능한 Self-Organizing Map이 있다[1].

본 연구는 거리 단위로 표시되는 입력값을 분류하는 작업이 필요하므로 Self-Organizing Map을 사용한다. 그중 헬싱키 공과대학의 튜보 코호넨(Tuevo Kohonen)에 의해 개발된 Kohonen's Self-Organizing Map(이하 KSOM)을 이용하여 최적화 문제를 풀었다.

3.1 Kohenen's Self - Organizing Map.

Self-Organizing Map에서는 기대된 출력이 없다. 대신에 신경 회로망은 자기 조직화 특성에 의해 어떤 관계를 추론할 수 있으며 더 많은 입력이 인가 될수록 네트워크는 그 학습을 개선하고 변화된 입력들에 적응 할 수 있다. 이러한 구조의 한 이점으로는 변화하는 상태와 입력에 대하여 대처할 수 있다는 것이다. 이러한 시스템은 입력을 다른 카테고리(Category)로 분류할 때와 음성 인식, 로봇 모터 제어 등에 사용된다.

KSOM의 목적은 N-차원의 입력 공간을 의미 있는 지형학적인 순서로 1차원 또는 2차원의 출력 공간(출력 뉴런)에 매핑할 수 있게 하는 것이다. 이러한 목적을 성취하기 위해 경쟁학습(competitive learning) 에 의한 승자독점(winner-take-all)원리와 측면 제어(Lateral inhibition)가 이용된다[1,3].

입력 벡터를

$$X = [x_1, x_2, \ldots x_p]^T \tag{1}$$

로 하고, 가중치 벡터를

$$W_j = [\omega_{j1}, \omega_{j2}, \ldots, \omega_{jp}]^T \quad j = 1, 2, \ldots, N \tag{2}$$

로 표시한다면, 출력 층의 승자 뉴런의 결정은 입력 값 X 와 가장 비슷한 가중치 W_j 를 갖는 출력 뉴런을 선택하는 것과 같다. 이러한 출력 뉴런을 선택하는 방법은 두 가지가 있다.

첫째는

$$I_{j\max} = \sum_{i=1}^{p} \omega_{ji}\, x_i \tag{3}$$

식(3)과 같은 I_j 를 선택하는 것이고, 두 번째는 입력값에서 부터 최소의 Euclidean norm에 위치한 가중치를 갖는 출력 뉴런을 선택하는 것이다. $i(X)$ 가 승자 뉴런이라 한다면 이 식은 다음과 같다.

$$i(X) = k \quad where\ \|W_k - X\| < \|W_i - X\| \tag{4}$$

위와 같이 뉴런을 선택하는 것이 경쟁학습에 의한 승자독점 원리이다. 보통 가중치 벡터와 입력 벡터는 정규화(Normalization) -가중치 벡터와 입력 벡터의 크기가 1로 설정됨 - 시키는데 그

이유는 트레이닝 규칙이 입력 벡터로 부터 가중치 벡터를 뺀 값을 사용하기 때문이다. 그리고, 측면 제어의 대표적인 방법은 이웃관계 함수(Neighborhood function) $\Lambda_{i(x)}(n)$ 을 이용하는 것으로써, 연결 강도는 거리에 반비례 한다. 여기서, n 은 시간을 나타내며, 트레이닝 법칙은 다음과 같다.

$$W_j(n+1)=\begin{cases} W_j(n)+\eta(n)[X-W_j(n)] & j\in\Lambda_{i(x)}(n) \\ W_j(n) & otherwise \end{cases} \quad (5)$$

여기서 $\eta(n)$ 은 n 시간 일때의 Learning rate이다.

3.2 FOV 생성 알고리즘

FOV 생성 알고리즘을 설명하기 위하여, FOV의 변수를 설명 후 KSOM을 이용한 실제 알고리즘을 설명 하겠다.

3.2.1 FOV의 변수

FOV의 이해를 위하여 그림. 4는 FOV의 변수들을 보여주고 있다. 부품의 변수는 ($C_{x,\min}$ $C_{y,\min}$),($C_{x,\max}$ $C_{y,\max}$)로서 각각 최소점의 좌표와 최대점의 좌표를 나타낸다.

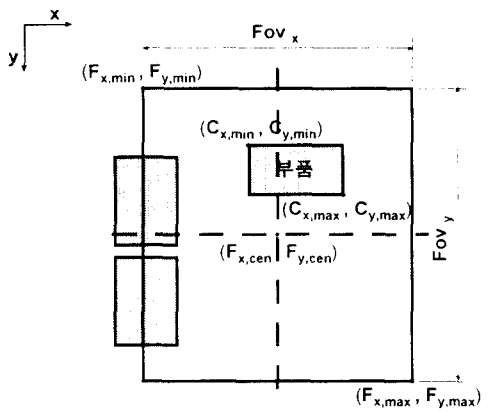

그림. 4 FOV의 변수
Fig. 4 Parameter of FOV

FOV의 변수는($F_{x,\min}$ $F_{y,\min}$), ($F_{x,\max}$ $F_{y,\max}$), ($F_{x,cen}$ $F_{y,cen}$). Fov_x , Fov_y 로서, 각각 FOV의 최소점 좌표, 최대점 좌표, 중심점 좌표, X방향의 길이, Y방향의 길이들을 나타내며 이 변수들의 관계는 다음과 같다.

$$\begin{aligned} F_{x,\min} &= F_{x,cen}-\frac{Fov_x}{2} \\ F_{y,\min} &= F_{y,cen}-\frac{Fov_y}{2} \\ F_{x,\max} &= F_{x,cen}+\frac{Fov_x}{2} \\ F_{x,\max} &= F_{x,cen}+\frac{Fov_y}{2} \end{aligned} \quad (6)$$

3.2.1 알고리즘

위의 FOV의 변수에서 보듯이 FOV의 크기는 이미 정해져 있다. 그러나 고유의 KSOM에는 거리의 제약을 줄 수 없으며, 출력 뉴런의 수가 변할 수 없는 커다란 단점을 갖는다. 그래서 본 연구에서는 고유의 KSOM을 변형하고, 새로운 항들을 만들어 사용하였다.

실제 이용하는 입력 벡터를 부품의 변수 $C_{x,\min}$, $C_{y,\min}$, $C_{x,\max}$, $C_{y,\max}$ 4개를 사용 하였고, 가중치 벡터(출력 뉴런층과 동일)로는 FOV의 중심 좌표($F_{x,cen}$, $F_{y,cen}$)를 사용하여 유동적인 뉴런의 수를 갖는 2차원의 Map으로 하였다. 그리고, KSOM의 수는 2개로 하였다. 즉, 입력 벡터를 $C_{x,\min}$, $C_{y,\min}$ 으로 하고 가중치 벡터를 $F_{x,cen}$, $F_{y,cen}$ 로 하는 SOM1과 입력 벡터를 $C_{x,\max}$, $C_{y,\max}$ 으로 하고 가중치 벡터를 $F_{x,cen}$, $F_{y,cen}$ 로 하는 SOM2이다. 이렇게 하는 이유는 그림. 4에서 보듯이 부품에 따라 크기가 다르므로 각 부품의 최소 좌표와 최대 좌표가 따로 계산 되어져야 하기 때문이다.

SOM1과 SOM2의 승자 뉴런은 각각 식(4)에 의하여 얻어지며 이를 각각 $i(X)_1$, $i(X)_2$ 로 한다면, 트레이닝 법칙은 이웃관계 함수 $\Lambda_{i(x)}(n)$ 을 제거하고 새로이 SOM1과 SOM2 관계 항으로 변형하여 다음 식과 같이 사용한다.

$$W_j(n+1)=\begin{cases} W_j(n)+\eta_1(n)[X_2-W_j(n)] & i(X)_1=i(X)_2 \\ W_j(n)+\eta_2(n)[X_1-W_j(n)] & otherwise \end{cases} \quad (6)$$

하첨자 1, 2는 SOM1, SOM2를 나타내는 것이고, Learning rate $\eta_1(n)$, $\eta_2(n)$ 은 $\eta_1(n)<\eta_2(n)$ 로 설정 해야하며, 시간이 경과할수록 그 값들이 줄어 들어야 한다. 위의 방식만 사용하면, 고정된 FOV 크기에 의해 FOV간의 필요없는 중첩과 자신의 FOV가 없는 부품이 생길 수가 있으므로 FOV의 이동과 삭제, 그리고 생성을 가능하게 하는 항을 새로이 만들었다.

i) 이동 : FOV(즉, 출력 뉴런 또는 승자 뉴런)사이의 거리 유지에 대한 어려움에 의해 생겨나는 자신의 FOV가 없는 부품 l 이 가장 인접한 임의의 FOV m 에 포함되기 위한 FOV의 이동식은

$$W_m(n+1)=\begin{cases} W_m(n)+\eta_{i(m)}[X_{1,l}-W_m(n)] & \delta\geq Gap_m \\ W_m(n) & otherwise \end{cases} \quad (7)$$

이고, 여기서 δ 는 이동 한계 변수이며, Gap_m 은 최소 부품 거리항으로 부품 l 과 가장 가까운 임의의 FOV m 의 거리를 나타내며, 식은 다음과 같다.

$$Gap_m=min\|W_m(n)-X_{1,l}\| \quad (8)$$

ii) 생성 : 자신의 FOV가 없는 부품이며, 이동항을 적용할 수 있는 FOV가 없는 부품 l 이 있는 경우, 이 부품을 포함시키는 FOV를 생성 해야하며, 식은 식(9)이다.

$$W_{new}(n+1)=X_{1,l}+\alpha \qquad where\ \delta<Gap_m \quad (9)$$

여기서, α 는 FOV 생성 변수로써 부품 l 을 포함시킬 FOV의 위치를 설정 해준다.

iii) 삭제 : FOV간의 중첩을 최소화하기 위해, 임의의 FOV r 을 삭제하기 위한 식은

$$W_r(n+1) = \begin{cases} W_r(n) & \phi \geq Dist_r \\ \infty & otherwise \end{cases} \quad (10)$$

이고, 여기서 ϕ 는 삭제 경계 변수이며, $Dist_r$ 은 FOV의 최소 주변 거리항으로 임의의 FOV r 이 주변 FOV와의 최소 거리를 나타내며, 식은 다음과 같다.

$$Dist_r = min\{\|W_r(n) - W_i(n)\|\} \quad (11)$$

위의 세가지 항들을 추가하여 모든 부품이 FOV에 속해질 때까지 알고리즘을 실행시키며, 이러한 과정을 정리 한다면 다음과 같다.

Step 0 : 입력 값 $C_{x,\min}$, $C_{y,\min}$, $C_{x,\max}$, $C_{y,\max}$ 을 읽고, 초기 FOV 개수를 설정한다. 그리고, 각각의 변수인 $\eta_1(n)$, $\eta_2(n)$, $\eta_3(n)$, δ, α, ϕ 를 정한다.

Step 1 : 초기 FOV(뉴런)의 배열을 위한 최소 부품 위치를 찾은 후, FOV를 개수만큼 2차원의 그물형으로 배열한다.

Step 2 : $C_{x,\min}$, $C_{y,\min}$ 를 입력 값으로 하는 SOM1과 $C_{x,\max}$, $C_{y,\max}$ 를 입력 값으로 하는 SOM2에서 승자뉴런 $i(X)_1$, $i(X)_2$ 을 찾기 위해 식(4)을 사용한다.

Step 3 : 승자뉴런 $i(X)_1$, $i(X)_2$ 을 이용하여 식(5)의 트레이닝 법칙에 적용한다.

Step 4 : 아래의 Step 4.1과 4.2 그리고 4.3을 적당한 시간 간격을 주고 실행한다. 실제로는 Step 4.1이 Step 4.2보다, Step 4.2는 Step 4.3보다 실행 빈도가 많아야 한다.

Step 4.1 : Learning rate $\eta_1(n)$, $\eta_2(n)$를 감소 시킨다.

Step 4.2 : 이동 항(식(7))과 생성 항(식(9))을 실행 시킨다.

Step 4.3 : 삭제 항(식(10))을 실행 시킨다.

Step 5 : 모든 부품이 자신의 FOV를 갖는 경우 정지하고, 그렇지 않은 경우 Step 2로 간다.

Step 6 : 부품이 존재하는 FOV만 남기고 나머지 FOV는 삭제한다.

4. 시뮬레이션 결과

시뮬레이션을 위하여 시편으로는 부품이 200개인 실제 PCB를 사용 하였다. PCB의 크기는 124×120(mm)이고, FOV의 x방향 길이는 10.6 mm, FOV의 y방향 길이는 14.0 mm 이다. 입력 값은 랜덤하게 하여 실행 했으며, 각 변수의 값은 $\eta_1 = 0.4$, $\eta_2 = 0.9$ 로 하여 300회마다 0.8씩 감소시켰다. $\eta_3 = 0.8$ 로 하였고, $\delta = 5(mm)$, $\alpha = Fov/2$, $\phi = Fov/2$ 또는 $Fov/3$ 으로 하였다. 그리고 이동 항과 생성 항은 900회마다, 삭제항은 1200회마다 실행하였다. 위와 같이 설정했을 때, 시뮬레이션 결과는 그림. 5와 같다.

6. 결론

본 연구는 기존의 방법과는 다른 신경 회로망을 이용한 알고리즘을 사용하여 FOV의 개수와 거리를 최적화 하였다. 비록 많은 입력에도 안정하지만, 기존의 순차적인 방법보다 더욱 나은 결과를 얻기 위해 보조항들의 개선 및 추가가 필요하다고 사료된다.

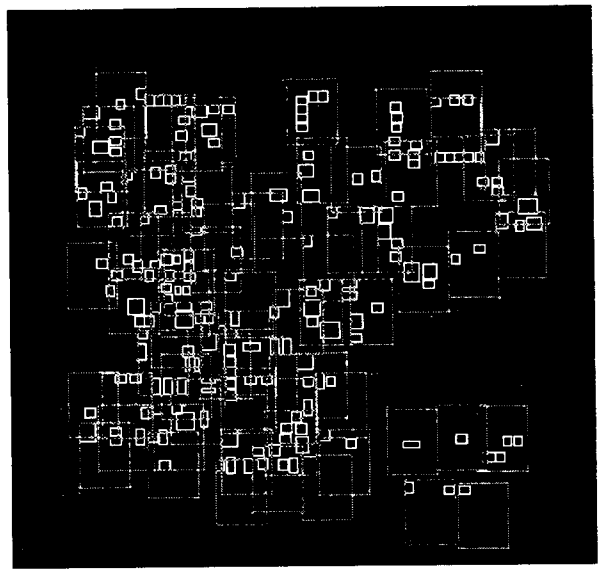

그림. 5 시뮬레이션 결과

Fig. 5 Simulation result

참고문헌

[1] T. Kohonen, "The Self-Organizing Map", Proc. of the IEEE, vol. 78, no. 9, pp.1464-1480, 1990

[2] T. S. Newman and A. K. Jain, "A Survey of Automated Visual Inspection", computer vision and image understanding, vol 61, no. 2, pp. 231-261, 1995

[3] A. S. Pandya and R. B. Macy, *"Pattern Recognition with Neural Networks in C++"*, CRS Press, 1995

[4] Y. K. Ryu and H. S. Cho, "A Neural Network Approach to Extended Gaussian Image based Solder Joint Inspection", mechatronics, vol. 7, no. 2, pp. 159-184, 1997

[5] 김철우, *"비전을 이용한 PCB 납땜 검사장치의 개발"*, 한국과학기술원, 1996