영상을 적용한 인공지능을 이용한 Robot Arm Placing 기술 개발

백영진, 김원하
경희대학교 전자공학과
ch03058@naver.com, wonha@khu.ac.kr

Development of Robot Arm Placing technology based on Artificial Intelligence using image data

Young-Jin Baek, Wonha Kim
Dept. of Electronic Engineering
Khung-Hee University

요약

최근 딥 러닝을 이용해 기계로 인간을 대체하는 스마트 팩토리에 대한 연구 및 개발이 활발히 진행되고 있다. 그러나 FPCB를 Placing하는 방법에 기계를 도입하는 과정은 발전이 더딘 상태이다. 현재 로봇 팔을 이용해 Placing하는 방법은 사람이 직접 로봇 팔을 튜닝해 사용하고 있다.

이에 본 논문은 딥 러닝을 이용한 영상처리 기법을 활용해 FPCB를 사람의 개입 없이 트레이에 삽입하는 기법을 개발하였다. 이를 위해 여러 알고리즘을 비교한 후 각각의 장단점을 고려해 적합한 알고리즘을 제시하였다. 본 논문에서 제시하는 기법은 FPCB에 아무 행동을 가하지 않으며, 힘 센서, 깊이 센서 등 기타 센서들의 도움 없이 RGB 센서(카메라)를 통해 획득한 이미지만을 이용해 자동화가 가능하다. 또한, 개발 단계에서 실제 기계를 이용해 이미지 촬영, 이동 등을 진행했기 때문에 조명, 로봇 팔 위치 등 알고리즘 외 조건들에 영향을 받지 않고 실제 사용이 가능하다.

1. 서론

스마트 팩토리에 대한 연구 및 개발은 활발히 진행 중이다. 그 중, 많은 알고리즘이 영상 신호처리와 딥 러닝을 결합한 방법이다. 특히 기존의 전통적인 방법으로 접근할 경우 시간이 너무 오래 걸려 사용되지 못했던 알고리즘들이 딥 러닝을 만나 빛을 보고 있다. 그 중, 광학 흐름(Optical Flow)는 이미지의 모든 픽셀의 모션 벡터를 구할 수 있는 장점이 있지만 모든 픽셀에 대한 계산을 진행하기 때문에 시간이 매우 오래 걸렸다. 그러나 딥 러닝과 결합하며 시간을 비약적으로 단축할 수 있어 조명받고 있다.

현재 로봇 팔을 이용해 Flexible Printed Circuits Board(FPCB)를 트레이에 삽입하는 기술 연구는 진행 중이나 사용 가능 범위가 제한적이거나 사용되는 리소스가 매우 커 쉽게 접근할 수 없다. 따라서 진행된 연구들이 있음에도 대부분 사람이 직접 FPCB Placing 작업을 진행 중이다. 이를 보완하기 위해 본 논문의 목적은 최소한의 리소스로 최대의 정확도를 얻는 것을 목표로 한다.

단순히 한 종류의 FPCB와 트레이에 대한 알고리즘을 만들면 문제는 매우 간단해진다. 트레이의 좌표와 FPCB의 특정한 좌표를 이용해 모션 벡터를 쉽게 추출할 수 있다. 하지만 이러한 방법은 FPCB의 모양이나 트레이의 색상이 변하면 사용할 수 없어 한계가 뚜렷하다. 현존하는 FPCB는 약 10만 가지의 다양한 FPCB가 있으며, 각각의 경우에 대해 일일이 알고리즘을 새우는 것은 불가능에 가깝다. 따라서, 로봇에 부착된 RGB 센서(카메라)에서 획득한 이미지에서 FPCB와 트레이를 인식해 삽입하는 알고리즘 개발이 필요하다.

이에 본 논문은 convolutional neural network(CNN)을 이용해 광학 흐름(Optical Flow)을 계산하는 FlowNet2.0와 극좌표 변환(Polar Coordinate)을 이용해 FPCB의 회전각을 획득하는 알고리즘 개발했다. 회전각을 안다면 다음은 단순한 이동 문제이기에 회전각이 Placing에 핵심 요소이다. 이는 RGB 센서 외 센서를 사용하지 않았으며 Real world에서 개발해 다양한 FPCB에 대해 용이하게 적용할 수 있다.

2. FlowNet2.0

Convolutional Neural Network(CNN)은 영상 신호처리의 많은 알고리즘들을 실제 필드에서 사용할 수 있도록 만들어주었다. 덕분에 컴퓨터 비전은 큰 발전을 이룰 수 있었다. 특히 소요 시간이 너무 커 가진 장점이 많음에도 사용될 수 없었던 광학 흐름의 경우 많은 연구가 이루어졌다. FlowNet은 CNN을 이용해 광학 흐름의 속도 및 정확도를 기존의 전통적인 알고리즘보다 비약적으로 증가시킨 기법이다[1]. FlowNet은 정형적인 CNN 구조를 이용한 FlowNetSimple과 두 개의 입력 이미지를 따로 convolution layer를 통과시켜 correlation layer에서 상관관계를 찾는 두 가지 기법을 제시했다. 두 모델 모두 속도 면에서 큰 향상을 보여 실생활의 문제에 적용할 가능성을 보였으나 정확도 측면에서 아직은 부족한 모습을 나타냈다.

FlowNet의 두 모델을 적절히 조합해 단점을 보완한 FlowNet2.0은 전통적인 알고리즘들과 비교해 속도와 정확도 모두 뛰어나 Real-time 문제에 적용할 수 있음을 보여주었다[2]. 이러한 FlowNet2.0을 이용해

FPCB의 모션 벡터를 추출할 수 있었다. 다만 FlowNet2.0을 이용해 모션 벡터를 추출하기 위해 정답 이미지와 불필요한 부분을 제거해 노이즈를 최대한 줄이는 사전 작업이 필요했다. 또한, GPU 사양에 맞게 이미지를 조절해 원활하게 작동할 수 있게 했다.

2.1. 로봇 팔 Tool 제거

광학 흐름을 이용하기 위해서는 구하고자 하는 FPCB를 제외한 나머지의 영향이 최소가 될수록 좋다. 사용하는 로봇 팔은 항상 동일하기 때문에 threshold등을 이용해 tool을 제거하기보다 정확한 좌표들을 이용해 제거했다. 이는 조명, 노이즈 등에 의해 생길 수 있는 변화를 방지한다. Tool 제거 이미지의 경우 최초 1회만 생성하고 이후에는 만들 필요가 없어 정확도를 높이는 방향으로 선택했다.

Figure 1. Robot Arm에서 Tool을 제거한 이미지

2.2. 정답 이미지 획득

정답 이미지(reference image)는 FPCB가 트레이에 정확하게 삽입될 수 있는 위치에 있는 이미지이다. 정답 이미지를 구하기 위해서는 두 장의 이미지가 필요하다. 하나는 트레이에 정확히 삽입할 수 있는 위치에 FPCB가 있는 이미지이고, 다른 하나는 FPCB를 잡고 있지 않은 상태의 이미지이다. 이 두 이미지의 차이를 계산해 값이 큰 경우 FPCB로 인식한다. FPCB를 고정한 상태에서 배경을 회전해 차이를 계산하는 방법도 있지만, 이 경우 눈으로 보는 것보다 조명과 각 위치가 찍히는 각도에 의해 차이가 크게 발생해 FPCB의 이미지만을 정확히 출력하기가 어렵다.

Figure 2. FPCB가 부착된 이미지와 그렇지 않은 이미지

Fig.2의 두 이미지의 차이를 이용하면 Fig.3의 이미지를 얻을 수 있다. Fig.3은 앞으로 사용할 정답 이미지를 배경이 있는 이미지, 배경을 차이를 이용해 제거한 후 threshold를 걸어 노이즈까지 제거해 FPCB만 존재하는 이미지이다.

Figure 3. FPCB reference image

2.3. Robot Arm 회전축 좌표 획득

FPCB의 모션 벡터를 구해 translation과 rotation으로 성분을 분해해 적용할 수 있다. Translation의 경우 기준점이 필요 없지만, rotation은 어디를 중심으로 회전시키는지에 따라 이미지가 크게 달라진다. 따라서 robot arm에서 회전축의 좌표가 필요하다. 이를 위해 FPCB를 잡지 않은 상태에서 회전시키며 데이터를 획득한다. 이때 회전각도는 알고 있다. 각각의 이미지에 대해 회전축이 위치한다고 추측되는 영역의 좌표들에 대해 회전각만큼 보상한 후 SSIM 알고리즘을 이용해 가장 점수가 높은 좌표를 획득한다. SSIM 알고리즘은 이미지의 밝기, 콘트라스트, 그리고 구조를 비교해 이미지의 유사도 점수를 계산한다.[3]

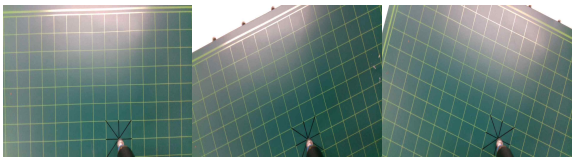

Figure 4. SSIM 알고리즘에 사용된 이미지(0 , -19 , 21)

Fig.5는 0 의 reference 이미지를 기준으로 회전각을 보상한 후 나머지 이미지들에 대해 SSIM 알고리즘을 적용한 결과이다.

각도( )	좌표	SSIM 점수
-19	(783,635)	0.912389
-6	(782,635)	0.945469
-3	(782,636)	0.947600
3	(784,639)	0.947236
6	(782,634)	0.946610
11	(782,635)	0.943099
21	(785,635)	0.925145

Figure 5. SSIM 알고리즘 적용 결과

Fig.5의 결과를 토대로 (782,635)의 회전축 좌표를 획득했다. 이미지마다 조명에 의해 2, 3 픽셀 차이가 발생하지만, 이 정도의 오차는 실제 회전시킬 때 큰 영향을 주지 않아 무시할 수 있다.

2.4. 이미지 Crop 및 FlowNet 실행

광학 흐름(Optical Flow)는 이미지의 모든 픽셀에 대해 모션 벡터를 계산하므로 이미지 사이즈가 커질수록 필요한 리소스가 매우 크다. 따라서 RGB센서가 제공하는 1280*720 해상도의 이미지를 그대로 사용할 경우 실제 사용하기 힘들 정도로 긴 시간이 필요해 이미지를 적절한 사이즈로 crop하는 과정을 거친다. 또한, RGB 레벨의 이미지를 그대로 사용할 경우 그레이 스케일에 비해 3배의 계산량이 요구되므로 이미지를 그레이 스케일로 변환했다. 그 후 FlowNet2.0을 이용해 Vector Map을 획득할 수 있다. Fig.6은 Optical Flow Map을 쉽게 보기 위해 HSV로 변환한 이미지이다.

Figure 6. Reference Image, Rotation Image, OF Map(HSV)

그러나 이렇게 벡터 맵을 작성하면 배경에 대한 광학 흐름까지 전부 출력되기 때문에 FPCB만의 모션 벡터 추출이 어렵다. 2.2에서 획득한

배경이 제거된 Reference 사용한다면 FPCB의 모션 벡터를 얻을 수 있을 것이다. 하지만 Rotation 이미지의 경우 FPCB 위치의 트레이 모양이 고정값이 아니기 때문에 같은 알고리즘으로 배경을 제거하기 어렵다. 따라서 Fig.6과 같이 회전된 FPCB가 Reference FPCB로 이동한 모션 벡터를 바로 구하기는 어렵다.

배경 간섭 문제를 해결하기 위해 2.2의 배경 제거된 이미지를 사용할 알고리즘을 개발했다. 우리가 목표로 하는 회전된 이미지에서 정답 이미지로의 모션 벡터를 바로 구하기보다 정답 이미지가 회전된 이미지로 이동하는 모션 벡터를 구한 후 역으로 계산한다. 이를 이용할 경우 벡터 맵의 값은 달라지나, 형태는 고정되어 있어 배경 제거 이미지에서 0인 영역들의 모션 벡터들을 지워 해결할 수 있다. Fig.7은 배경 제거된 벡터 맵이다.

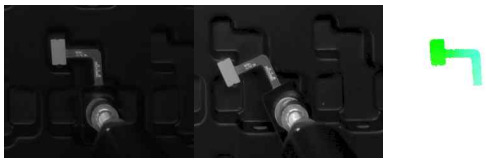

Figure 7. Reference Image, Rotation Image, OF Map(HSV)

2.5. 회전각 획득

Fig.7의 벡터 맵을 구한 후 FPCB의 회전각을 추출하는 단계를 거친다. 회전과 이동을 분리해 생각하면 각각 FPCB 전체적으로 통일된 값을 가진다. FPCB의 이동은 트레이에서 비어 있는 공간을 스캐닝해 좌표를 계산하기 때문에 이번 단계에서는 회전각을 중심으로 구한다.

FPCB의 회전각을 구하기 위해 벡터의 내적을 이용한다. 각 픽셀의 모션 벡터 (u, v)를 이용해 이동 전 좌표와 이동 후 좌표를 구할 수 있다. 이 두 좌표와 이동 전, 후의 공통 좌표인 회전축 좌표를 이용해 두 개의 벡터를 만든다.

$$\theta = \cos^{-1}\left(\frac{v1 \cdot v2}{\|v1\| \times \|v2\|}\right)$$

두 개의 벡터가 있을 때 위 식과 같이 사이 각도를 구할 수 있다. 이를 이용해 각각의 픽셀에 대한 회전각을 구할 수 있다. 이때 어떤 픽셀의 회전각을 이용할지 결정해야 한다. 모션 벡터의 크기가 가장 큰 픽셀을 이용하면 그 값이 노이즈일 확률이 높다. 비슷하게 크기가 작은 픽셀 역시 노이즈일 확률이 높다. 평균을 이용할 경우 회전축과 가까운 영역이 변화가 적게 보여 원하는 각도보다 적게 측정된다. 따라서 전체를 대표할 수 있도록 가장 많은 픽셀이 속한 회전각을 선택하는 방법을 사용했다. 가장 많은 픽셀이 속한 회전각을 구하기 위해 Fig.7의 벡터 맵에서 값이 있는 부분에 대해 히스토그램을 작성해 최댓값을 가진 각도를 얻는다.

Figure 8. 결과 각도를 이용해 Reference 이미지를 회전시킨 이미지와 결과에 -1을 곱해 Rotation 이미지를 회전시킨 이미지

2.6. 오차 보정

광학 흐름 알고리즘은 큰 변화(Large displacement)에 전반적으로 취약하다. FlowNet2.0 역시 이러한 단점을 가지고 있어 회전각이 큰 경우 정확한 각도를 한 번에 얻어내기 쉽지 않다. 하지만 변환 방향이 정답과 일치하고 변화량이 유사해 한 번에 정답과 매우 가까워질 수 있다. 하지만 여전히 오차가 발생하는데 이를 보완하기 위해 두 가지 방법이 있다.

첫 번째는 FlowNet2.0 실행 시 정답 이미지로 사용되는 이미지의 교체이다. 배경이 없는 이미지는 FPCB 그 자체의 모션 벡터 추출에 뛰어나지만, 배경의 경우 없던 것이 생겨난 것과 같아 결과 예측이 어렵다. 이런 문제는 큰 변화에 약한 광학 흐름 알고리즘의 단점과 맞물리게 된다. FPCB의 변화가 큰 상태에서 배경의 변화마저 큰 경우 두 이미지를 연속된 동작이 아닌 다른 상태의 이미지들로 판단할 수 있다. 이 경우 모션 벡터 측정이 불가능하며 노이즈만 검출된다. 이를 방지하기 위해 벡터 맵 작성 과정에서는 배경이 있는 이미지를 사용해 정상적으로 벡터 맵을 생성해야 한다. 성공적으로 벡터 맵이 작성되면 배경이 제거된 이미지를 이용해 값이 0인 부분의 모션 벡터를 제거한다. 이를 통해 두 이미지를 아예 다른 이미지로 판단할 가능성을 낮춘다.

두 번째는 반복적인 광학 흐름 계산이다. FPCB의 회전각에 따라 적게는 1, °2 , 많게는 °9 정도 오차를 보인다. 특히 회전각이 30 가 넘어가면 오차가 크게 나타난다. 이를 보완하기 위해 1차 결과에서 얻은 결과를 이용해 이미지를 회전시킨 후, 2차 결과와 정답 이미지를 이용해 같은 작업을 반복하면 정답에 매우 근사한 결과를 얻을 수 있다. 이 과정을 최종적으로 n차 결과와 정답 이미지의 회전각이 °1 이하면 정답에 도달했다고 생각할 수 있다. Fig.9는 n차 결과를 이용해 회전시킨 이미지와 정답 이미지를 비교한 이미지다.

Figure 9. 회전각만큼 Reference 이미지를 회전시켜 Rotation 이미지와 비교한 결과(1차, 2차)

두 방법을 이용해 정확도를 1도 내외로 줄일 수 있으나, 그만큼 시간이 오래 걸리고 소요 시간을 정의할 수 없다는 문제가 있다.

3. Polar Coordinate

광학 흐름을 이용해 FPCB의 모션 벡터를 구할 수 있으나 회전각이 크면 두 번 이상 알고리즘을 적용해야 해서 시간이 오래 걸리고 회전각에 따라 소요 시간이 달라 시스템 설계에 적합하지 않다. 또한, 이미지의 많은 부분을 차지하는 배경 영역을 사용하지 않아 리소스 낭비가 크다. 이를 보완하기 위해 회전에 특화된 극좌표로 이미지를 변환시킬 경우 변환 이미지의 평행 이동 거리로 회전각을 구할 수 있다. 즉 2차원 이미지의 회전각을 1-D correlation을 이용해 구할 수 있다.

극좌표 변환은 방법으로 L1 Norm과 L2 Norm이 있다. L1 Norm은 원소들의 차이의 절댓값의 합을 이용하며 L2 Norm은 두 벡터의 직선거리를 이용한다. Fig.10은 L1 Norm과 L2 Norm의 차이를 보여준다.

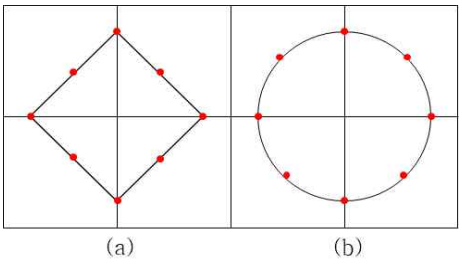

Figure 10. (a)L1 Norm 변환 시 같은 거리에 있는 점들, (b)L2 Norm 변환 시 같은 거리에 있는 점들

L1 Norm을 이용해 극좌표 변환 시 물체가 회전한 정도에 따라 같은 물체를 변환시킴에도 왜곡이 일어난다. 반면에 L2 Norm을 이용해 변환하면 같은 거리에 있는 경우 그 형태가 유지되어 회전된 물체도 형태를 유지할 수 있다. Fig.11은 FPCB를 L1 Norm, L2 Norm을 이용해 극좌표 변환한 결과이다.

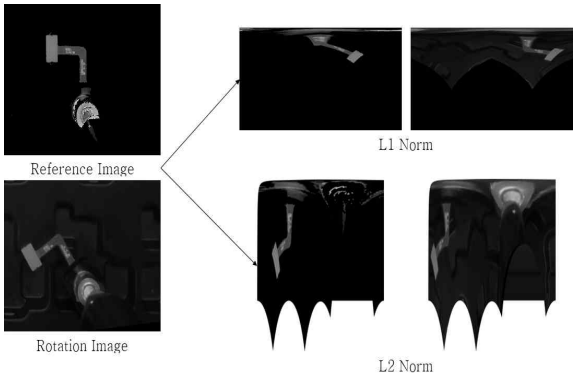

Figure 11. L1 Norm, L2 Norm 변환 시 결과 이미지

L2 Norm을 이용해 각도를 획득하면 FlowNet2.0으로 회전각을 구할 때보다 시간을 단축할 수 있다. FlowNet2.0은 반복된 실행으로 2n초가 걸리는 반면, 극좌표 변환 알고리즘은 같은 이미지에 대해 0.13초의 짧은 실행 시간을 나타냈다. 그러나 정확도는 FlowNet2.0은 정답 이미지와 1도 미만의 오차가 나타날 때까지 진행해 오차가 거의 발생하지 않지만, 극좌표 변환은 약 2, °3 의 오차가 발생했다. Fig.12는 FlowNet2.0과 같은 이미지에 대한 극좌표 변환 결과다.

Figure 12. Reference, Rotation 이미지 극좌표 변환 결과 및 두 이미지의 차를 출력한 결과

4. FPCB catching이 성공적이지 못한 케이스

여태까지 로봇 팔이 FPCB를 정답 이미지와 같은 위치를 잡는 가정하에 알고리즘을 검증했다. 그러나 항상 FPCB를 잘 잡는다는 보장이 없어 FPCB를 잘못 잡을 때를 검증했다. 그 결과 FlowNet2.0은 두 이미지를 연속적인 이미지로 연결하지 못해 노이즈가 발생했고, 극좌표 변환은 1-D correlation 과정에서 높은 상관관계를 찾지 못했다. Fig.13은 정답 이미지, 정답 이미지를 극좌표 변환에 적합하게 crop한 이미지, 그리고 catching이 성공적이지 못한 이미지다.

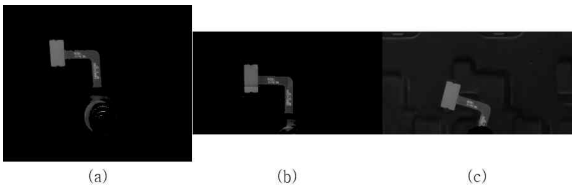

Figure 13. (a)정답 이미지, (b)극좌표 변환에 적합하게 180 로 crop한 이미지, (c)catching이 성공적이지 못한 케이스

극좌표 변환 알고리즘을 사용하기 위해 Fig.13 (c)의 정답 이미지를 생성해야 한다. Fig.13 (b)를 내려야 하는데 극좌표 변환의 경우 correlation을 이용하기 때문에 정확한 높이가 요구되지 않는다. 이 경우 광학 흐름을 이용하면 정답에 근사한 수치를 얻을 수 있다. 이를 이용해 FPCB 모션 벡터의 y축 성분만큼 translation 시키면 케이스에 적합한 정답 이미지를 얻을 수 있다. 2.6과 달리 1번만 실행하면 되기 때문에 소요 시간 역시 정의할 수 있다. Fig.14는 케이스에 맞춘 정답 이미지와 Fig.13 (c)를 극좌표 변환을 이용해 회전각을 구한 결과이다.

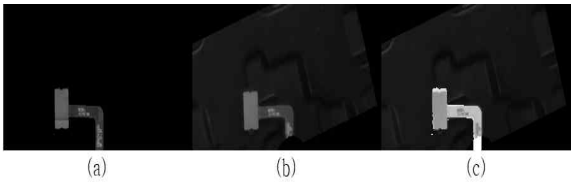

Figure 14. (a)정답 이미지, (b)회전 이미지, (c)오버레이 결과

5. 결론

본 논문은 영상 신호 처리를 사용해 FPCB Placing에 중요한 요소인 회전각을 추출하는 알고리즘을 소개하였다. 본 논문은 정확도와 속도로 조건을 세분화해 각각 FlowNet2.0과 극좌표 변환 기법을 개발하였다. 또한, RGB 센서 외 물리 센서를 사용하지 않고 Vision으로만 문제를 해결해 다양한 FPCB에 적용할 수 있다.

감사의 글

이 논문의 연구는 한국연구재단-현장맞춤형 이공계 인재양성 지원사업의 지원(No. 2017H1D8A1031522) 와 통상자원부의 재원으로 수행되었음 (10052967,재난재해 대응용 특수목적기계 통합제어시스템개발)

6. 참고문헌

[1] A. Dosovitskiy, P. Fischery, E. Ilg, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox, et al. FlowNet:Learning optical flow with convolutional networks. In IEEE International Conference on Computer Vision (ICCV), 2015.

[2] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, andT. Brox. FlowNet 2.0: Evolution of optical flow estimationwith deep networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[3] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. In IEEE transactions on image processing, 2004