KIPS Trans. Softw. and Data Eng.
Vol.11, No.2 pp.87~92
ISSN: 2287-5905 (Print), ISSN: 2734-0503 (Online)

https://doi.org/10.3745/KTSDE.2022.11.2.87

Image-Based Application Testing Method Using Faster D2-Net for Identification of the Same Image

Chun Hye-Won† · Jo Min-Seok† · Han Sung-Soo†† · Jeong Chang-Sung†††

ABSTRACT

Image-based application testing proposes an application testing method via image structure comparison. This test method allows testing on various devices without relying on various types of device operating systems or GUI. Traditional studies required the creation of a tester for each variant in the existing case, because it differs from the correct image for operating system changes, screen animation execution, and resolution changes. The study determined that the screen is the same for variations. The tester compares the underlying structure of the objects in the two images and extracts the regions in which the differences exist in the images, and compares image similarity as characteristic points of the Faster D2-Net. The development of the Faster D2-Net reduced the number of operations and spatial losses compared to the D2-Net, making it suitable for extracting features from application images and reducing test performance time.

Keywords : Application Test, Deep Learning, Image Matching, Feature Matching, Image Compare

동일 이미지 판별을 위해 Faster D2-Net을 이용한 이미지 기반의 애플리케이션 테스트 방법

전 혜 원† · 조 민 석† · 한 성 수†† · 정 창 성†††

요 약

이미지 기반 애플리케이션 테스트는 이미지 구조 비교를 통한 애플리케이션 테스트 방법을 제안한다. 이 연구는 다양한 디바이스 운영체제의 종류나 GUI에 의존 없이 다양한 기기에서 테스트가 가능하다. 기존 연구는 운영체제 변경, 화면상의 애니메이션 실행 그리고 해상도 변경의 경우 정답 이미지와 달라지기 때문에 기존의 경우 각각 변형마다 테스터를 생성해야 했다. 하지만 이 방법은 운영체제 변경, 해상도 크기의 변경, 화면상의 애니메이션 실행과 같은 변화가 발생해도 동일한 기준으로 판별하기 때문에 하나의 테스터로 테스트할 수 있다. 두 이미지의 객체들의 기본 구조를 비교하고 이미지에 차이가 존재하는 영역을 추출해서 Faster D2-Net의 특징점으로 이미지 유사성을 비교한다. Faster D2-Net 개발로 D2-Net보다 연산의 수와 공간적 손실을 줄여 애플리케이션 이미지에서 특징점을 추출하기 적합하고 수행 시간 단축이 가능했다.

키워드 : 애플리케이션 테스트, 딥러닝, 이미지 매칭, 특징점 매칭, 이미지 비교

1. 서 론

각종 애플리케이션과 스마트 기기들이 현재 시장의 빠른 흐름에 맞춰서 기획에서부터 출시까지 빠르게 진행되고 있다. 시장의 흐름에 맞춰 하나의 기기를 여러 애플리케이션에서, 하나의 애플리케이션을 다양한 기기 그리고 안드로이드와 IOS에서 제대로 실행되는지 테스트가 필요하다. 이 테스트는 정확성과 속도 모두 중요하다. 또 기업에서 제공하는 기술의 지속적인 유지 보수를 해야 사용자들의 이탈은 줄이고 유입은 늘릴 수 있다. 출시와 유지 보수를 위한 테스트는 정확성이 보장되어야 하고 빠르게 진행되어야 테스트 비용과 애플리케이션의 전체 비용을 줄일 수 있다. 그래서 운영체제 변경과 이미지의 아이콘 변화에도 한 개의 애플리케이션 테스트 구조에 맞춰서 하나의 테스터를 사용해야 해당 테스터의 범용성과 수익성을 높일 수 있다. 기존 연구는 테스트 과정에서 운영체제와 애플리케이션의 일대일 관계가 변경될 때마다 각각의 테스터를 생성한다. GUI(Graphical user interface) 테스트 방법으로 Record Play-back, 랜덤, 모델 기반 테스트가 있다. Model-Based Android GUI Testing[1]과 모바일 애플리케이션 GUI 테스팅[2] 연구는 각 GUI 단계와 발생하는 이벤트들을 사용해 테스트한다. 하지만 애플리케이션은 운영체제마다 GUI의 구성과 구조가 달라서 운영체제가 변경될 경우 기존의 테스트 정답 값을 사용할 수 없다. 이것은 테스트에 사용하는 기기가 변경될 경우에도 동일하다. 새로운 기기에 맞는 테스트 구조를 구현, 실험해야 한다. 이를 위해 GUI 기반이 아닌 이미지 기반의 애플리케이션 테스트를 제안

※ 이 논문은 2021년도 4단계 BK21 사업에 의하여 지원되었음.
※ 이 논문은 2021년 한국정보처리학회 춘계학술발표대회에서 "스크린 이미지 매칭을 위한 Faster D2-Net"의 제목으로 발표된 논문을 확장한 것임.
† 준 회 원 : 고려대학교 전기전자공학과 석사과정
†† 종신회원 : 강원대학교 자유전공학부 조교수
††† 정 회 원 : 고려대학교 전자공학과 정교수
Manuscript Received : June 29, 2021
First Revision : August 27, 2021
Accepted : October 8, 2021
* Corresponding Author : Jeong Chang-Sung(csjeong@korea.ac.kr)

한다. 이미지 기반의 테스트는 애플리케이션의 정답 화면과 테스트 화면을 비교해서 정답을 판별한다. 단순한 이미지 비교가 아닐 때 이미지 기반으로 테스트하기 어려운 두 가지의 경우가 발생하는데 첫 번째는 애플리케이션에서 이미지의 아이콘들은 동일하지만 애니메이션 발생, 두 번째는 기기의 변경으로 비교하려는 두 이미지상 아이콘의 크기 변형이다. 이 두 경우 모두 아이콘은 동일하기 때문에 동일한 이미지라고 판단해야 한다. 이 문제들을 해결하기 위해 feature matching 방식인 D2-Net[3]의 feature extraction 단계를 변경한 Faster D2-Net을 연구해 사용하였다. 이 연구를 통해 각종 변형에 유연하게 대응할 수 있게 기존의 GUI 또는 이벤트 발생 방식이 아닌 이미지를 기반으로 하는 테스터를 구현할 수 있고 하나의 테스터로 다양한 변형에도 테스트를 진행할 수 있다.

2. 관련 연구

2.1 ResNet

ResNet[4]은 모델의 깊이가 깊어질수록 성능이 좋아지는 것이 아닌 것을 증명했고 그 과정에서 Residual Block을 제안했다. 기존 신경망의 residual을 줄이기 위해 제안된 구조로 VGG19의 구조를 사용하고 추가로 합성곱과 Residual Block을 더해준다. 이를 통해 layer의 개수를 늘리고 특징 맵의 크기는 절반으로 줄여 연산하게 된다.

2.2 VGGNet

VGGNet[5]의 중심 연구는 모델을 깊게 구성하는 것이 성능에 어떤 결과를 가져오는지 연구하고 모델의 깊이가 깊어질수록 성능이 증가한다는 것을 확인했다. 실험을 위해 커널의 크기를 3x3으로 설정했다. 그 이유는 깊은 모델을 만들기 위해서는 연산을 해야 하는 이미지 크기를 천천히 감소시켜야 하기 때문이다. 총 6가지의 모델 중에서 VGG19과 VGG16의 차이점은 합성곱(convolution) 연산의 수이고 VGG19이 VGG16보다 더 layer가 많은 모델이다. 두 모델의 차이는 3번째 layer부터 존재한다. VGG19의 3번째 layer에서부터 합성곱 연산의 수가 하나 더 많아지면서 총합 3개의 합성곱 연산이 추가된다. Faster D2-Net에 적합한 구조를 찾기 위해서 기본이 되는 ResNet과 VGGNet의 구조와 연산이 필요했다. 해당 구조를 변경해서 실험했다.

2.3 SSIM

SSIM[6] Structureal Similarity Index의 약어로 영상 품질을 측정하기 위한 구조적 유사도 지수이다. 이미지의 품질 평가에서 영향력 있는 알고리즘 중 하나이다. 시각적으로 확인할 수 있는 화질의 차이와 유사도를 평가하기 위해서 사용된다. 기본적으로 ssim은 원본 이미지와 비교 대상 이미지의 휘도, 대조, 구조를 비교해서 계산한 수치값이다. Diff image를 비교할 때 수치값 계산을 위해 사용했다.

3. 이미지 기반의 애플리케이션 테스트

입력으로 받은 정답 이미지와 비교 이미지로 이 두 이미지의 비교를 위해 이미지 색상 변화와 de-noising으로 이미지

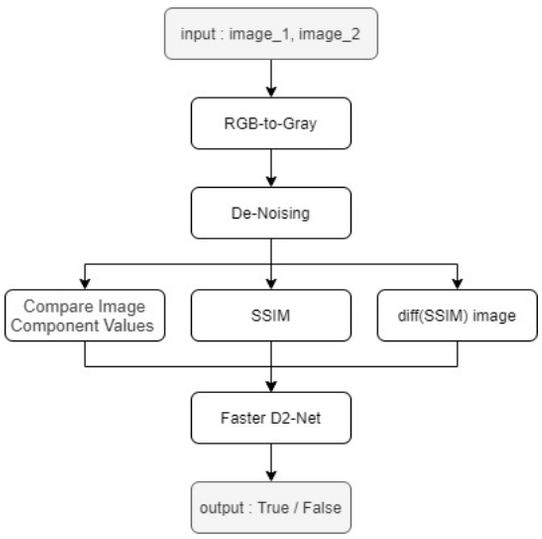

Fig. 1. Image Based Application Test Algorithm

전처리를 진행한다. 전처리가 진행된 이미지들로 이미지 성분 값 비교와 ssim, 차분 비교 이미지를 통해 작은 영역에서도 차이가 존재할 경우 그 영역만을 한 번 더 판단하기 위해 차이가 존재하는 영역에 Faster D2-Net을 수행한다. 최종적으로 이미지 성분 값 비교, ssim 값, 차분 비교 이미지의 3가지 값으로 1차 비교를 하고 Faster D2-Net으로 최종 결과값을 반환한다. Fig. 1로 이미지 기반의 애플리케이션 테스트의 전체 흐름을 확인할 수 있다.

3.1 Image PreProcessing

이미지 전처리 과정으로 Gray 스케일 변경과 de-noising을 수행한다. Gray 스케일 변경은 이미지를 비교할 때 아이콘의 단순한 색상 변경은 다양하게 발생해도 동일한 이미지라고 판단해야 하지만 이 경우를 제외한 모든 변화는 동일하지 않은 이미지라고 판단해야 하기 때문에 색상의 변화는 고려하지 않는다. 이를 위해 이미지의 스케일 변경이 필요하다. 이미지 비교를 위해서는 뚜렷한 객체와 객체 위치가 중요하다. 정확한 비교를 위해서 이미지에 존재하는 노이즈를 제거해야 하고 이를 통해서 보다 깨끗한 이미지를 구할 수 있다. 노이즈 제거를 위해서 이미지에 커널을 컨볼루션하게 되는데 블러링(blurring)의 종류에는 평균(Averaging blurring), 가우시안(Gaussian blurring), 미디안(Median blurring) 블러링 등이 있다. 실제 3가지 블러링을 실험해 본 결과 무작위 노이즈를 제거하는데 가장 결과가 좋은 미디안 블러링을 사용했다.

이미지 전처리 과정은 내비게이션 등의 화질이 낮고 이미지 안에 와이파이 연결 강도 표시 아이콘과 같은 변형되는 아이콘이 존재하는 이미지 비교에 특히 더 높은 정확도를 보였다.

3.2 Compare image component values

이미지를 1차 비교하는 방법으로 이미지의 성분 값 비교 방법을 연구했는데 두 이미지의 픽셀을 1:1로 비교하기에는 속도가 매우 느렸다. 그래서 정답 이미지 픽셀의 합으로 비교 이미지 픽셀의 합을 나눠서 총 픽셀 합의 비율을 구했다. 이 비

율은 두 이미지가 완전히 동일한 픽셀 값일 경우 픽셀 합의 비율 값은 1이고 유사한 이미지일수록 1의 가까운 값을 가질 것이다. 90% 이상의 총 픽셀 합의 비율일 경우 동일한 이미지라고 판단했다.

3.3 Compare diff image

이미지 성분 값 비교보다 더 세밀한 비교를 위해서 ssim 값과 차분 비교 이미지를 구했다. Fig. 2의 두 이미지를 예시로 설명하고자 한다.

두 이미지의 로테이션 애니메이션으로 차이가 발생하는 영역을 검은색 박스로 표시했다. 하지만 Fig. 2처럼 이미지에 로테이션(rotation), 아이콘의 크기 변화는 정답 이미지와 비교할 때 차이가 발생해서 테스트해야 하는 객체가 아니므로 동일한 이미지로 판별해야 한다. 그러므로 해당 차분 비교 이미지를 구해서 차이가 발생하는 영역만을 추출했다. Fig. 2의 두 이미지상에서 차이가 있는 부분은 Fig. 4와 동일한 로테이션 아이콘뿐이므로 차분 비교 이미지를 구하면 Fig. 3의 왼쪽 이미지에서 동일한 영역은 아무것도 표시되지 않고 차이가 발생하는 로테이션 영역만 검은색으로 표시했다.

Fig. 3의 오른쪽 이미지는 차이가 발생하는 영역만 캡처(capture) 한 이미지이다. 이미지 성분 값과 ssim 값으로 판별하지 못한 경우 차분 비교 이미지가 Faster D2-Net로 입력된다.

3.4 Faster D2-Net

Faster D2-Net은 2가지 상황을 처리할 수 있는데 첫 번째는 이미지상에 존재하는 아이콘 크기의 변화가 발생할 경우이고 두 번째는 동일한 아이콘의 애니메이션이 발생할 경우이다. 애니메이션은 로테이션 등의 이미지상에 아이콘은 모두 동일하지만 아이콘에 이미지 효과가 발생한 경우로 정의한다. 차분 비교 이미지에서 차이가 발생한 위치의 좌표를 구해서 실제 비교하려는 두 이미지에서 해당 좌표 영역의 이미지를 구했다. 두 이미지에 차이가 발생하는 영역을 자른 것이 Fig. 4이다.

Faster D2-Net으로 자른 이미지로 아이콘의 크기와 애니메이션 변화를 비교했다.

matching point는 inlier point와 outlier point로 나눌 수 있다. 이 둘의 구분은 해당 point가 데이터의 분포보다 오차가 크면 outlier, 아닐 경우 inlier로 구분한다. 전체 matching point에서 inlier point를 구하는 방법은 4장에서 설명한 fitting algorithm을 사용했다.

아이콘의 크기 비교는 Faster D2-Net으로 구한 두 이미지의 inlier point들 사이의 거리를 내림차순으로 정렬했다. 정렬된 point들로 두 이미지의 affine 맵 행렬을 각각 생성해서 크기 변환 행렬과 비교했다. 두 행렬이 배수 관계일 경우 동일한 아이콘에 크기 차이가 발생한 것이라고 판별했다.

아이콘의 애니메이션 비교는 Faster D2-Net으로 구한 두 이미지의 inlier point 값을 사용했다. 정확한 point들의 비율을 구하기 위해서 전체 matching point 중에서 inlier point의 존재 비율을 계산했다. 이 비교 비율이 85% 이상일 경우 동일한 아이콘에 애니메이션이 발생한다고 판단했다.

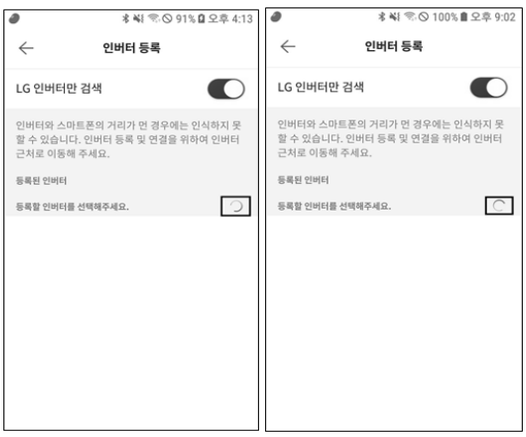

Fig. 2. Images with Correct Answer True

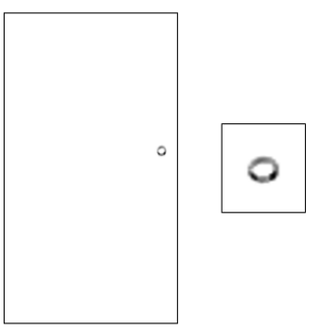

Fig. 3. Image of Difference and Region of Interest

Fig. 4. Faster D2-Net Input Image

4. Faster D2-Net 구조

Feature를 찾기 위한 과정은 detection, description 그리고 matching으로 나눌 수 있다. detection은 특징점을 추출하는 것이고 descriptor는 추출한 특징점과 그 주변의 색상, 방향, 크기, 밝기 등의 정보를 통해 픽셀들을 계산하는 것이다. 마지막 matching은 계산된 descriptor를 비교하여 매칭하는 것이다.

D2-Net에서는 VGG16을 사용해서 feature extraction을 진행하게 되고 descriptor와 detection을 한 번에 추출할 수 있게 된다. 해당 과정들은 기존의 연구들과는 다르게 detection을 마지막에 실행한다. 그 결과인 matching point들이 더 안정적이므로 논문 결과상 계절의 변화, 밤낮의 변화, 그림과 실제 이미지의 매칭에서도 높은 정확도를 보인다. 그러므로 변형이 발생하는 스마트 기기 화면의 이미지에도 높은 정확도를 보일 것으로 판단했고 실제 실험 결과 템플릿 매칭보다 더 높은 정확도를 확인할 수 있었다. 기존 D2-Net에

Table 1. 기존 D2-Net Training Architecture

Layer	Stride	Dilation	ReLU	Resolution
input(256x256) - 3C				x 1
conv(3x3) - 64C	1	1	O	x 1
conv(3x3) - 64C	1	1	O	x 1
pool(2x2) - max	2	1		x 1/2
conv(3x3) - 128C	1	1	O	x 1/2
conv(3x3) - 128C	1	1	O	x 1/2
pool(2x2) - max	2	1		x 1/4
conv(3x3) - 256C	1	1	O	x 1/4
conv(3x3) - 256C	1	1	O	x 1/4
conv(3x3) - 256C	1	1	O	x 1/4
pool(2x2) - max	2	1		x 1/8
conv(3x3) - 512C	1	1	O	x 1/8
conv(3x3) - 512C	1	1	O	x 1/8
conv(3x3) - 512C	1	1		x 1/8

Table 2. 기존 D2-Net Testing Architecture

Layer	Stride	Dilation	ReLU	Resolution
input(~1200x1600) - 3C				x 1
conv(3x3) - 64C	1	1	O	x 1
conv(3x3) - 64C	1	1	O	x 1
pool(2x2) - max	2	1		x 1/2
conv(3x3) - 128C	1	1	O	x 1/2
conv(3x3) - 128C	1	1	O	x 1/2
pool(2x2) - max	2	1		x 1/4
conv(3x3) - 256C	1	1	O	x 1/4
conv(3x3) - 256C	1	1	O	x 1/4
conv(3x3) - 256C	1	1	O	x 1/4
pool(2x2) - avg	1	1		x 1/4
conv(3x3) - 512C	1	2	O	x 1/4
conv(3x3) - 512C	1	2	O	x 1/4
conv(3x3) - 512C	1	2		x 1/4

서는 VGG16의 중간 특징맵을 사용하고 training과 testing 단계의 모델에서 마지막 layer의 구조가 다르다. 표의 C의 값은 채널의 수이다.

Table 1이 기존 D2-Net의 training 구조로 VGG16의 4번째 layer의 3번째까지의 합성곱 결과를 사용한다. Table 2는 기존 D2-Net의 testing 구조로 VGG16의 4번째 layer의 3번째까지의 특징맵을 사용하지만 training 구조와 다르게 4번째 layer에서 3개의 합성곱 모두에 dilation을 2로 설정한다. 즉 dilation convolution을 사용한다.

하지만 D2-Net은 이미지 한 장당 1초 이상 걸린다. 그 이유는 VGG 모델 계열이 ResNet과 같은 계열들보다 연산하는 특징 맵(feature map) 크기가 비교적 천천히 감소하기 때문이다. 특징맵 크기가 천천히 감소해서 더 많은 feature를 얻을 수 있지만 연산의 수가 많은 만큼 오래 걸리게 된다. 하지만 속도 단축을 위해 특징맵 감소를 빠르게 할 수는 없다. 그 이유는 특징맵은 작아지고 모델의 깊이는 깊어질수록 추출할 수 있는 특징점들의 수가 한정돼서 정확도가 떨어지기 때문이다. 그래서 연산을 해야 하는 특징맵의 크기는 천천히 감소시키되

Table 3. Faster D2-Net Training Architecture

Layer	Stride	Dilation	ReLU	Resolution
input(256x256) - 3C				x 1
conv(3x3) - 64C	1	1	O	x 1
conv(3x3) - 64C	1	1	O	x 1
pool(2x2) - max	2	1		x 1/2
conv(3x3) - 128C	1	1	O	x 1/2
conv(3x3) - 128C	1	1	O	x 1/2
pool(2x2) - max	2	1		x 1/4
conv(3x3) - 256C	1	1	O	x 1/4
conv(3x3) - 256C	1	1	O	x 1/4
conv(3x3) - 256C	1	1	O	x 1/4
conv(3x3) - 256C	2	1		x 1/8
pool(2x2) - max	1	1	O	x 1/8
conv(3x3) - 256C	1	1	O	x 1/8

Table 4. Faster D2-Net Testing Architecture

Layer	Stride	Dilation	ReLU	Resolution
input(~1200x1600) - 3C				x 1
conv(3x3) - 64C	1	1	O	x 1
conv(3x3) - 64C	1	1	O	x 1
pool(2x2) - max	2	1		x 1/2
conv(3x3) - 128C	1	1	O	x 1/2
conv(3x3) - 128C	1	2	O	x 1/2
pool(2x2) - max	2	1		x 1/4
conv(3x3) - 256C	1	2	O	x 1/4
conv(3x3) - 256C	1	2	O	x 1/4
conv(3x3) - 256C	1	2	O	x 1/4
conv(3x3) - 256C	1	2	O	x 1/4
pool(2x2) - avg	1	2		x 1/4

빠른 속도와 높은 정확도로 descriptor와 detection을 추출하기 위해서 feature extraction 모델을 변경했다.

정확도와 속도 둘 다 안정적으로 구하기 위해서 VGG16 대신 VGG19를 사용해서 testing과 training 구조를 변경했다. 특히 실제 촬영 이미지보다 스마트 기기의 이미지의 feature가 비교적 더 단순해서 VGG19의 사용이 더 적합했다. VGG19 모델에서 4번째 layer에 존재하는 1번째 합성곱 특징맵을 사용해서 기존 D2-Net의 training 모델과 동일한 해상도의 특징점을 사용했다.

Table 3은 Faster D2-Net의 training 모델이다. Training에서도 기존의 training보다 적은 채널과 합성곱 개수로 많은 특징점들을 빠른 속도로 얻을 수 있었다. Table 4는 Faster D2-Net의 testing 구조이다. 기존 D2-Net의 testing 모델에서는 training과 동일한 layer를 사용했다. 하지만 Faster D2-Net testing 모델은 3번째 layer까지만 사용했다. 3번째 layer의 모든 합성곱 연산에 dilation을 2로 설정했다.

Dilation 이란 dilated convolution으로 커널 사이의 간격을 더해서 receptive field의 크기를 키운다. 예를 들어 3x3 커널에 dilation의 값을 2로 주면 5x5 커널 된다. 즉, 9개의 파라미터를 가지고 5x5 커널과 동일해진다. Dilation

을 사용할 경우 적은 수의 파라미터로도 receptive field가 넓어진다. 그에 따라 공간적 차원의 손실은 줄이고 해당 연산마다 빠른 속도가 가능하다,

Testing 단계에서 속도를 줄이고 공간상의 손실 감소를 위해서 dilation이 필요했고 D2-Net보다 dilation 값을 키워서 3번째의 모든 layer에 dilation 연산을 했다.

최종 단계로 이미지들의 특징점들을 매칭시키고 그 결과로 얻은 matching point를 fitting algorithm에 사용해서 inlier point를 구했다. D2-Net의 경우 fitting algorithm 중에서 RANSAC(RANdom Sample Consensus)[7]을 사용했다. 하지만 RANSAC을 사용할 경우 정확한 특징점을 얻기 어렵고 데이터들의 모델링이 어려운 경우 오랜 시간이 걸리고 정확한 모델을 얻기 힘들다. 스마트 기기 화면 이미지에서 유사한 아이콘들이 여러 개 존재할 경우 matching point으로 정확하게 모델링 하는 것이 어려워서 RANSAC으로 얻을 수 있는 매칭 점의 수가 매우 부족했다. 그래서 Faster D2-Net에서는 RANSAC을 사용하지 않고 MAGSAC(Marginalizing Sample Consensue)[8]을 사용했다. MAGSAC은 정확한 특징점이 많지 않아도 적은 수의 반복으로 RANSAC보다 정확하고 빠르게 matching point을 구할 수 있었다. 그 결과 스마트 기기의 이미지에서도 많은 수의 point를 얻을 수 있었다.

5. 실험 방법과 결과

이미지 기반 애플리케이션 테스트의 최종 실험은 정답 이미지와 테스트할 기기의 이미지를 가지고 진행했다. 실험에 사용한 최종 이미지 수는 300장이고 테스트할 기기의 이미지에 랜덤으로 아이콘의 크기 변화와 애니메이션 변화가 존재한다. 총 300장의 이미지 중에서 6장의 이미지를 제외하고 모두 정답 값과 동일했다.

Faster D2-Net과 이미지 비교 과정(이미지 성분 비교, 차분 비교)을 모두 포함했을 때 Android 애플리케이션에서는 98%의 정확도와 0.78초의 속도를 확인할 수 있었고 iOS 애플리케이션에서는 94%의 정확도와 0.82의 속도를 Table 5를 통해 확인할 수 있었다. 즉, Android와 iOS의 결과로 운영체제의 변화에도 유연한 사용이 가능한 것을 확인할 수 있었다.

5.1 Faster D2-Net

스마트 기기 화면을 캡처해서 데이터 셋을 구성하고 평가하였는데 해상도의 변화 즉, 기기의 변화를 평가하기 위해 한 가지 애플리케이션 이미지 당 총 3종류의 해상도(1440x2880, 1440x2560, 1080x1920)로 구성했다. 또 한 이미지에 공백, 색상과 크기의 변형이 존재하기 때문에 Faster D2-Net이 변형에도 정답 영역을 잘 찾는지 확인할 수 있었다.

먼저 VGG 계열 중에 batch normalization(bn)을 포함한 모델이 주로 활용되기 때문에 batch normalization이 포함된 3개의 모델을 선택했다. batch normarlization[9]은 평균, 분산의 과정이 학습할 때 포함되어 같이 조정되는 것이다. 이 방법으로 학습 과정에서 gradient vanishing과 exploding 문제로 학습이 안되거나 해당 데이터에 overfitting 되는 경우를 방지하고 학습 속도를 높일 수 있기 때문에 주로 활용된다. VGG13, VGG16 그리고 VGG19 중 어떤 모델이 애플리케이션 이미지에 가장 적합한지 알아보기 위해 layer의 변형 없이 실험했다.

Table 6에서 3개의 모델 모두 90% 이상의 높은 정확도를 보였지만 속도가 3초 이상이기 때문에 테스트하기에 적합하지 않았다.

Table 7은 D2-Net과 Faster D2-Net의 정확도와 속도의 비교 결과이자 Faster D2-Net의 최종 결과이다.

LG ThinQ 앱, 안드로이드 기본 앱들로 총 5개의 애플리케이션을 결과 확인에 사용했다. 기존 D2-Net은 90% 이하의 정확도와 1.7초대의 속도이지만 Faster D2-Net은 D2-Net보다 1% 이상의 정확도 향상과 1초 이상의 속도가 감소했다는 것을 확인할 수 있다. 1초 이상의 속도 감소로 사용자가 애플리케이션 테스트하기에 더 용이했다. Faster D2-Net의 수행 시간이 0.7초이고 이미지 비교 과정(이미지 성분 비교, 차분 비교)이 0.08초 걸렸다.

이미지에서 매칭된 영역을 표시하기 위해서 추출된 point를 기준으로 매칭할 영역의 크기만큼 초록색 박스로 표시했다. Fig. 5는 Android 기기에서 인터넷 애플리케이션을 사용한 것이고 Google을 실행한 이미지이다. Fig. 5에서 오른쪽 상단의 '컬렉션'과 '연결 가능한 기기 알림'이 매칭할 이미지이고 왼쪽의 기기 전체 이미지에서 왼쪽 하단의 박스 영역이 Faster D2-Net으로 매칭된 영역이다. Fig. 5는 Faster D2-Net의 기본 성능을 확인하기 위해 feature matching을 표시한 이미지이고 Fig. 2는 애니메이션이 존재하는 이미지이다.

Table 5. Image Based Application Test Results

	True	False	Total	Ratio	speed
Android	632	9	641	98%	0.78
iOS	356	20	376	94.6%	0.82

Table 6. VGG Model Result of D2-Net

model	Speed	True	False	Acc.
vgg13bn	3.43sec	939	67	0.93%
vgg16bn	2.99sec	933	73	0.92%
vgg19bn	3.4sec	943	63	0.93%

Table 7. D2-Net and Faster D2-Net Result

Mothod	Accuracy(%)	Speed(sec.)
D2-Net	89.30	1.79
Faster D2-Net	90.20	0.70

Fig. 5. Sample Figure

6. 결 론

이 논문은 이미지를 기반으로 하는 애플리케이션 테스트 방법을 제안한다. 기존의 GUI 방식과 랜덤 방식은 테스트하려는 애플리케이션이나 스마트 기기의 운영체제와 종류에 의존적이다. 위의 문제를 이미지 기반 테스트 방법으로 해결함으로써 다양한 이미지 변화에도 동일한 테스트가 가능하고 동일한 정확도를 얻을 수 있다. 이미지상 발생하는 변화는 Faster D2-Net을 사용해 동일한 영역을 찾을 수 있다. D2-Net을 애플리케이션 이미지와 테스트 수행 속도에 맞는 Faster D2-Net로 추가로 변경했다.

평균 96.3%의 정확도와 0.8의 속도로 높은 정확도와 함께 테스트에 적합한 속도로 테스트할 수 있다. 이 테스트 방법으로 스마트 기기의 운영체제와 종류, 이미지의 변화에도 범용적으로 테스트가 가능하기 때문에 자동차의 내비게이션이나 자전거 등의 작은 스마트 기기에 존재하는 다양한 애플리케이션들의 테스트도 가능하다.

References

[1] Y. M. Baek and D. H. Bae, "Automated model-based android gui testing using multi-level gui comparison criteria," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, pp.238-249, 2016.

[2] W. Yang, M. R. Prasad, and T. Xie, "A grey-box approach for automated GUI-model generation of mobile applications," in *International Conference on Fundamental Approaches to Software Engineering*, Springer, Berlin, Heidelberg, pp.250-265, 2013.

[3] M. Dusmanu, et al., "D2-net: A trainable cnn for joint detection and description of local features," *arXiv preprint arXiv:1905.03561*, 2019.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778, 2016.

[6] Z. Wang, et al., "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, Vol.13, No.4, pp.600-612, 2004.

[7] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, Vol.24, No.6, pp.381-395, 1981.

[8] D. Barath, J. Matas, and J. Noskova, "Magsac: Marginalizing sample consensue," in P*roceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.10197-10205, 2019.

[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, PMLR, pp.448-456. 2015.

전 혜 원

https://orcid.org/0000-0002-6761-5865
e-mail : uclacarol@korea.ac.kr
2020년 한국외국어대학교 컴퓨터전자시스템공학과(학사)
2020년 ~ 현 재 고려대학교 전기전자공학과 석사과정
관심분야 : 컴퓨터 비전, 딥러닝, 분산 시스템

조 민 석

https://orcid.org/0000-0002-7483-4142
e-mail : jms0923@korea.ac.kr
2020년 충남대학교 컴퓨터공학과(학사)
2020년 ~ 현 재 고려대학교 전기전자공학과 석사과정
관심분야 : 컴퓨터 비전, 딥러닝, 분산 시스템

한 성 수

https://orcid.org/0000-0002-4915-6247
e-mail : sshan1@kangwon.ac.kr
1998년 경상대학교 정보통신공학과(학사)
2005년 순천향대학교 정보통신공학과(석사)
2019년 고려대학교 영상정보처리협동과정 (박사)
2015년 ~ 2016년 오리온 테크놀리지 이사
2018년 ~ 2019년 순천향대학교 조교수
2019년 ~ 현 재 강원대학교 자유전공학부 조교수
관심분야 : 컴퓨터 교육, 영상정보처리, 병렬처리, 딥러닝

정 창 성

https://orcid.org/0000-0001-9654-8406
e-mail : csjeong@korea.ac.kr
1981년 서울대학교 전기공학과(학사)
1984년 Northwestern University 전자계산학과(석사)
1987년 Northwestern Unviersity 전자계산학과(박사)
1987년 ~ 1992년 포항공과대학교 전자계산학과 조교수
1992년 ~ 1998년 고려대학교 전자공학과 부교수
1998년 ~ 현 재 고려대학교 전자공학과 정교수
관심분야 : 컴퓨터 비전, 유비쿼터스 컴퓨팅, 네트워크 가상 컴퓨팅, 클라우드 컴퓨팅