

Bottom-up Object Detection by Grouping Extreme and Center Points

📎 파일	비어 있음
🔗 URL	비어 있음
☰ 분야	Object Detection
☰ 이해도	비어 있음
# 발행년도	2019
+ 속성 추가	

ଓ 댓글 추가

0. Abstract

4개의 극 점(가장 위쪽, 가장 아래쪽, 가장 오른쪽, 가장 왼쪽) 과 하나의 중심점을 통해 물체 탐지

Top-down 방식이 아닌 Bottom-up 방식

1. Introduce

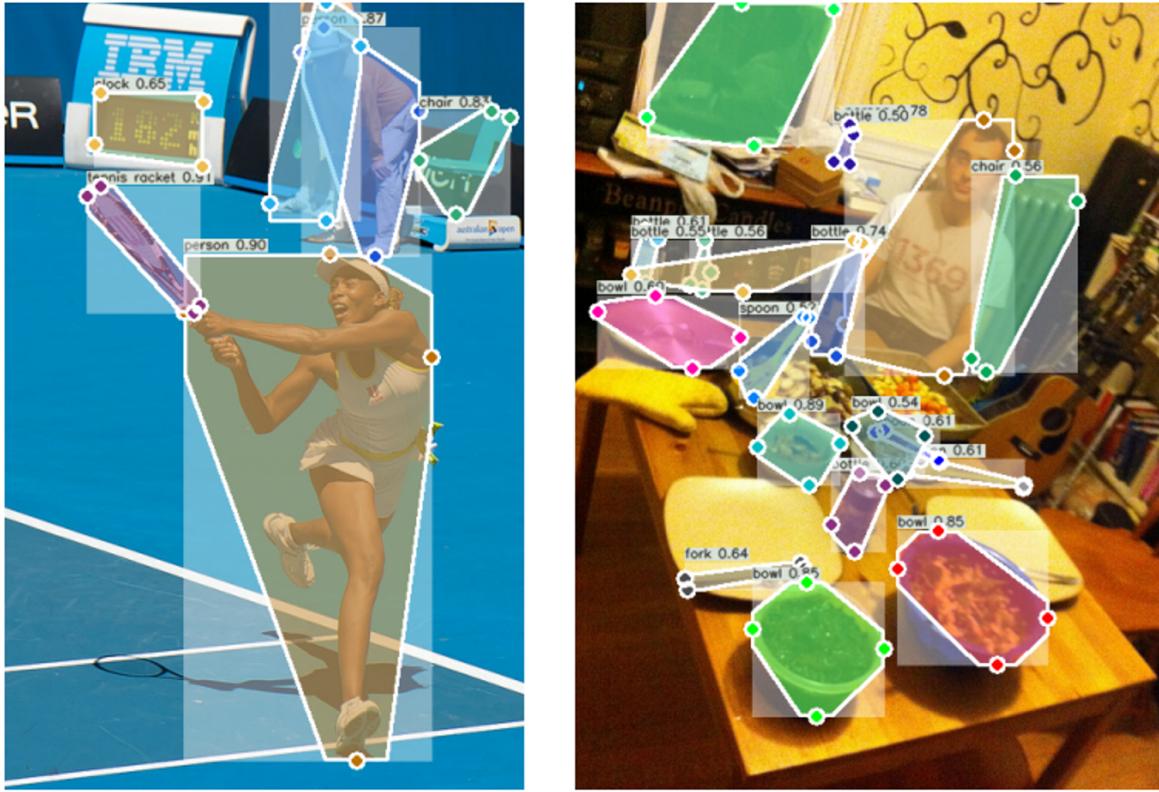


Figure 1: We propose to detect objects by finding their extreme points. They directly form a bounding box , but also give a much tighter octagonal approximation of the object.

- 수많은 anchorbox 중 물체가 포함된 box는 극소수 → CornerNet
 - CornerNet은 좌측 상단 모서리, 우측 하단 모서리로 2개의 특징점을 사용
 - ExtremeNet은 CornerNet을 계승했지만 2가지 다른 점을 가짐
1. 포인트 정의

CornerNet은 bounding box의 또다른 형태 → 모서리 부분은 객체 외부인 경우가 많음

극점은 물체 위에 있으며(ex. 사람의 최상위 극점은 머리, 자동차의 최하위 극점은 바퀴)
극점 감지가 더 쉬움

2. 기하학적 그룹화

네 개의 극점을 직접 연결

2. Related Work

2-stage object detector : localization → classification

our model : localization이 필요하지 x, 네 개의 극점도 객체를 표현하기에 효과적이며 bounding box 만큼 많은 정보를 제공함

1-stage object detector : localization & classification

our model : 1-stage detector

but 앵커를 설정하는 것 대신, 4개의 극점과 1개의 중심을 감지한다.

앵커의 scale, aspect ration를 설정하는 것 대신 해당 위치가 key point(극점)이 될 확률만 예측한다. (CornerNet과 극점 예측 방식이 비슷한 만큼 손실 함수가 매우 유사하다)

HourglassNet

heatmap을 얻기 위해 backbone network로 사용

모래시계 모양 구조 → local 정보(ex. 사람의 손, 발) 와 full body 정보(ex. 사람 전체)를 모두 파악하기 위해서

모래시계 구조인 hourglass module을 여러번 쌓아 하나의 output을 다음 module의 input으로 사용한다.

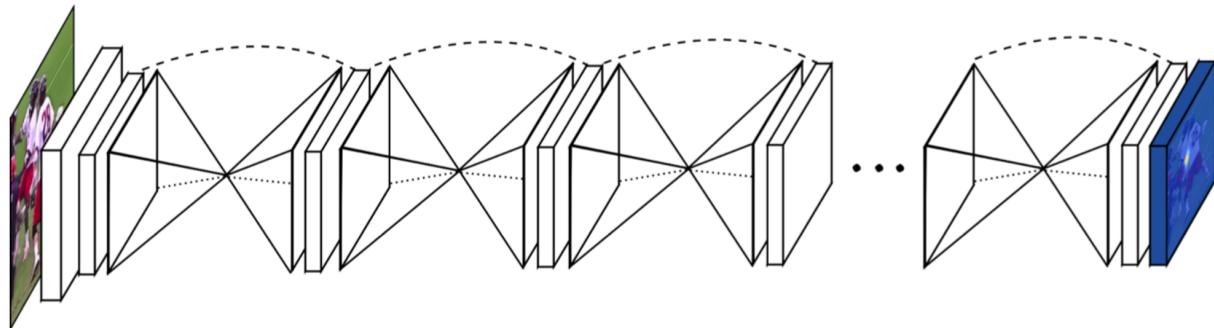
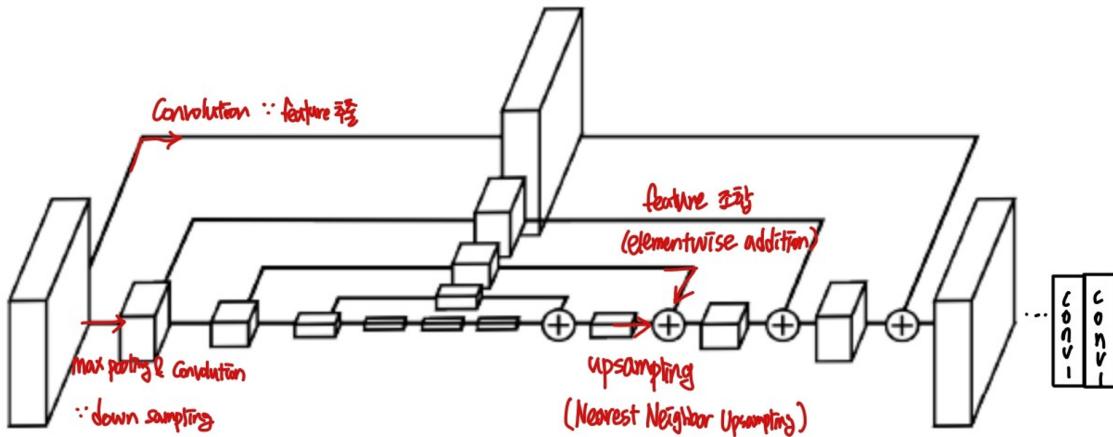


Fig. 1. Our network for pose estimation consists of multiple stacked hourglass modules which allow for repeated bottom-up, top-down inference.



- low resolution 일 때

max pooling & convolution : down sampling 수행

convolution : feature map 추출

- high resolution 일 때

Nearest neighbor up sampling 기법 사용

추출된 feature map과 concat

3. Preliminaries

CornerNet

CornerNet은 HourglassNet을 객체 감지기로 사용하여 key point 추정을 수행한다.

corner의 positive location이 negative location 보다 극단적으로 수가 적으므로 focal loss(불균형 문제 해결 가중치 설정 방법)를 사용했다.

$$L_{det} = -\frac{1}{N} \sum_{i=1}^H \sum_{j=1}^W \begin{cases} (1 - \hat{Y}_{ij})^\alpha \log(\hat{Y}_{ij}) & \text{if } Y_{ij} = 1 \\ (1 - \hat{Y}_{ij})^\beta (\hat{Y}_{ij})^\alpha \log(1 - \hat{Y}_{ij}) & \text{o.w.} \end{cases}, \quad (1)$$

where α and β are hyper-parameters and fixed to $\alpha = 2$ and $\beta = 4$ during training. N is the number of objects in the image.

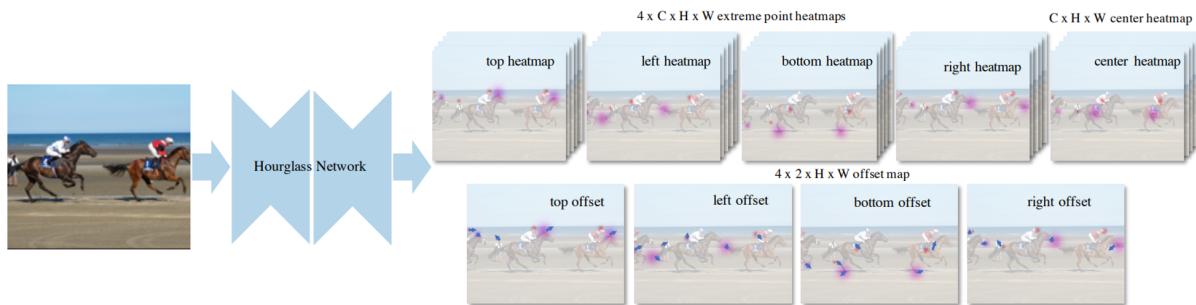
HourglassNet이 down-sampling을 하며 잃어버리는 정보들을 조정해주는 offset map은 smooth L1 Loss를 통해 학습된다.

$$L_{off} = \frac{1}{N} \sum_{k=1}^N SL_1(\Delta^{(a)}, \vec{x}/s - \lfloor \vec{x}/s \rfloor), \quad (2)$$

where s is the down-sampling factor ($s = 4$ for Hourglass-Net), \vec{x} is the coordinate of the keypoint.

extreme points를 찾는 데 CornerNet의 아키텍처와 loss를 사용했지만, CornerNet의 Associative embedding에 해당하는 loss는 사용하지 않고 후에 서술할 Center Grouping으로 대체했다.

4. ExtremeNet for Object detection



ExtremeNet은 hourglassNet을 사용하여 클래스당 5개의 키포인트(4개의 extreme point + 1개의 center point)를 감지

CornerNet의 훈련 설정, 손실 함수, offset 예측을 따름

따라서 hourglassNet의 결과물은 다음과 같다. ($C =$ 클래스 수)

- $4 \times C \times H \times W$ extreme point heatmap
- $C \times H \times W$ center heatmap
- $4 \times 2 \times H \times W$ offset map
 - hourglassNet의 down sampling에서 원래 이미지 크기로 복원하는 과정에서 소수점 내림을 사용하므로 x, y 에 대한 offset을 계산해준다.

4.1 Center Grouping

Algorithm 1: Center Grouping

Input : Center and Extremepoint heatmaps of an image for one category: $\hat{Y}^{(c)}, \hat{Y}^{(t)}, \hat{Y}^{(l)}, \hat{Y}^{(b)}, \hat{Y}^{(r)} \in (0, 1)^{H \times W}$
Center and peak selection thresholds: τ_c and τ_p

Output: Bounding box with score

// Convert heatmaps into coordinates of keypoints.

// $\mathcal{T}, \mathcal{L}, \mathcal{B}, \mathcal{R}$ are sets of points.

$\mathcal{T} \leftarrow \text{ExtractPeak}(\hat{Y}^{(t)}, \tau_p)$
 $\mathcal{L} \leftarrow \text{ExtractPeak}(\hat{Y}^{(l)}, \tau_p)$
 $\mathcal{B} \leftarrow \text{ExtractPeak}(\hat{Y}^{(b)}, \tau_p)$
 $\mathcal{R} \leftarrow \text{ExtractPeak}(\hat{Y}^{(r)}, \tau_p)$

for $t \in \mathcal{T}, l \in \mathcal{L}, b \in \mathcal{B}, r \in \mathcal{R}$ **do**

// If the bounding box is valid

if $t_y \leq l_y, r_y \leq b_y$ **and** $l_x \leq t_x, b_x \leq r_x$ **then**

// compute geometry center

$c_x \leftarrow (l_x + r_x)/2$

$c_y \leftarrow (t_y + b_y)/2$

// If the center is detected

if $\hat{Y}_{c_x, c_y}^{(c)} \geq \tau_c$ **then**

Add Bounding box (l_x, t_y, r_x, b_y) with score

$(\hat{Y}_{t_x, t_y}^{(t)} + \hat{Y}_{l_x, l_y}^{(l)} + \hat{Y}_{b_x, b_y}^{(b)} + \hat{Y}_{r_x, r_y}^{(r)} + \hat{Y}_{c_x, c_y}^{(c)})/5.$

end

end

end

Y : 0과 1로 이루어진 $H \times W$ 크기인 heat map

τ : threshold

$\text{ExtractPeak}(Y, \tau)$: 주변 3×3 픽셀값 보다 τ 보다 큰 픽셀 값

(1).

0.1	0.3	0.6
0.12	0.4	0.55
0.5	0.3	0.5

(2).

0.1	0.3	0.6
0.12	0.8	0.55
0.5	0.3	0.5

- (1) 그림은 주변 3×3 픽셀 값보다 충분히 큰 값($\tau_p = 0.1$)을 가지지 못하므로 *peak*가 아니다
- (2) 그림은 주변 3×3 픽셀 값보다 충분히 큰 값($\tau_p = 0.1$)을 가지므로 *peak*이다.

4.2 Ghost box suppression

같은 클래스를 갖는 세 물체가 나란히 있는 이미지일 경우 “Ghost box” 가 생긴다.

이를 제거하기 위해 후처리를 진행함 → non-max suppression (특정 bounding box에 포함된 모든 상자의 점수 합계가 해당 상자의 3배를 초과하는 경우 점수 자체를 2로 나눔)

4.3 Edge aggregation



(a) Original heatmap.



(b) After edge aggregation.

대형 버스처럼 직사각형 형태의 물체의 extreme point는 물체의 가장자리를 따라 쭉 존재한다.

한 지점만 extreme point로 잡는 것이 아니라, 값이 작은 여러 지점을 extreme point로 예측한다. → 가중치 적용 방식 변화

$$\tilde{Y}_m = \hat{Y}_m + \lambda_{aggr} \sum_{i=i_0}^{i_1} N_i^{(m)},$$

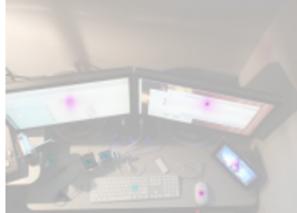
$$\lambda_{aggr} = 0.1$$

5. Experiments

Extreme point heatmap



Center heatmap



Octagon mask



Extreme points+DEXTR [29]

