HorizonNet 논문 해석

길다영

2021. 07. 12

https://github.com/sunset1995/HorizonNet 참고.

1D Layout Representation
Feature Extractor
Recurrent Nueral Network
for Capturing Global Infor-
mation

3. Approach

목차
Contents

4. Experiments

3.1 HorizonNet

- 1D Layout Representation
- Feature Extractor
- Recurrent Nueral Network for Capturing Global Infor-mation

3.2 Post-processing

- Recovering the Floor and Ceiling Planes
- Recovering Wall Planes

3.3 Pano Stretch Data Augmentation

4.1 Datasets

4.2 Training Details

4.3 Cuboid Room Results

4.4 Ablation Study

4.5 Non-cuboid Room Results

3.1 HorizonNet

3. Approach

The goal of our approach is to estimate Manhattan room layout from a panoramic image that covers 360° H-FOV. Unlike conventional dense prediction (target output size = $\mathcal{O}(HW)$) for layout estimation using deep learning [4, 9, 7, 15, 19, 23, 33], we formulate the problem as regressing the boundaries and classifying the corner for each column of image (target output size = $\mathcal{O}(W)$). The proposed HorizonNet trained for predicting the $\mathcal{O}(W)$ target is presented in Sec. 3.1. In Sec. 3.2, we introduce a simple yet fast and effective post-processing procedure to derive the layout from output of HorizonNet. Finally in Sec. 3.3, we introduce *Pano Stretch Data Augmentation* which effectively augments the training data on-the-fly by stretching the image and ground-truth layout along x or z axis (Fig. 5).

All training and test images are pre-processed by the panoramic image alignment algorithm mentioned in [36]. Our approach exploits the properties of the aligned panoramas that the wall-wall boundaries are vertical lines under equirectangular projection. Therefore, we can use only one value to indicate the column position of wall-wall boundary instead of two (each for a boundary endpoint).

3.1. HorizonNet

Fig. 2 shows an overview of our network, which comprises a feature extractor and a recurrent neural network. The network takes a single panorama image with the dimension of $3 \times 512 \times 1024$ (channel, height, width) as input.

목표 : 360도 파노라마 이미지로부터 Manhattan room layout 추정

경계를 회귀시키고 이미지의 각 열의 모서리를 분류함으로써 문제를 공식화함.

3.1
: 대상을 예측하기 위해 훈련된 HorizonNet이 제시되어 있음.

3.2
: HorizonNet의 결과로부터 Layout 도출하는 후처리 결과 소개.

3.3
: Pano Stretch Data Augmentation (x축이나 z축을 따라 ground-truth layout과 image를 늘리고 줄임으로써 훈련 데이터를 효과적으로 확장.)

접근 방식 : 정렬된 파노라마의 특성(등각 투영 하에 벽-벽 경계는 수직선임) 이용.
두개의 값(각 경계 끝점) 대신 하나의 값(벽-벽 경계의 열 위치)만을 사용 가능.

Fig2
: 특징 추출기와 순환 신경망을 이룸.
: 3*512*1024 (channel, height, width)의 단일 파노라마 이미지를 input으로 취함.

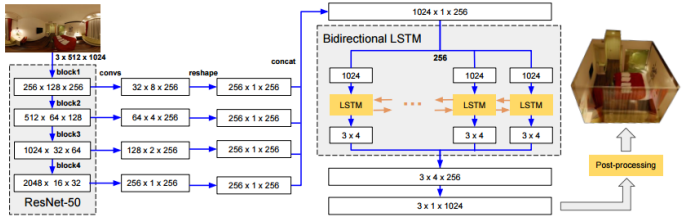
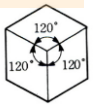


Figure 2: An illustration of the HorizonNet architecture.

Manhattan World 가정 : 영상에 나타나는 평면들은 3차원상에서 서로 직교하는 평면들 로만 이루어져 있다.
등각 투영 : 육면체를 투영할 때, 3축의 선분이 각각 120도로 이루어져 윤곽이 정육각형이 되는 투영.



3.1 HorizonNet

1D Layout Representation

1D Layout Representation: The size of network output is $3 \times 1 \times 1024$. As illustrated in Fig. 3, two of the three output channels represent the ceiling-wall (y_c) and the floor-wall (y_f) boundary position of each image column, and the other one (y_w) represents the existence of wall-wall boundary (*i.e.* corner). The values of y_c and y_f are normalized to $[-\pi/2, \pi/2]$. Since defining y_w as a binary-valued vector with 0/1 labels would make it too sparse to detect (only 4 out of 1024 non-zero values for simple cuboid layout), we set $y_w(i) = c^{dx}$ where i indicates the ith column, dx is the distance from the ith column to the nearest column where wall-wall boundary exists, and c is a constant. To check the robustness of our method against the choice of c, we have tried $0.6, 0.8, 0.9, 0.96, 0.99$ and get similar results. Therefore, we stick to $c = 0.96$ for all the experiments. One benefit of using 1D representation is that it is less affected by zero dominant backgrounds. 2D whole-image representations of boundaries and corners would result in 95% zero values even after smoothing [36]. Our 1D boundaries representation introduces no zero backgrounds because the prediction for each component of y_c or y_f is simply a real-valued regression to the ground truth. The 1D wall-wall (corners) representation also changes the peak-background ratio of ground truth from $\frac{2N}{512 \cdot 1024}$ to $\frac{N}{1024}$ where N is the number of wall-wall corners. Therefore, the 1D wall-wall representation is also less affected by zero-dominated background. In addition, computation of 1D compact output is more efficient compared to 2D whole-image output. As depicted in Sec. 3.2, recovering the layout from our three 1D representations is simple, fast, and effective.

네트워크 출력 크기 : 3 * 1 * 1024

Fig3
: Yc(천장-벽), Yf(바닥-벽), Yw(벽-벽, corner)

Figure 3: Visualization of our 1D ground truth represen-

Yc, Yf : $[-\pi/2, \pi/2]$로 정규화

Yw
: 0과 1의 이진 벡터로 정의하면 감지하기 어려움(단순 입방체 레이아웃의 경우 1024개 값 중 4개만 0이 아님)
그래서 $y_w(i) = c^{dx}$ 설정
(i : i번째 열, dx : i번째 열에서 가장 가까운 열까지의 거리, c : 상수(실험 결과 0.96 고수))

1D 표현의 이점 ❶
zero 지배적인 배경에 영향을 덜 받음. (Yc와 Yf의 각 성분에 대한 예측은 ground truth에 대한 실제 값 회귀이기 때문)
경계와 corner의 2D 표현 : 평활화 후 95%가 zero 값임.

1D 표현의 이점 ❷
1D 벽-벽 표현 : $\frac{2N}{512*1024}$에서 $\frac{N}{1024}$로 ground truth의 peak-background 비율 변경.(N : 벽-벽 corner 개수)
그래서 Zero 지배적인 배경에 영향을 덜 받음.

1D 표현의 이점 ❸
2D의 출력에 비해 1D의 출력이 연산이 더 빠르고, 간단하며 효율적임

1 D Layout Representation

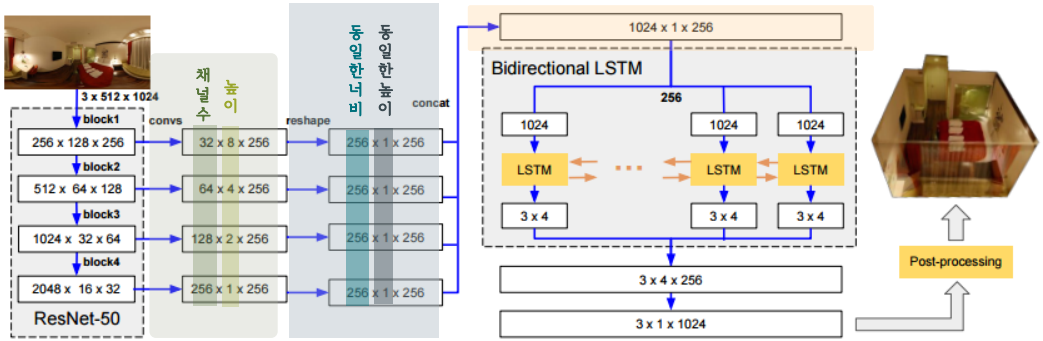


Figure 2: An illustration of the HorizonNet architecture.

ResNet-50 참고
https://velog.io/@arittung/ResNet-50

Feature Extractor: We adopt ResNet-50 [11] as our feature extractor. The output of each block of ResNet-50 has half spatial resolution compared to that of the previous block. To capture both low-level and high-level features, each block of the ResNet-50 contains a sequence of convolution layers in which the number of channels and the height is reduced by a factor of 8 ($= 2\times2\times2$) and 16 ($= 4\times2\times2$), respectively. More specifically, each block contains three convolution layers with $4 \times 1, 2 \times 1, 2 \times 1$ kernel size and stride, and the number of channels after each Conv is reduced by a factor of 2. All the extracted features from each layer are upsampled to the same width 256 (a quarter of input image width) and reshaped to the same height. The final concatenated feature map is of size $1024 \times 1 \times 256$. The activation function after each Conv is ReLU except the final layer in which we use Sigmoid for y_w and an identity function for y_c, y_f. We have tried various settings for the feature extractor, including deeper ResNet-101, different designs of the convolution layers after each ResNet block, and upsampling to the image width 1024, and find that the results are similar. Therefore, we stick to the simpler and computationally efficient setting.

특징 추출기로써 ResNet-50 선택.
ResNet50의 각 block의 출력은 이전 block에 비해 공간 분해능이 절반.

ResNet-50의 각 block에는 **채널수가 8(2*2*2)**로 감소, **높이는 16(4*2*2)**로 감소하는 convolution layer가 포함되어 있음.
Low-level 및 high level의 기능을 모두 캡쳐 하기 위함.

각 block은 3개의 convolution layers를 가짐. 4x2, 2x1, 2x1의 커널 사이즈와 stride 가짐.
각 conv의 채널 수는 1/2로 감소.

동일한 너비 256(입력 이미지 너비의 ¼)로 upsampled 되고, **동일한 높이로** 재구성됨.

마지막으로 연결된 기능 지도의 size : 1024 x 1 x 256

ResNet 101 등의 다양한 기능 추출기, 다양한 convolution layer 설계, 1024로 upsampling 하는 등 다양한 설정을 시도했지만 결과는 비슷했음.

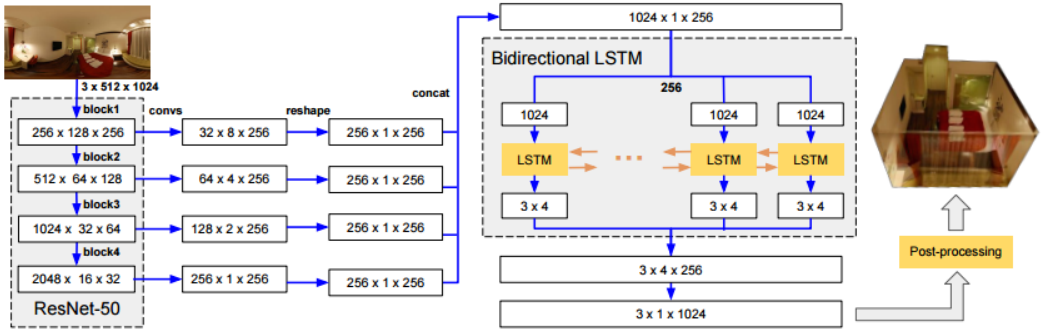


Figure 2: An illustration of the HorizonNet architecture.

Recurrent Neural Network for Capturing Global Information: Recurrent neural networks (RNNs) are capable of learning patterns and long-term dependencies from sequential data. Geometrically speaking, any corner of a room can be roughly inferred from the positions of other corners; therefore, we use the capability of RNN to capture global information and long-term dependencies. Intuitively, because LSTM [13], a type of RNN architecture, stores information about its prediction for other regions in the cell state, it has the ability to predict for occluded area accurately based on the geometric patterns of the entire room. In our model, RNN is used to predict y'_c, y'_f, y'_w column by column. That is, the sequence length of RNN is proportional to the image width. In our experiment, RNN predicts for four columns instead of one column per time step, which requires less computational time without loss of accuracy. As the y_c, y_f, y_w of a column is related to both its left and right neighbors, we adopt the bidirectional RNN [25] to capture the information from both sides. Fig. 7 and Table 1 demonstrate the difference between models with or without RNN.

RNN 사용 이유
: 순차 데이터 로부터 패턴과 장기 의존성 학습 가능.
: 방의 모서리는 다른 모서리의 위치에서 대략적으로 추론 가능

RNN 참고
https://velog.io/@arittung/ANN-DNN-CNN-RNN#rnn-recurrent-neural-network

감사합니다.

The End