HorizonNet 논문 해석

길다영

2021. 07. 12

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

목차
Contents

3. Approach

3.1 HorizonNet

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

3.2 Post-processing

- Recovering the Floor and Ceiling Planes
- Recovering Wall Planes

3.3 Pano Stretch Data Augmentation

4. Experiments

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.

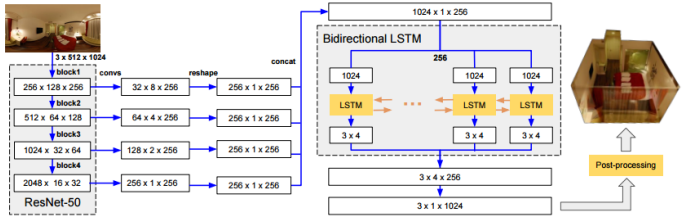
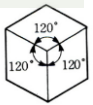


Figure 2: An illustration of the HorizonNet architecture.

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

3.1 HorizonNet

Feature Extractor

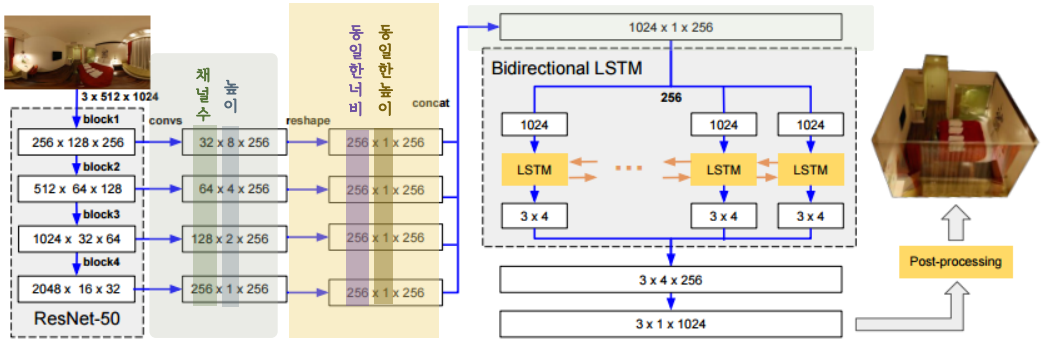


Figure 2: An illustration of the HorizonNet architecture.

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.

ResNet-50 참고

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

ResNet 50 이외의 다른 특징 추출기 선택하여 진행해보기

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.

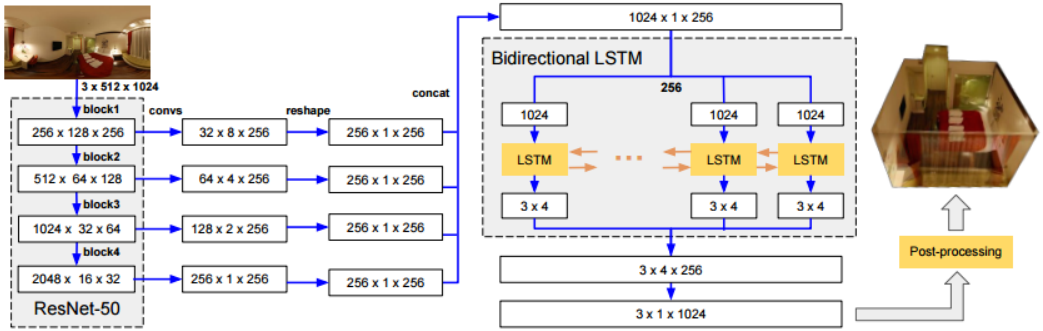


Figure 2: An illustration of the HorizonNet architecture.

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 참고

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

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

LSTM
- RNN의 종류 중 하나.

이 부분(RNN, LSTM)은 추후 더 공부하여 완성할 것임

3.2 Post-processing

3.2. Post-processing

We recover general room layouts that are not limited to cuboid under following assumptions: *i*) intersecting walls are perpendicular to each other (Manhattan world assumption); *ii*) all rooms have the one-floor-one-ceiling layout where floor and ceiling are parallel to each other; *iii*) camera height is 1.6 meters following [32]; *iv*) the pre-processing step correctly align the floor orthogonal to y-axis.

As described in Sec. 3.1, raw outputs of our deep model $y'_f, y'_c, y'_w \in \mathcal{R}^{1024}$ contain the layout information for each image column. Each value in y'_f and y'_c is the position of floor-wall boundary and ceiling-wall boundary at the corresponding image column. y'_w represents the probability of wall-wall existence of each image column.

Recovering the Floor and Ceiling Planes: For each column of the image, we can use the corresponding values in y'_f, y'_c to vote for the ceiling-floor distance. Based on the assumed camera height, we can project the floor-wall boundary y'_f from image to 3D XYZ position (they all shared the same Y). The ceiling-wall boundary y'_c shares the same 3D X, Z position with the y'_f on the same image column, and therefore the distance between floor and ceiling can be calculated. We take the average of results calculated from all image columns as the final floor-ceiling distance.

다음의 4가지 가정 하에 cuboid에 한정되지 않는 room layout을 복구.

① 교차하는 벽은 서로 수직이다. (Manhattan world 가정)
② 모든 방은 서로 평행한 하나의 바닥과 하나의 천장을 가짐.
③ 카메라 높이는 1.6m
④ 전처리 단계는 바닥을 y축에 직교하도록 정렬.

3.1에 나타났듯이
딥러닝 모델의 출력은 각 이미지 열에 대한 layout information이 포함.

Yf : 바닥-벽 경계
Yc : 천장-벽 경계
Yw : 벽-벽 존재 확률

바닥과 천장 평면 복구

바닥-천장 거리에 대한 투표를 위해 Yf, Yc에 상응하는 값을 사용 가능.

① 카메라 높이에 기초하여, Yf(바닥-벽 경계)를 3D (X,Y,Z) 위치로 투영 가능. → 모두 같은 Y 공유함.

② Yc(천장-벽 경계)는 같은 이미지 열에서 Yf와 동일한 3D X,Z 위치를 공유.

따라서 바닥과 천장 거리 계산 가능.

우리는 모든 이미지의 열로부터 계산된 결과들의 평균을 바닥-벽 천장 거리로 계산.

Manhattan World 가정 : 영상에 나타나는 평면들은 3차원상에서 서로 직교하는 평면들 로만 이루어져 있다.

Post-processing

Recovering the Floor and Ceiling Planes

3.2 Post-processing

벽 평면 복구

Recovering Wall Planes: We first find the prominent peaks on the estimated wall-wall probability y'_w with two criteria: *i*) the signal should be larger than any other signal within 5°H-FOV, and *ii*) the signal should be larger than 0.05.

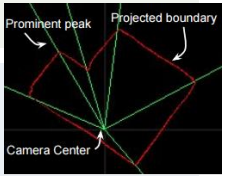
Fig. 4a shows the projected y'_c (red points) on ceiling plane. The green lines are the detected prominent peaks which split the ceiling-wall boundary (red points) into multiple parts. To handle possibly failed horizontal alignment in the pre-processing step, we calculate the first principal component of each part, then rotate the scene by the average angle of all first principal components (top right figure in Fig. 4a). So now we have two types of walls: *i*) X-axis orthogonal walls and *ii*) Z-axis orthogonal walls. We construct the walls from low to high variance suggested by the first principal component. Adjacency walls are forced to be orthogonal to each other, thus only walls whose two adjacent walls are not yet constructed have the freedom to decide the orthogonal type. We use a simple voting strategy: each projected red point votes for all planes within 0.16 meters (bottom right figure in Fig. 4a). The most voted plane is selected. Two special cases are depicted in Fig 4b which occur when the two adjacency walls are already constructed and they are orthogonal to each other. Finally, the XYZ positions of all corners are decided according to the intersection of three adjacent Manhattan junction planes.

The time complexity of our post-processing procedure is $\mathcal{O}(W)$, where W is the image width. Thus the post-processing can be efficiently done; in average, it takes less than 20ms to finish.

Yw(벽-벽의 존재 확률)에서 다음 두가지 기준을 가지고 눈에 띄는 봉우리를 찾음.

① 그 신호는 어떤 다른 신호보다 커야 함.

② 그 신호는 0.0보다 커야 함.



천장에서 Yc(천장-벽 경계, 빨간 선)를 보여줌.

녹색 선 : 천장-벽 경계(빨간 선)를 여러 부분으로 나누는 눈에 띄는 peak를 감지함.

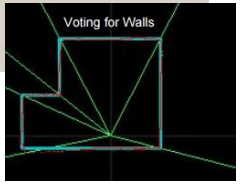
전처리 단계에서 실패할 수 있는 수평 정렬을 처리하기 위해 각 부분의 주 성분 계산함. 그리고 모든 첫번째 주성분의 평균 각에 의해 scene을 회전시킴.

그러면 우리는 다음 두가지 유형의 벽을 가지게 된다.

① X축 직교벽

② Z축 직교벽

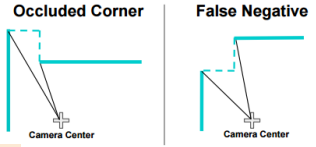
첫번째 주성분에 의해 제시된 낮은 variance에서 높은 variance까지 벽을 만듦.
인접한 벽은 서로 직교해야 함.
오직 두개의 인접한 벽을 가져야만 직교유형을 결정할 자유도를 가질 수 있음.



간단한 투표 전략을 사용함.
각 투영된 red point는 모든 평면들에 대해 투표함.
가장 많이 투표 받은 평면이 선택됨.

다음 두가지 특별한 경우 있음
두 직교하는 벽이 이미 만들어졌을 경우와 그들이 서로 직교할 경우

모든 코너의 XYZ 위치는 인접한 Manhattan 접속평면 3개의 교차점에 따라 결정됨



(b) Two special cases: Instead of voting for a wall, we add a corner according to the two prominent peaks and the positions of two walls.

벽에 투표하는 대신, 두개의 두드러진 봉우리와 두 개의 벽의 위치에 따라 코너를 추가함.

post-processing의 시간 복잡도 = O(W)
W 는 이미지 너비.
평균적으로 20ms 보다 덜 걸림.

Recovering Wall Planes

3.3

Panorama Stretch Data

4.1 Datasets

4. Experiments

4.1. Datasets

We train and evaluate our model using the same dataset as LayoutNet [36]. The dataset consists of PanoContext dataset [32] and the extended Stanford 2D-3D dataset [1] annotated by [36]. To train our model, we generate $3 \times 1 \times 1024$ ground truth from the annotation. We follow the same training/validation/test split of LayoutNet.

LayoutNet[36]과 동일한 데이터 세트를 사용하여 모델을 교육하고 평가한다.
데이터 세트는 [32] PanoContext 데이터 세트와 [36] 주석이 달린 확장된 Stanford 2D-3D 데이터 세트[1]로 구성된다.
모델을 훈련시키기 위해, 주석으로부터 3 * 1 * 1024개의 ground truth를 생성한다.
LayoutNet의 동일한 교육/검증/테스트 분할을 수행한다.

4.2. Training Details

The Adam optimizer [16] is employed to train the network for 300 epochs with batch size 24 and learning rate 0.0003. The L1 Loss is used for the ceiling-wall boundary (y_c) and floor-wall boundary (y_f). The Binary Cross-Entropy Loss is used for the wall-wall corner (y_w). The network is implemented in PyTorch [20]. It takes four hours to finish the training on three NVIDIA GTX 1080 Ti GPUs.

The data augmentation techniques we adopt include standard left-right flipping, panoramic horizontal rotation, and luminance change. Moreover, we exploit the proposed Pano Stretch Data Augmentation (Sec. 3.3) during training. The stretching factors k_x, k_z are sampled from uniform distribution $U[1, 2]$, and then take the reciprocals of sampled values with probability 0.5. The process time of Pano Stretch Data Augmentation is roughly 130ms per 512×1024 RGB image. Therefore, it is feasible to be applied on-the-fly during training.

L1 손실은 천장-벽 경계(yc) 및 바닥-벽 경계(yf)에 사용된다.
Binary CrossEntropy Loss는 벽면 코너(yw)에 사용된다.
네트워크는 PyTorch [20]에서 구현된다.
NVIDIA GTX 1080Ti GPU 3개에 대한 교육을 완료하는 데 4시간이 소요된다.
우리가 채택하는 데이터 확대 기법에는 표준 좌측-우측 플립, 파노라마 수평 회전 및 휘도 변화가 포함된다.
또한, 우리는 훈련 중에 제안된 Pano Stretch 데이터 확대(3.3절)를 활용한다.
스트레칭 인자 kx, kz는 균일 분포 U[1, 2]에서 샘플링된 다음 0.5 확률로 샘플링된 값의 역수를 구한다.
Pano Stretch Data Augment의 프로세스 시간은 512 x 1024 RGB 이미지당 약 130ms이다.
따라서 훈련 중에 즉시 적용하는 것이 가능하다.

4.2 Training Details

4.3 Cuboid Room Results

4.3. Cuboid Room Results

We generate cuboid room by only selecting the four most prominent peaks in the post-processing step (Sec. 3.2).

Quantitative Results: Our approach is evaluated on three standard metrics: *i*) **3D IoU**: intersection over union between 3D layout constructed from our prediction and the ground truth; *ii*) **Corner Error**: average Euclidean distance between predicted corners and ground-truth corners (normalized by image diagonal length); *iii*) **Pixel Error**: pixel-wise error between predicted surface classes and ground-truth surface classes.

The quantitative results of different training and testing settings are summarized in Table 1 and Table 2. To clarify the difference, the input resolution of DuLa-Net [30] and CFL [8] are 256×512 while LayoutNet [36] and ours are 512×1024. Other than conventional augmentation technique, CFL [8] is trained with Random Erasing while ours is trained with the proposed Pano Stretch. DuLa-Net [30] did not report corner errors and pixel errors. Our approach achieves state-of-the-art performance and outperforms existing methods under all settings.

Qualitative Results: The qualitative results are shown in Fig. 6. We present the results from the best to the worst based on their corner errors. Please see more results in the supplemental materials.

Computation time: The 1D layout representation is easy to compute. Forward passing a single *512 x 1024* RGB image takes 8ms and 50ms for our HorizonNet with and without RNN respectively. The post-processing step for extracting layout from our 1D representation takes only 12ms. We evaluate the result on a single NVIDIA Titan X GPU and an Intel i7-5820K 3.30GHz CPU. The reported execution time is averaged across all the testing data.

Post - processing 단계에서 가장 두드러진 네 개의 피크를 선택하여 cuboid room을 생성한다(3.2절).
정량적 결과 : 접근 방식은 다음 세 가지 표준 메트릭스에서 평가된다
① 3D IoU : 우리의 예측에서 생성된 3D 배치와 실측 진실 사이의 결합에 대한 교차점
② 코너 오류 : 예측 코너와 지면 실측 코너 사이의 평균 유클리드 거리(이미지 대각선 길이로 정규화)
③ 픽셀 오류: 예측 표면 클래스와 지면 진실 표면 클래스 사이의 픽셀 단위 오류.

다른 훈련 및 시험 설정의 정량적 결과는 표 1과 표 2에 요약되어 있음.
차이를 명확히 하기 위해,
DuLa-Net [30]과 CFL [8]의 입력 해상도는 256 * 512이고, LayoutNet [36]과 우리의 입력 해상도는 512 * 1024.
기존의 증강 기법 외에 CFL[8]은 무작위 소거로 훈련되고, 반면 우리는 제안된 Pano Stretch로 훈련한다.
DuLa-Net [30]은 코너 오류와 픽셀 오류를 보고하지 않았다.
우리의 접근 방식은 최첨단 성능을 달성하고 모든 설정에서 기존 방법을 능가한다.

정성적 결과는 그림 6에 나타나 있다. 코너 오류를 기준으로 최고에서 최악으로 결과를 제시한다.

계산 시간 : 1D 레이아웃 표현은 계산하기 쉽다.
단일 512 x 1024 RGB 이미지를 전달하려면 HorizonNet의 경우 RNN이 존재하면 8ms, 존재하지 않으면 50ms가 걸린다.
1D 표현에서 레이아웃을 추출하기 위한 후 처리 단계는 12ms에 불과하다.
단일 NVIDIA Titan X GPU와 Intel i7- 5820K 3.30GHz CPU에서 결과를 평가한다.
보고된 실행 시간은 모든 테스트 데이터에 걸쳐 평균화 된다.

RNN 존재하는 경우와 존재하지 않는 경우 직접 비교

4.4 Ablation Study

4.4. Ablation Study

Ablation experiments are presented in Table 3. We report the result averaged across all the testing instances. For a fair comparison, we also experiment with dense $\mathcal{O}(HW)$ prediction following LayoutNet [36] but replace the U-Net [24] with the same backbone as our architecture. [1] The results of this setting are presented in the first two rows. We do not try dense $\mathcal{O}(HW)$ output with RNN since it would consume too many computing resources. We can see that learning on our 1D $\mathcal{O}(W)$ layout representation is better than conventional dense $\mathcal{O}(HW)$ layout representation.

We observe that training with the proposed Pano Stretch Data Augmentation can always boost the performance. Note that the proposed data augmentation method can also be adopted in other tasks on panoramas and has the potential to increase their accuracy as well. See supplemental material for the experiment using Pano Stretch Data Augmentation on semantic segmentation task.

For the rows where RNN columns are unchecked, the RNN components shown in Fig 2 are replaced by fully connected layers. Our experiments show that using RNN in network architecture also improves performance. Fig. 7 shows some representative results with and without RNN. The raw output of the model with RNN is highly consistent with the Manhattan world even without post-processing, which demonstrates the ability of RNN to capture the geometric pattern of the entire room.

절제 연구

제안된 Pano Stretch Data Argumentation를 통한 훈련이 성능을 향상시킬 수 있다는 것이 관찰됨.
제안된 Data Augmentation은파노라마의 다른 작업에 채택되고 잠재적인 정확도를 높이기 위한 것.

RNN 열이 선택되지 않은 행의 경우, RNN의 구성 요소는 fully connected layers로 구성된다.

우리의 실험은 네트워크 아키텍처에서 RNN을 사용하는 것도 성능을 향상시킨다는 것을 보여준다.
그림 7은 RNN을 포함하거나 포함하지 않은 몇 가지 대표적인 결과를 보여준다.

RNN을 사용한 모델의 출력은 사후 처리가 없어도 Manhattan world와 매우 일치하며, 이는 RNN이 전체 room의 기하학적 패턴을 캡처할 수 있는 능력을 보여준다.

4.5 Non-cuboid Room Results

4.5. Non-cuboid Room Results

Since the non-cuboid rooms in PanoContext and Stanford 2D-3D dataset are labeled as cuboids, our model is never trained to recognize non-cuboid layouts and concave corners. This bias makes our model tend to predict complex-shaped rooms as cuboids. To estimate general room layouts, we re-label 65 rooms from the training split to fine-tune our trained model. We fine-tune our model for 300 epochs with learning rate 5e−5 and batch size 2.

To quantitatively evaluate the fine-tuning result on general-shaped rooms, we use 13-fold cross validation on the 65 re-annotated non-cuboid data. The results are summarized in Table 4. We depict some examples of reconstructed non-cuboid layouts from the testing and validation splits in Fig.1 and Fig.8. See supplemental material for more reconstructed layouts. The results show that our approach can work well on general room layout even with corners occluded by other walls.

PanoContext 및 Stanford 2D-3D 데이터 세트의 non-cuboid room은 cuboid로 labeled되므로, 우리의 모델은 non-cuboid 레이아웃을 인식하고 모서리를 오목하도록 결코 훈련되지 않는다.

이 편향은 우리의 모델이 복잡한 모양의 방을 cuboid로 예측하는 경향이 있게 만든다.
일반적인 룸 레이아웃을 추정하기 위해, 우리는 훈련 분할에서 65개의 룸에 레이블을 다시 지정하여 훈련된 모델을 미세 조정한다.

우리는 5e-5 learning rate와 batch size 2를 가지고 300 epochs로 모델을 미세 조정한다.

일반 모양의 룸에 대한 미세 조정 결과를 정량적으로 평가하기 위해 65개의 다시 주석 처리된 non-cuboid 데이터에 대해 13배 교차 검증을 사용한다.
결과는 표 4에 나타난다.

Fig 1과 Fig 8의 테스트 및 검증 분할에서 재구성된 non-cuboid 레이아웃의 예를 설명한다.

결과는 우리의 접근 방식이 다른 벽으로 가려진 모서리를 사용하더라도 일반적인 방 배치에서 잘 작동할 수 있다는 것을 보여준다.

Non-cuboid 보완해보기

5

Conclusions

5. Conclusion

We have presented a new 1D representation for the task of estimating room layout from a panorama. The proposed HorizonNet trained with such 1D representation outperforms previous state-of-the-art methods and requires fewer computation resources. Our post-processing method which recovers 3D layout from the model output is fast and effective, and it also works for complex room layouts even with occluded corners. The proposed Pano Stretch Data Augmentation further improves our results, and can also be applied to the training procedure of other panorama tasks for potential improvement.

우리는 파노라마에서 룸 레이아웃을 추정하는 작업을 위한 새로운 1D 표현을 제시했다.
그러한 1D 표현을 사용하여 훈련된 제안된 HorizonNet은 이전의 최첨단 방법을 능가하고 더 적은 계산 리소스를 필요로 한다.
모델 출력에서 3D 레이아웃을 복구하는 우리의 후처리 방법은 빠르고 효과적이며, 막힌 모서리가 있더라도 복잡한 룸 레이아웃에 효과적이다.
제안된 Pano Stretch 데이터 확대는 우리의 결과를 더욱 향상시키며, 잠재적 개선을 위한 다른 파노라마 작업의 훈련 절차에도 적용할 수 있다.

감사합니다.

The End