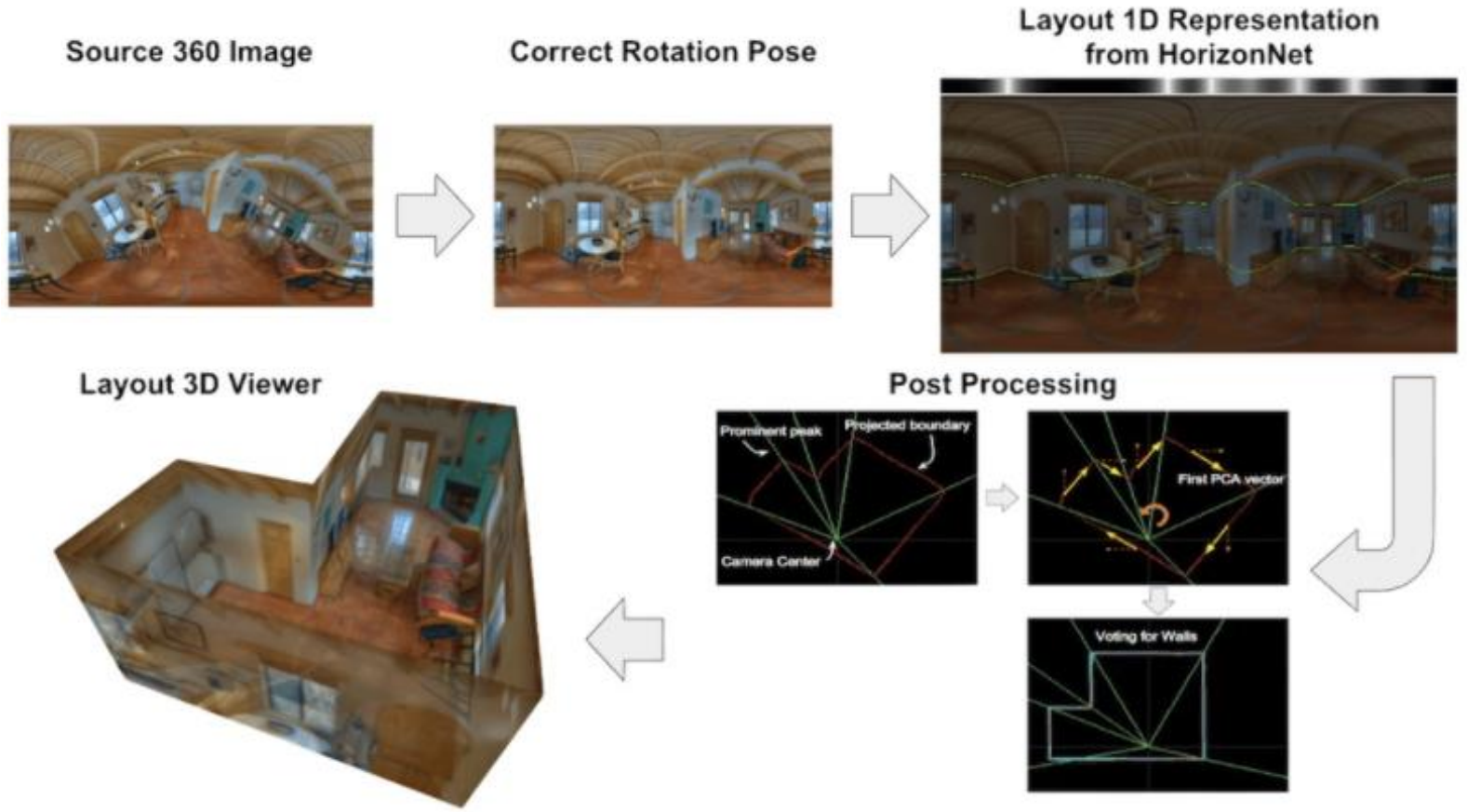
3D Room Layout Estimation

AI 융합학부 길다영

10.18

01 Horizonnet

코드가 제일 잘 돌아가고 제일 많이 이것저것 실험해 봤던 HorizonNet을 중점으로 하기로 함.



01 Network 적용 방법 변경

Commit history에서 network(resnet50, LSTM)을 다른 방법으로 적용한 것을 발견하여 두 코드를 비교해 봄.

변경

```
- model_urls = {
-     'resnet18': 'https://download.pytorch.org/models/resnet18-5c106cde.pth',
-     'resnet50': 'https://download.pytorch.org/models/resnet50-19c8e357.pth',
-     'resnet101': 'https://download.pytorch.org/models/resnet101-5d3b4d8f.pth',
- }
```

```
  def lr_pad(x, padding=1):
      return torch.cat([x[..., -padding:], x, x[..., :padding]], dim=3)

  class LR_PAD(nn.Module):
      def __init__(self, padding=1):
          super(LR_PAD, self).__init__()
          self.padding = padding
@@ -26,185 +30,93 @@ def forward(self, x):
          return lr_pad(x, self.padding)
```

```
- def conv3x3(in_planes, out_planes, stride=1):
-     """3x3 convolution with padding"""
-     return nn.Conv2d(in_planes, out_planes, kernel_size=3, stride=stride,
-                      padding=(1, 0), bias=False)
-
-
- class BasicBlock(nn.Module):
-     expansion = 1
-
-     def __init__(self, inplanes, planes, stride=1, downsample=None):
-         super(BasicBlock, self).__init__()
-         self.conv1 = conv3x3(inplanes, planes, stride)
-         self.bn1 = nn.BatchNorm2d(planes)
-         self.relu = nn.ReLU(inplace=True)
-         self.conv2 = conv3x3(planes, planes)
-         self.bn2 = nn.BatchNorm2d(planes)
-         self.downsample = downsample
-         self.stride = stride

      def forward(self, x):
-         residual = x
-
-         out = self.conv1(lr_pad(x, 1))
-         out = self.bn1(out)
-         out = self.relu(out)
-
-         out = self.conv2(lr_pad(out, 1))
-         out = self.bn2(out)
-
-         if self.downsample is not None:
-             residual = self.downsample(x)
-
-         out += residual
-         out = self.relu(out)
-
-         return out
```

기존

```
+ ENCODER_RESNET = [
+     'resnet18', 'resnet34', 'resnet50', 'resnet101', 'resnet152',
+     'resnext50_32x4d', 'resnext101_32x8d'
+ ]
+ ENCODER_DENSENET = [
+     'densenet121', 'densenet169', 'densenet161', 'densenet201'
+ ]


  def lr_pad(x, padding=1):
+     ''' Pad left/right-most to each other instead of zero padding '''
      return torch.cat([x[..., -padding:], x, x[..., :padding]], dim=3)


  class LR_PAD(nn.Module):
+     ''' Pad left/right-most to each other instead of zero padding '''
      def __init__(self, padding=1):
          super(LR_PAD, self).__init__()
          self.padding = padding

          return lr_pad(x, self.padding)


+ '''
+ Encoder
+ '''
+ class Resnet(nn.Module):
+     def __init__(self, backbone='resnet50', pretrained=True):
+         super(Resnet, self).__init__()
+         assert backbone in ENCODER_RESNET
+         self.encoder = getattr(models, backbone)(pretrained=pretrained)
+         del self.encoder.fc, self.encoder.avgpool

      def forward(self, x):
+         features = []
+         x = self.encoder.conv1(x)
+         x = self.encoder.bn1(x)
+         x = self.encoder.relu(x)
+         x = self.encoder.maxpool(x)
+
+         x = self.encoder.layer1(x); features.append(x)  # 1/4
+         x = self.encoder.layer2(x); features.append(x)  # 1/8
+         x = self.encoder.layer3(x); features.append(x)  # 1/16
+         x = self.encoder.layer4(x); features.append(x)  # 1/32
+         return features
+
+     def list_blocks(self):
+         lst = [m for m in self.encoder.children()]
+         block0 = lst[:4]
+         block1 = lst[4:5]
```

https://github.com/sunset1995/HorizonNet/commit/edde54ac6aa640b7d6b631784d51841066cfcfc3#diff-fada037ad086638e65c7ae77e3d223963e9afaa26326aab0ea718f4013176e43

01 Network 적용 방법 변경
– 변경된 네트워크와 기존 네트워크 비교

- 변경한 코드와 기존 코드를 가지고 각각 모델을 훈련시킴.
  - train : 80, valid : 20, test : 20
  - resnet50, epoch300
  - Test에는 리코세타 카메라로 찍은 이미지를 포함시킴.

- Train.py에 훈련 시 시간 측정 코드를 도입하여 비교해 봄.
  - 변경된 코드가 기존 코드보다 약 5분 정도 빨랐음.
  - 데이터 셋 개수를 늘리면 그 차이는 더 커질 것이라 예상.

Train.py

```
    terminate_time = timeit.default_timer()  # 종료 시간 체크
    tottime += terminate_time - start_time
    print("%f초 걸렸습니다. -> 총 %f초" % (terminate_time - start_time,tottime))

    # Periodically save model
    if ith_epoch % args.save_every == 0:
        save_model(net,
                   os.path.join(args.ckpt, args.id, 'epoch_%d.pth' % ith_epoch),
                   ith_epoch)
        avgtime = tottime/ith_epoch
        print("평균 %f초 걸렸습니다"% (avgtime))
```

기존

```
Train ep295: 100%| 80/80 [00:23<00:00,  3.46it/s]
Valid ep295: 100%| 20/20 [00:02<00:00,  7.19it/s]
Ep295 0.4255 vs. Best 0.5576
9066.852147초 걸렸습니다. -> 총 1355705.290019초
Train ep296: 100%| 80/80 [00:22<00:00,  3.52it/s]
Valid ep296: 100%| 20/20 [00:02<00:00,  7.11it/s]
Ep296 0.4277 vs. Best 0.5576
9092.406172초 걸렸습니다. -> 총 1364797.696191초
Train ep297: 100%| 80/80 [00:22<00:00,  3.53it/s]
Valid ep297: 100%| 20/20 [00:02<00:00,  7.19it/s]
Ep297 0.4234 vs. Best 0.5576
9117.883991초 걸렸습니다. -> 총 1373915.580182초
Train ep298: 100%| 80/80 [00:22<00:00,  3.49it/s]
Valid ep298: 100%| 20/20 [00:02<00:00,  7.19it/s]
Ep298 0.4484 vs. Best 0.5576
9143.605899초 걸렸습니다. -> 총 1383059.186081초
Train ep299: 100%| 80/80 [00:22<00:00,  3.51it/s]
Valid ep299: 100%| 20/20 [00:02<00:00,  7.11it/s]
Ep299 0.4418 vs. Best 0.5576
9169.211614초 걸렸습니다. -> 총 1392228.397695초
Train ep300: 100%| 80/80 [00:22<00:00,  3.53it/s]
Valid ep300: 100%| 20/20 [00:02<00:00,  7.07it/s]
Ep300 0.4283 vs. Best 0.5576
9194.700583초 걸렸습니다. -> 총 1401423.098278초
평균 4671.410328초 걸렸습니다
Epoch: 100%| 300/300 [2:33:11<00:00, 30.64s/ep]
```

훈련 시, 평균 4671.410328초 걸림. **약 1시간 18분.**

훈련 시, 평균 4348.282792초 걸림. **약 1시간 13분.**

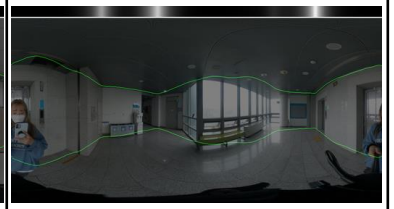
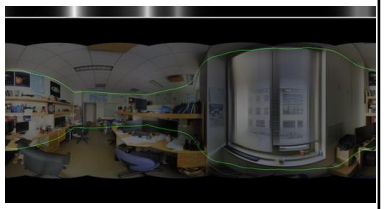
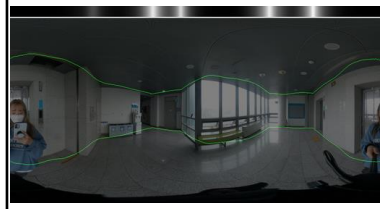
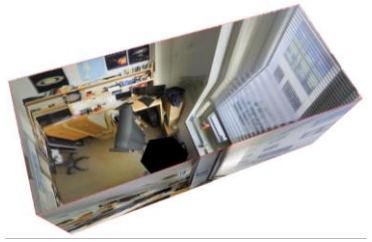
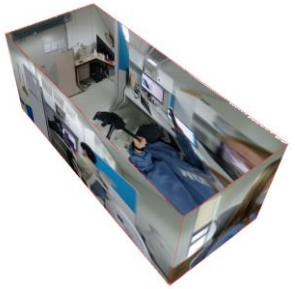
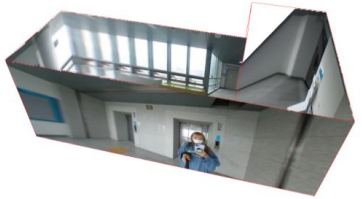
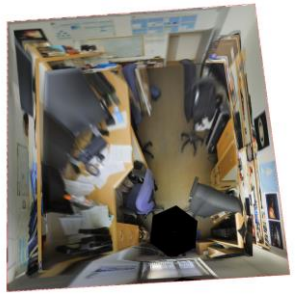
변경

```
Train ep295: 100%| 80/80 [00:27<00:00,  2.95it/s]
Valid ep295: 100%| 10/10 [00:01<00:00,  6.60it/s]
8517.148627초 걸렸습니다. -> 총 1261467.931235초
Train ep296: 100%| 80/80 [00:27<00:00,  2.92it/s]
Valid ep296: 100%| 10/10 [00:01<00:00,  7.27it/s]
8545.923968초 걸렸습니다. -> 총 1270013.855203초
Train ep297: 100%| 80/80 [00:27<00:00,  2.91it/s]
Valid ep297: 100%| 10/10 [00:01<00:00,  7.53it/s]
8574.780311초 걸렸습니다. -> 총 1278588.635514초
Train ep298: 100%| 80/80 [00:27<00:00,  2.93it/s]
Valid ep298: 100%| 10/10 [00:01<00:00,  7.27it/s]
8603.428710초 걸렸습니다. -> 총 1287192.064224초
Train ep299: 100%| 80/80 [00:27<00:00,  2.93it/s]
Valid ep299: 100%| 10/10 [00:01<00:00,  7.27it/s]
8632.062247초 걸렸습니다. -> 총 1295824.126471초
Train ep300: 100%| 80/80 [00:27<00:00,  2.93it/s]
Valid ep300: 100%| 10/10 [00:01<00:00,  7.27it/s]
8660.711133초 걸렸습니다. -> 총 1304484.837604초
평균 4348.282792초 걸렸습니다 | 10/10 [00:01<00:00,  7.56it/s]
Epoch: 100%| 300/300 [2:24:16<00:00, 28.86s/ep]
```

01 Network 적용 방법 변경

- 변경된 네트워크와 기존 네트워크 비교

성능은 크게 차이 나지 않음 (제일 차이나는 사진으로만 간추려놓음).

	기존			변경		
입력 이미지						
1d Representation						
결과						

02 전처리 결과 좌우 반전

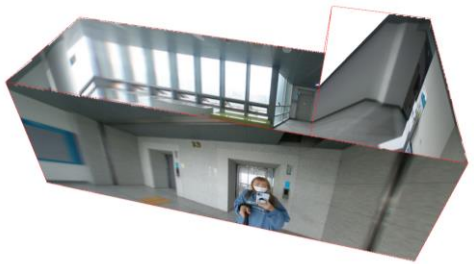
```
python preprocess.py --img_glob mydataset/test/*.jpg --output_dir mydataset/test/preprocessed/ --rev 1.0
```

리코세타 카메라로 찍은 이미지를 적용해본 결과, 좌우 반전을 해야 올바른 3d layout estimation이 도출된다는 것을 발견

- preprocess.py 에 사진 좌우 반전하는 코드를 도입하여 전처리 결과, 좌우 반전되도록 함.
- --rev라는 새로운 arguments 생성

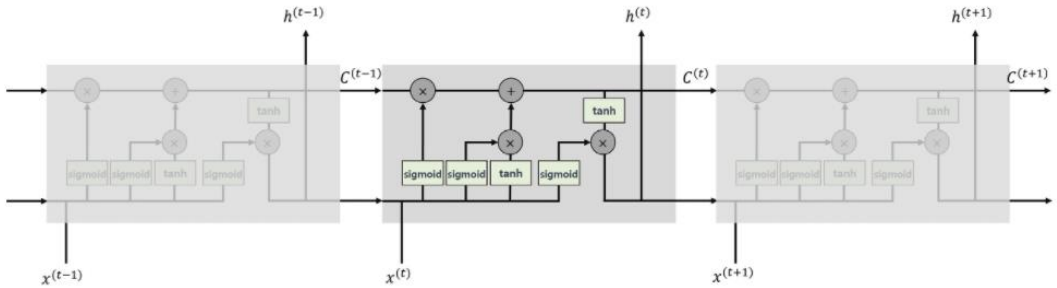
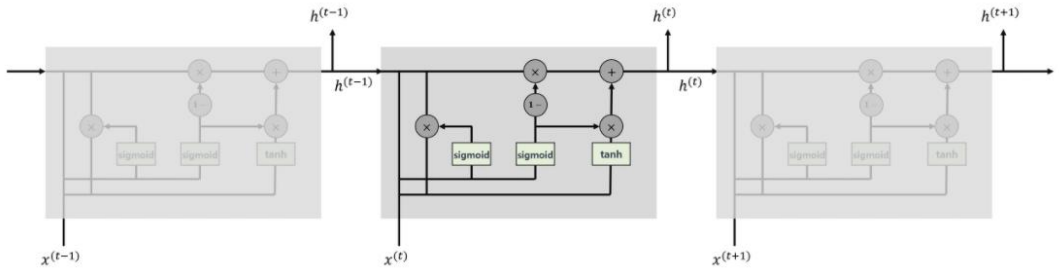
```
37      parser.add_argument('--rev', default=0, help="img left <-> right")
38      args = parser.parse_args()
```

```
70      # 사진 좌우 반전
71      rev = 0
72      if args.rev:
73          i_img = Image.fromarray((i_img * 255).astype(np.uint8)).transpose(Image.FLIP_LEFT_RIGHT)
74          rev = 1
75
76      # Dump results
77      basename = os.path.splitext(os.path.basename(i_path))[0]
78      if args.rgbonly:
79          path = os.path.join(args.output_dir, '%s.png' % basename)
80          if rev == 1:
81              i_img.save(path)
82          else:
83              Image.fromarray((i_img * 255).astype(np.uint8)).save(path)
84      else:
85          path_VP = os.path.join(args.output_dir, '%s_VP.txt' % basename)
86          path_i_img = os.path.join(args.output_dir, '%s_aligned_rgb.png' % basename)
87          path_l_img = os.path.join(args.output_dir, '%s_aligned_line.png' % basename)
88
89          with open(path_VP, 'w') as f:
90              for i in range(3):
91                  f.write('%.6f %.6f %.6f\n' % (vp[i, 0], vp[i, 1], vp[i, 2]))
92
93          if rev == 1:
94              i_img.save(path_i_img)
95          else:
96              Image.fromarray((i_img * 255).astype(np.uint8)).save(path_i_img)
97
98          Image.fromarray((l_img * 255).astype(np.uint8)).save(path_l_img)
```

	기존	좌우반전
입력 이미지		
전처리 결과		
최종 결과		

03 LSTM -> GRU로 변경

기존의 LSTM보다 단순한 구조를 가지는 GRU구조로 네트워크를 바꾸고 훈련 시켜봄.

LSTM	
GRU	 Figure 1: GRU, 개요

model.py

```
241                 GlobalHeightConv(512 * _exp, int(512 * _exp / _out_scale)),
242             ])
243             self.step_cols = 4
244             self.rnn_hidden_size = 512
245             if self.use_rnn:
246                 #self.bi_rnn = nn.LSTM(input_size=_exp * 256,
247                 #                       hidden_size=self.rnn_hidden_size,
248                 #                       num_layers=2,
249                 #                       dropout=0.5,
250                 #                       batch_first=False,
251                 #                       bidirectional=True)
252                 self.bi_rnn = nn.GRU(input_size=_exp * 256,
253                                      hidden_size=self.rnn_hidden_size,
254                                      num_layers=2,
255                                      dropout=0.5,
256                                      batch_first=False,
257                                      bidirectional=True)
258                 self.drop_out = nn.Dropout(0.5)
259                 self.linear = nn.Linear(in_features=2 * self.rnn_hidden_size,
260                                         out_features=3 * self.step_cols)
261                 self.linear.bias.data[0::4].fill_(-1)
262                 self.linear.bias.data[4::8].fill_(-0.478)
263                 self.linear.bias.data[8::12].fill_(0.425)
264             else:
265                 self.linear = nn.Sequential(
266                     nn.Linear(_exp * 256, self.rnn_hidden_size),
267                     nn.ReLU(inplace=True),
268                     nn.Dropout(0.5),
```

03 LSTM -> GRU로 변경

LSTM과 GRU의 파라미터 수 비교

- GRU 수가 더 적음.

model.py

```python
model = HorizonNet(backbone='resnet50', use_rnn=True)
# (batch, channels, height, width)

if torch.cuda.is_available():
    model.cuda()

summary(model, (1, 3, 512, 1024))
```

```
    └─Bottleneck: 3-15                  [1, 2048, 16, 32]         4,462,592
    └─Bottleneck: 3-16                  [1, 2048, 16, 32]         4,462,592
├─ModuleList: 1-1                       --                        --
    └─GlobalHeightConv: 2-9             [1, 32, 8, 256]           --
    │   └─Sequential: 3-17              [1, 32, 8, 256]           535,584
    └─GlobalHeightConv: 2-10            [1, 64, 4, 256]           --
    │   └─Sequential: 3-18              [1, 64, 4, 128]           2,140,224
    └─GlobalHeightConv: 2-11            [1, 128, 2, 256]          --
    │   └─Sequential: 3-19              [1, 128, 2, 64]           8,556,672
    └─GlobalHeightConv: 2-12            [1, 256, 1, 256]          --
    │   └─Sequential: 3-20              [1, 256, 1, 32]           34,218,240
├─GRU: 1-3                              [256, 1, 1024]            9,449,472
├─Dropout: 1-4                          [256, 1, 1024]            --
├─Linear: 1-5                           [256, 1, 12]              12,300
==========================================================================================
Total params: 78,420,524
Trainable params: 78,420,524
Non-trainable params: 0
Total mult-adds (G): 70.65
==========================================================================================
Input size (MB): 6.29
Forward/backward pass size (MB): 1964.40
Params size (MB): 313.68
Estimated Total Size (MB): 2284.37
==========================================================================================

Process finished with exit code 0
```

GRU

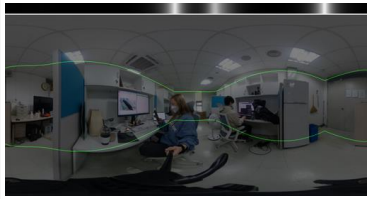
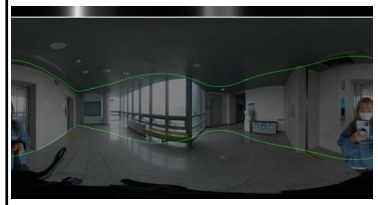
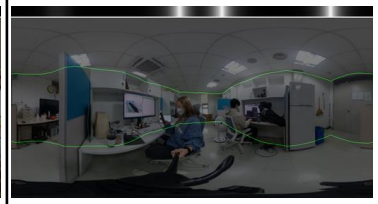
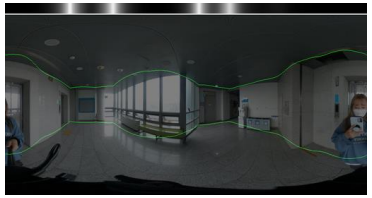
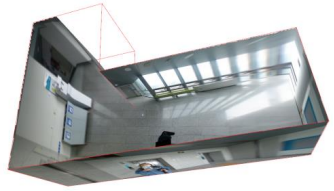
```
├─ModuleList: 1-1                       --                        --
    └─GlobalHeightConv: 2-9             [1, 32, 8, 256]           --
    │   └─Sequential: 3-17              [1, 32, 8, 256]           535,584
    └─GlobalHeightConv: 2-10            [1, 64, 4, 256]           --
    │   └─Sequential: 3-18              [1, 64, 4, 128]           2,140,224
    └─GlobalHeightConv: 2-11            [1, 128, 2, 256]          --
    │   └─Sequential: 3-19              [1, 128, 2, 64]           8,556,672
    └─GlobalHeightConv: 2-12            [1, 256, 1, 256]          --
    │   └─Sequential: 3-20              [1, 256, 1, 32]           34,218,240
├─LSTM: 1-3                             [256, 1, 1024]            12,599,296
├─Dropout: 1-4                          [256, 1, 1024]            --
├─Linear: 1-5                           [256, 1, 12]              12,300
==========================================================================================
Total params: 81,570,348
Trainable params: 81,570,348
Non-trainable params: 0
Total mult-adds (G): 71.46
==========================================================================================
Input size (MB): 6.29
Forward/backward pass size (MB): 1964.40
Params size (MB): 326.28
Estimated Total Size (MB): 2296.97
==========================================================================================

Process finished with exit code 0
```

LSTM

03 LSTM -> GRU로 변경

LSTM과 GRU로 훈련된 모델로 3D Room layout estimation 결과, 성능에서 큰 차이를 보이지 않았음.

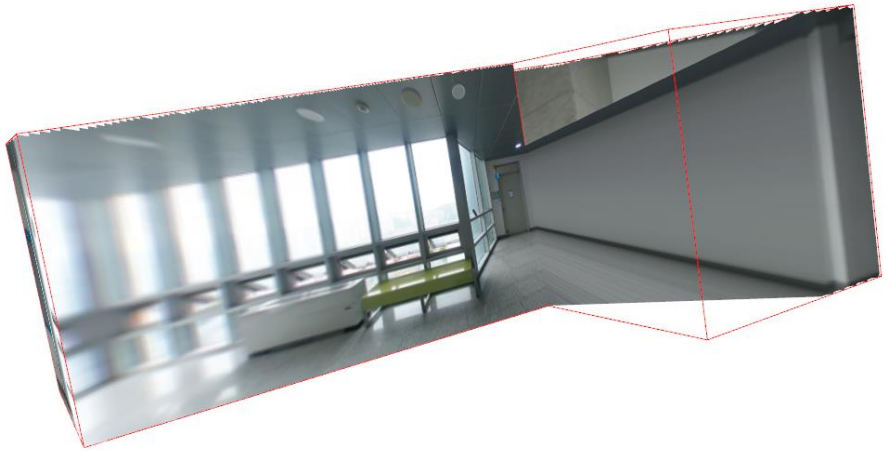
	GRU			LSTM		
입력 이미지						
1d Representation						
결과						

04 Layout_viewer 변경

horizonnet 내에서 point 로 결과가 나타나도록 할 수 있음

그러나, 현재 layout이 더 보기 편하므로 코드에 적용하지는 않음.

point

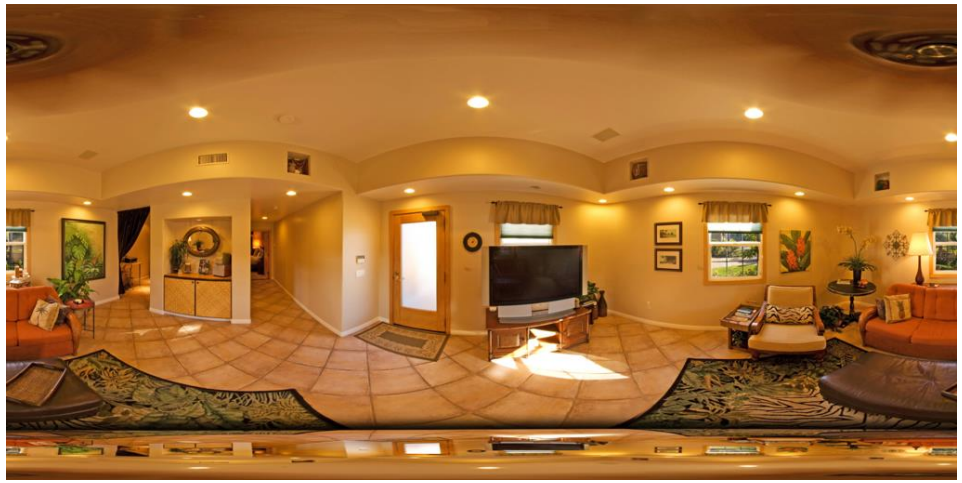


현재

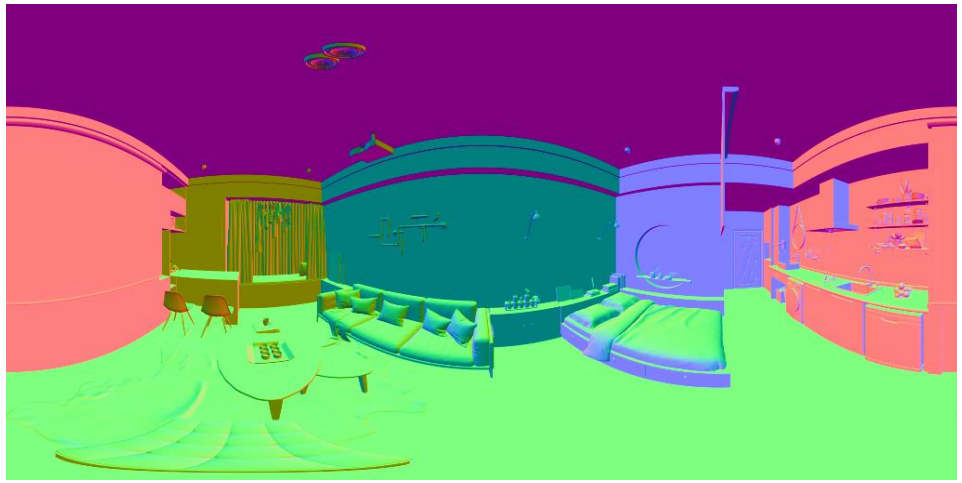
https://github.com/sunset1995/HorizonNet/commit/8d1aaded6144b755aa63634251ab54a647c4ac2f

05 지금 하고 있는 것 – 데이터셋 유형 바꾸어 훈련시키는 중임.

1



2



기존에는 1번 이미지로만 train set에 80개를 넣어 모델을 훈련시켰지만,

현재는 1번 이미지 40개, 2번 이미지 40개 넣어 훈련시키는 중이다.

END