Computer Vision
Term Project Presentation

Seongje Woo

School of integrated technology
Program of Intelligent Robot Technology
Gwangju Institute of Science and Technology (GIST)

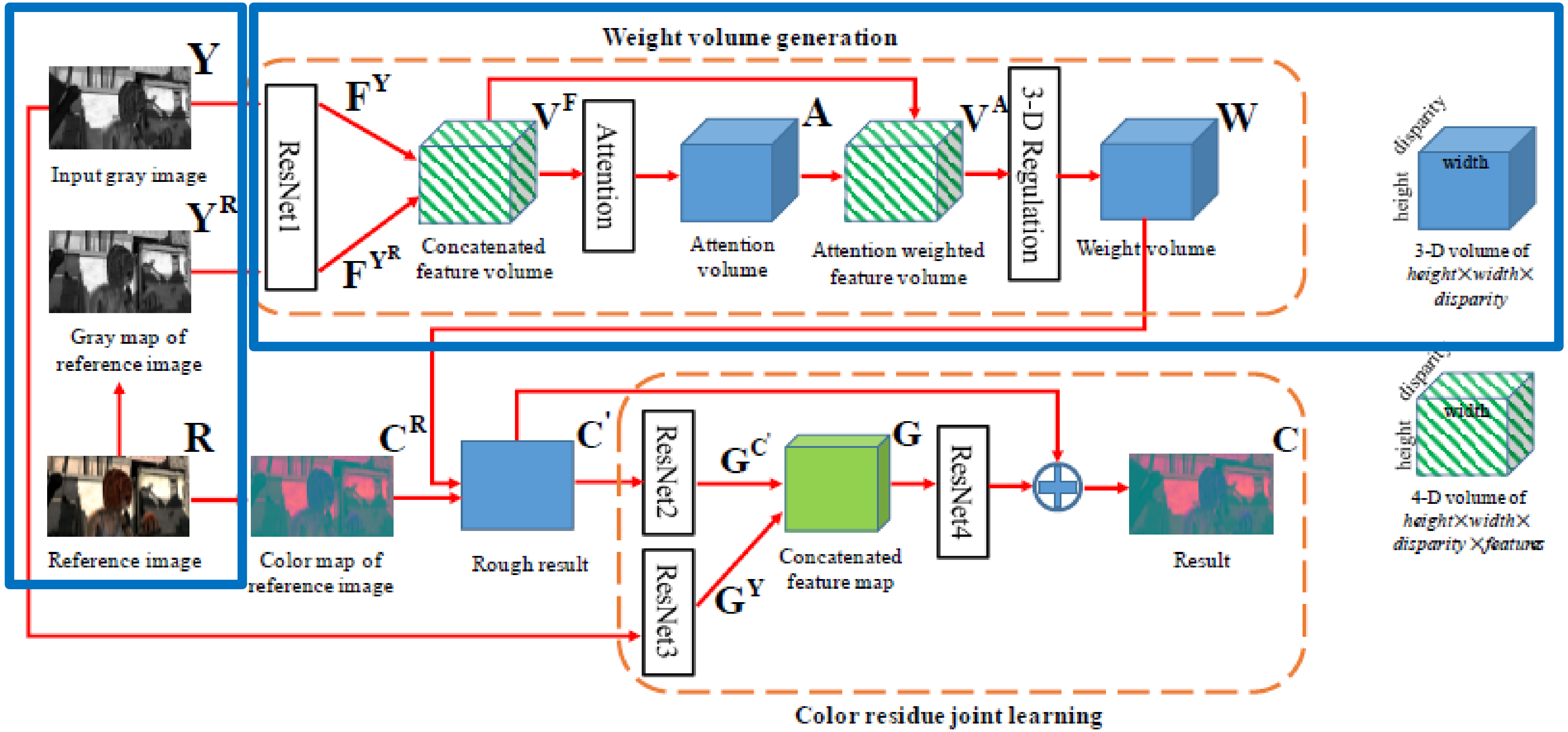
세계초일류
이공계대학

GIST
Gwangju Institute of Science and Technology

Contents

Introduction

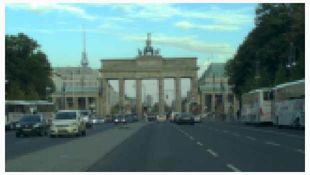
- **Learning a Deep Convolutional Network for Colorization in Monochrome-Color Dual-Lens System**(Xuan Dong et, al. Association for the Advancement of Artificial Intelligence 2019)



Data

- **./dataset.py**
- **Cityscapes** dataset for training
- **Left images** for **Input gray images** and **ground truth**
- **Right images** for **gray map** of reference images
- Add **Gaussian noises**

noise std.	color camera	monochrome camera
Setup1	$0.03\sqrt{\kappa}$	$0.01\sqrt{\kappa}$
Setup2	$0.07\sqrt{\kappa}$	$0.01\sqrt{\kappa}$



leftImg8bit_trainextra.zip (44GB) [md5]

left 8-bit images - trainextra set (19998 images, note that the image "troisdorf_000000_000073_leftImg8bit.png" is corrupt/black)

rightImg8bit_trainextra.zip (44GB) [md5]

right 8-bit images - trainextra set (19998 images)

Model

- **./models.py**
- Define networks(**ResNet1-4, Attention, 3D-regulation**)
- Implement model structure for **Weight volume generation**

```python
class WeightVolGen(nn.Module):
    def __init__(self):
        super(WeightVolGen, self).__init__()
        self.block = ResidualBlock
        # resnet1 for Y(gray image)
        self.resnet1_y = ResNet(8, self.block, 1)
        # resnet1 for YR(ref image)
        self.resnet1_yr = ResNet(8, self.block, 1)
        # attention for Concatenated feature volume
        self.attention = Attention()
        # 3-D Regulation
        self.regulation = Regulation3d(1)

    def forward(self, target_image, guide_image, gt_image):
        feature_map1 = self.resnet1_y(target_image)
        feature_map2 = self.resnet1_y(guide_image)
        attn_weighted_fvol = self.attention(torch.stack([feature_map1, feature_map2], dim=2))
        weight_volume = self.regulation(attn_weighted_fvol)
        return weight_volume
```

Table 1: Summary of our deep colorization architecture. Each 2-D or 3-D convolutional layer represents a block of convolution, batch normalization and ReLu.

	Layer Description	Output Tensor Dim.
	Input gray image $\mathbf{Y}$	$h \times w$
	Gray map of reference Image $\mathbf{Y}^{\mathbf{R}}$	$h \times w$
	ResNet1	
1	5×5 conv, n feat., stride 2	$\frac{h}{2} \times \frac{w}{2} \times n$
2	3×3 conv, n feat.	$\frac{h}{2} \times \frac{w}{2} \times n$
3	3×3 conv, n feat.	$\frac{h}{2} \times \frac{w}{2} \times n$
	add layer 1 and 3 feat. (residue connection)	$\frac{h}{2} \times \frac{w}{2} \times n$
4-17	(repeat layers 2,3 and residual connection)$\times$7	$\frac{h}{2} \times \frac{w}{2} \times n$
18	3×3 conv, n feat., no ReLu/BN	$\frac{h}{2} \times \frac{w}{2} \times n$
	Attention	
19	3-D conv,$1 \times 1 \times 1$,n feat.,Sigmoid,no BN/ReLu	$\frac{h}{2} \times \frac{w}{2} \times \frac{d}{2} \times n$
20	3-D conv,$1 \times 1 \times 1$,1 feat.,Sigmoid,no BN/ReLu	$\frac{h}{2} \times \frac{w}{2} \times \frac{d}{2}$
	3-D regulation	
21	3-D conv, $3 \times 3 \times 3$, n feat.	$\frac{h}{2} \times \frac{w}{2} \times \frac{d}{2} \times n$
22	3-D conv, $3 \times 3 \times 3$, n feat.	$\frac{h}{2} \times \frac{w}{2} \times \frac{d}{2} \times n$
23	3-D conv, $3 \times 3 \times 3$, $2n$ feat., stride 2	$\frac{h}{4} \times \frac{w}{4} \times \frac{d}{4} \times 2n$
24	3-D conv, $3 \times 3 \times 3$, $2n$ feat.	$\frac{h}{4} \times \frac{w}{4} \times \frac{d}{4} \times 2n$
25	3-D conv, $3 \times 3 \times 3$, $2n$ feat.	$\frac{h}{4} \times \frac{w}{4} \times \frac{d}{4} \times 2n$
26-34	(repeat layer 23, 24, 25)$\times$3	$\frac{h}{32} \times \frac{w}{32} \times \frac{d}{32} \times 2n$
35	$3 \times 3 \times 3$, 3-D trans conv, $2n$ feat., stride 2	$\frac{h}{16} \times \frac{w}{16} \times \frac{d}{16} \times 2n$
	add layer 35 and 31 (residual connection)	$\frac{h}{16} \times \frac{w}{16} \times \frac{d}{16} \times 2n$
36	$3 \times 3 \times 3$, 3-D trans conv, $2n$ feat., stride 2	$\frac{h}{8} \times \frac{w}{8} \times \frac{d}{8} \times 2n$
	add layer 36 and 28 (residual connection)	$\frac{h}{8} \times \frac{w}{8} \times \frac{d}{8} \times 2n$
37	$3 \times 3 \times 3$, 3-D trans conv, $2n$ feat., stride 2	$\frac{h}{4} \times \frac{w}{4} \times \frac{d}{4} \times 2n$
	add layer 37 and 25 (residual connection)	$\frac{h}{4} \times \frac{w}{4} \times \frac{d}{4} \times 2n$
38	$3 \times 3 \times 3$, 3-D trans conv, n feat., stride 2	$\frac{h}{2} \times \frac{w}{2} \times \frac{d}{2} \times n$
	add layer 38 and 22 (residual connection)	$\frac{h}{2} \times \frac{w}{2} \times \frac{d}{2} \times n$
39	$3 \times 3 \times 3$, 3-D trans conv, 1 feat., no ReLu/BN	$h \times w \times d$
	ResNet2	
40	5×5 conv, n feat.	$h \times w \times n$
41-57	repeat layers 2-18	$h \times w \times n$
	ResNet3	
58-75	repeat layers 40-57	$h \times w \times n$
	ResNet4	
76-92	repeat layers 40-56	$h \times w \times n$
93	3×3 conv, 1 feat. (no ReLu, BN)	$h \times w$

Train

- Implemented with PyTorch
- Model optimizer : RMSProp
- Batch size : 256 x 512 randomly located crop
- Learning late : 0.001
- Loss : Mean Squared Error

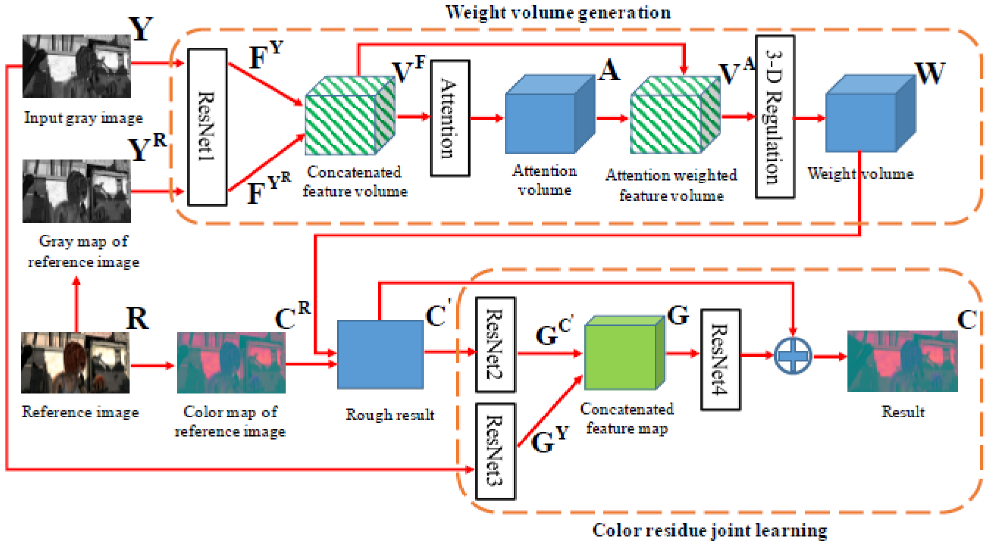
```
criterion = nn.MSELoss()
optimizer = torch.optim.RMSprop(model.parameters(), lr=lr, weight_decay=0, momentum=0)
psnr = PSNR(255.0).to(device)

for epochs in range(epochs):
    train_bar = tqdm(trainloader)

    for train_iter, items in enumerate(train_bar):
        model.train()

        # train 1
        target = Variable(items[0]).to(device)
        guide = Variable(items[1]).to(device)
        gt = Variable(items[2]).to(device)

        output = model(target, guide, gt)
        print('result of Weight volume generation : {}'.format(output.shape))
```



$$C'_{j,i} = \sum_{k=0}^{d-1} W_{j,i,k} C^R_{j,i+k}.$$

Summary

- Stereo datasets preprocessing
- Implement each networks structure
- Define model
- Extract concatenated feature map
- Extract attention volume
- Extract weight volume
- Rough colorization
- Color residue joint learning

Thank you for your Attention.