Multi-camera feature learning for post-capture white balance

Prerequisite

1. Python 3.6
2. pytorch (tested with 1.2.0 and 1.5.0)
3. torchvision (tested with 0.4.0 and 0.6.0)
4. cudatoolkit
5. tensorboard (optional)
6. numpy
7. Pillow
8. future
9. tqdm
10. matplotlib
11. scipy
12. scikit-learn

The code may work with library versions other than the specified.

Get Started

Demos:

1. Run `demo_single_image.py` to process a single image.
   Example of applying AWB + different WB settings:
   `python demo_single_image.py --input_image ../example_images/00.jpg --output_image ../result_images --show` . This example should save the output image in `../result_images` and output the following figure:

2. Run `demo_images.py` to process image directory. Example:
   `python demo_images.py --input_dir ../example_images/ --output_image ../result_images --task AWB` . The available tasks are AWB, all, and editing. You can also specify the task in the `demo_single_image.py` demo.

Training Code:

Run `training.py` to start training. You should adjust training image directories before running the code.

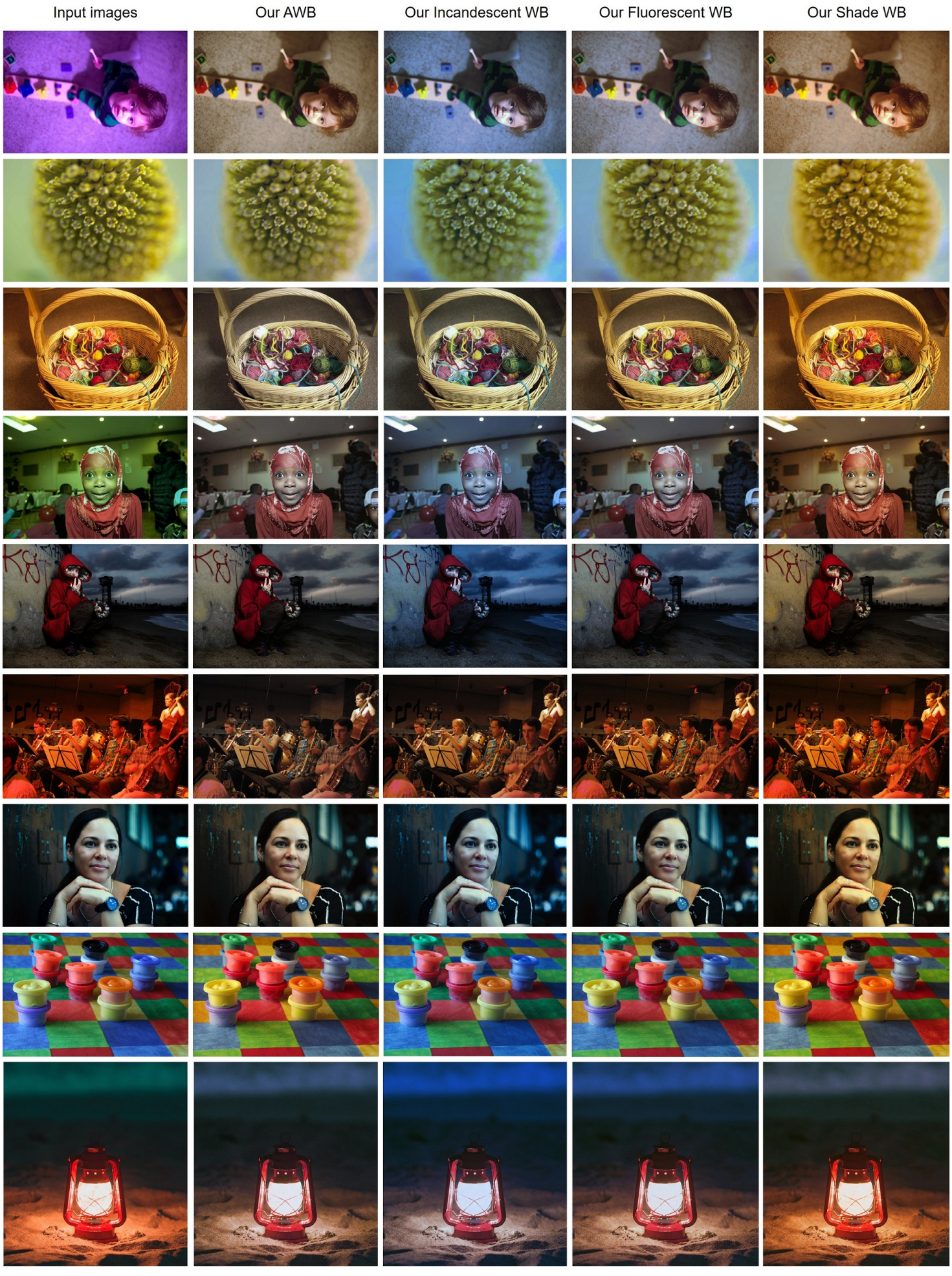
Example:
`CUDA_VISIBLE_DEVICE=0 python train.py --training_dir ../dataset/ --fold 0 --epochs 500 --learning-rate-drop-period 50 --num_training_images 0`
. In this example, `fold = 0` and `num_training_images = 0` mean that the training will use all training data without fold cross-validation. If you would

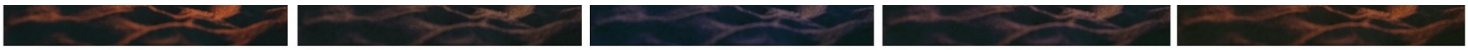
like to limit the number of training images to be `n` images, set `num_training_images` to `n`. If you would like to do 3-fold cross-validation, use `fold = testing_fold`. Then the code will train on the remaining folds and leave the selected fold for testing.

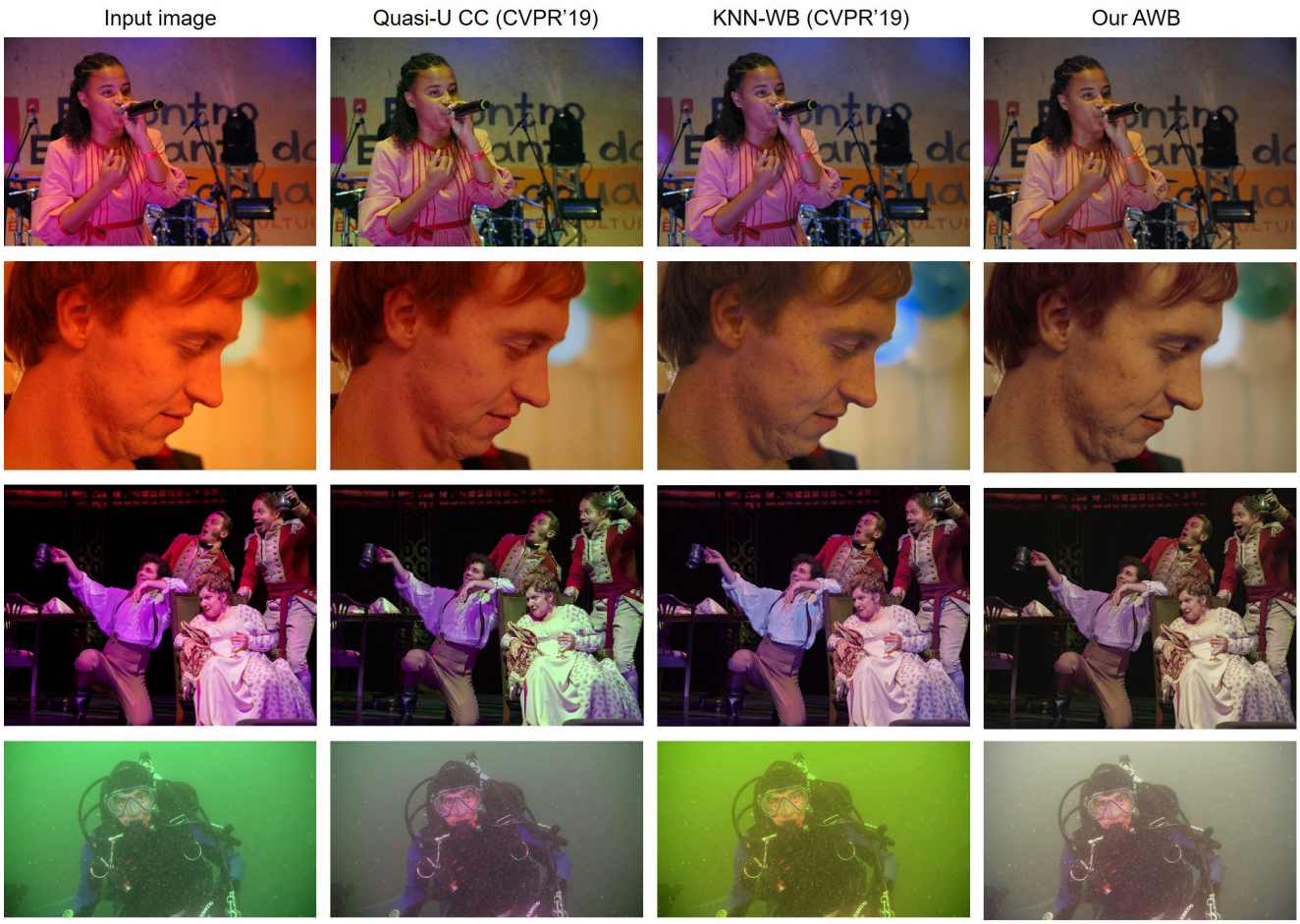
Other useful options include: `--patches-per-image` to select the number of random patches per image, `--learning-rate-drop-period` and `--learning-rate-drop-factor` to control the learning rate drop period and factor, respectively, and `--patch-size` to set the size of training patches. You can continue training from a training checkpoint `.pth` file using `--load` option.

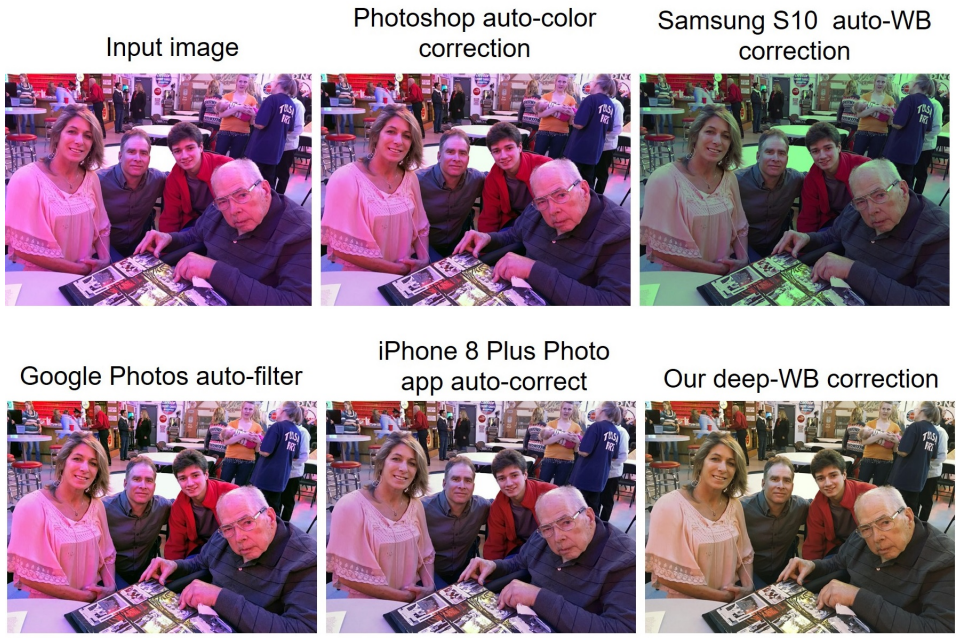
If you have TensorBoard installed on your machine, run `tensorboard --logdir ./runs` after start training to check training progress and visualize samples of input/output patches.

Results


Input images
Our AWB
Our Incandescent WB
Our Fluorescent WB
Our Shade WB







This software is provided for research purposes only and CAN NOT be used for commercial purposes.

Maintainer: Mahmoud Afifi (m.3afifi@gmail.com)

Related Research Projects

- When Color Constancy Goes Wrong: The first work to directly address the problem of incorrectly white-balanced images; requires a small memory overhead and it is fast (CVPR 2019).
- White-Balance Augmenter: An augmentation technique based on camera WB errors (ICCV 2019).
- Interactive White Balancing:A simple method to link the nonlinear white-balance correction to the user's selected colors to allow interactive white-balance manipulation (CIC 2020).
- Exposure Correction: A single coarse-to-fine deep learning model with adversarial training to correct both over- and under-exposed photographs (CVPR 2021).