

SIM Denoising Pipeline

Generated by Doxygen 1.9.1

1 Namespace Index	1
1.1 Packages	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	9
5.1 analyse Namespace Reference	9
5.1.1 Function Documentation	10
5.1.1.1 reshape_to_bcwh()	10
5.1.2 Variable Documentation	10
5.1.2.1 args	10
5.1.2.2 ckpt	10
5.1.2.3 cmap	10
5.1.2.4 default	10
5.1.2.5 device	11
5.1.2.6 df	11
5.1.2.7 exist_ok	11
5.1.2.8 gt	11
5.1.2.9 gt_dir	11
5.1.2.10 gt_files	11
5.1.2.11 gt_samples	12
5.1.2.12 img_idx	12
5.1.2.13 int	12
5.1.2.14 model	12
5.1.2.15 model_1	12
5.1.2.16 model_1_dir	12
5.1.2.17 model_1_files	12
5.1.2.18 model_1_samples	13
5.1.2.19 model_2	13
5.1.2.20 model_2_dir	13
5.1.2.21 model_2_files	13
5.1.2.22 model_2_samples	13
5.1.2.23 N	13
5.1.2.24 output_dir	13
5.1.2.25 parents	14
5.1.2.26 parser	14
5.1.2.27 psnr	14

5.1.2.28 raw	14
5.1.2.29 raw_dir	14
5.1.2.30 raw_files	14
5.1.2.31 raw_samples	14
5.1.2.32 RCAN_hyperparameters	14
5.1.2.33 required	15
5.1.2.34 rng	15
5.1.2.35 ssim	15
5.1.2.36 str	15
5.1.2.37 True	15
5.1.2.38 type	15
5.2 apply Namespace Reference	15
5.2.1 Function Documentation	16
5.2.1.1 normalize_between_zero_and_one()	16
5.2.2 Variable Documentation	16
5.2.2.1 action	16
5.2.2.2 args	17
5.2.2.3 choices	17
5.2.2.4 ckpt	17
5.2.2.5 data	17
5.2.2.6 default	17
5.2.2.7 device	17
5.2.2.8 imagej	17
5.2.2.9 input_path	18
5.2.2.10 int	18
5.2.2.11 model	18
5.2.2.12 output_file	18
5.2.2.13 output_path	18
5.2.2.14 overlap_shape	18
5.2.2.15 parents	18
5.2.2.16 parser	19
5.2.2.17 percentile	19
5.2.2.18 raw	19
5.2.2.19 raw_files	19
5.2.2.20 RCAN_hyperparameters	19
5.2.2.21 required	19
5.2.2.22 restored	19
5.2.2.23 str	20
5.2.2.24 type	20
5.3 convert_omx_to_czxy Namespace Reference	20
5.3.1 Variable Documentation	20
5.3.1.1 action	20

5.3.1.2 args	20
5.3.1.3 converted	21
5.3.1.4 imagej	21
5.3.1.5 input_dir	21
5.3.1.6 input_files	21
5.3.1.7 int	21
5.3.1.8 n_angles	21
5.3.1.9 n_phases	21
5.3.1.10 original	22
5.3.1.11 parser	22
5.3.1.12 required	22
5.3.1.13 str	22
5.3.1.14 type	22
5.4 convert_omx_to_paz Namespace Reference	22
5.4.1 Variable Documentation	23
5.4.1.1 action	23
5.4.1.2 args	23
5.4.1.3 converted	23
5.4.1.4 imagej	23
5.4.1.5 input_dir	23
5.4.1.6 input_files	23
5.4.1.7 int	23
5.4.1.8 n_angles	24
5.4.1.9 n_phases	24
5.4.1.10 original	24
5.4.1.11 parser	24
5.4.1.12 required	24
5.4.1.13 str	24
5.4.1.14 type	24
5.5 convert_slices_to_volumes Namespace Reference	25
5.5.1 Variable Documentation	25
5.5.1.1 args	25
5.5.1.2 default	25
5.5.1.3 exist_ok	25
5.5.1.4 imagej	26
5.5.1.5 input_dir	26
5.5.1.6 input_files	26
5.5.1.7 input_slice	26
5.5.1.8 output_dir	26
5.5.1.9 output_file	26
5.5.1.10 parents	26
5.5.1.11 parser	27

5.5.1.12 required	27
5.5.1.13 str	27
5.5.1.14 subvolume	27
5.5.1.15 True	27
5.5.1.16 tuple_of_ints	27
5.5.1.17 type	27
5.5.1.18 volume	28
5.6 generate_sim Namespace Reference	28
5.6.1 Function Documentation	28
5.6.1.1 arange_zero()	28
5.6.1.2 threshold_norm()	28
5.6.2 Variable Documentation	29
5.6.2.1 args	29
5.6.2.2 default	29
5.6.2.3 int	29
5.6.2.4 parser	29
5.6.2.5 required	29
5.6.2.6 runner	29
5.6.2.7 str	30
5.6.2.8 type	30
5.7 image_noising Namespace Reference	30
5.7.1 Function Documentation	31
5.7.1.1 save_image_pair()	31
5.7.2 Variable Documentation	31
5.7.2.1 args	31
5.7.2.2 choices	31
5.7.2.3 data	31
5.7.2.4 default	31
5.7.2.5 float	31
5.7.2.6 gt	32
5.7.2.7 img_idx_all	32
5.7.2.8 img_idx_test	32
5.7.2.9 img_idx_train	32
5.7.2.10 img_idx_val	32
5.7.2.11 input_path	32
5.7.2.12 int	32
5.7.2.13 n_acquisitions	32
5.7.2.14 n_img	33
5.7.2.15 output_path	33
5.7.2.16 output_test_gt_path	33
5.7.2.17 output_test_raw_path	33
5.7.2.18 output_train_gt_path	33

5.7.2.19 output_train_raw_path	33
5.7.2.20 output_val_gt_path	33
5.7.2.21 output_val_raw_path	33
5.7.2.22 parents	34
5.7.2.23 parser	34
5.7.2.24 required	34
5.7.2.25 rng	34
5.7.2.26 split	34
5.7.2.27 str	34
5.7.2.28 train_size	34
5.7.2.29 type	34
5.7.2.30 val_size	35
5.8 manage_stack Namespace Reference	35
5.8.1 Variable Documentation	35
5.8.1.1 action	35
5.8.1.2 args	35
5.8.1.3 choices	36
5.8.1.4 default	36
5.8.1.5 exist_ok	36
5.8.1.6 filename	36
5.8.1.7 files	36
5.8.1.8 img_data	36
5.8.1.9 int	36
5.8.1.10 n_acq	37
5.8.1.11 n_z	37
5.8.1.12 number_of_stacks	37
5.8.1.13 output_data	37
5.8.1.14 output_dir	37
5.8.1.15 output_file	37
5.8.1.16 parents	37
5.8.1.17 parser	38
5.8.1.18 required	38
5.8.1.19 sample	38
5.8.1.20 stack	38
5.8.1.21 stack_number	38
5.8.1.22 str	38
5.8.1.23 True	39
5.8.1.24 type	39
5.9 rcan Namespace Reference	39
5.10 rcan.data_generator Namespace Reference	39
5.10.1 Function Documentation	39
5.10.1.1 load_SIM_dataset()	39

5.11 rcan.model Namespace Reference	40
5.11.1 Function Documentation	41
5.11.1.1 _conv()	41
5.11.1.2 _destandardize()	41
5.11.1.3 _global_average_pooling()	42
5.11.1.4 _standardize()	42
5.12 rcan.plotting Namespace Reference	42
5.12.1 Function Documentation	42
5.12.1.1 plot_learning_curve()	43
5.12.1.2 plot_reconstructions()	43
5.13 rcan.utils Namespace Reference	44
5.13.1 Function Documentation	44
5.13.1.1 apply()	44
5.13.1.2 load_rcan_checkpoint()	45
5.13.1.3 normalize()	45
5.13.1.4 References	46
5.13.1.5 percentile()	46
5.13.1.6 tuple_of_ints()	46
5.14 recon_postprocess Namespace Reference	46
5.14.1 Variable Documentation	46
5.14.1.1 args	46
5.14.1.2 files	46
5.14.1.3 img_data	47
5.14.1.4 parser	47
5.14.1.5 required	47
5.14.1.6 str	47
5.14.1.7 type	47
5.15 recon_preprocess Namespace Reference	47
5.15.1 Function Documentation	48
5.15.1.1 normalize_acquisition_intensity()	48
5.15.2 Variable Documentation	48
5.15.2.1 action	48
5.15.2.2 args	48
5.15.2.3 choices	48
5.15.2.4 default	48
5.15.2.5 exist_ok	48
5.15.2.6 files	49
5.15.2.7 img_data	49
5.15.2.8 int	49
5.15.2.9 output_dir	49
5.15.2.10 output_file	49
5.15.2.11 parents	49

5.15.2.12 parser	49
5.15.2.13 percentile	49
5.15.2.14 required	50
5.15.2.15 str	50
5.15.2.16 True	50
5.15.2.17 type	50
5.16 stats Namespace Reference	50
5.16.1 Function Documentation	51
5.16.1.1 paired_t()	51
5.16.2 Variable Documentation	51
5.16.2.1 alpha	52
5.16.2.2 args	52
5.16.2.3 ax	52
5.16.2.4 choices	52
5.16.2.5 color	52
5.16.2.6 data	52
5.16.2.7 default	52
5.16.2.8 df	53
5.16.2.9 dflong	53
5.16.2.10 dflongssim	53
5.16.2.11 dodge	53
5.16.2.12 exist_ok	53
5.16.2.13 fig	54
5.16.2.14 figsize	54
5.16.2.15 hist_range_psnr	54
5.16.2.16 hist_range_ssim	54
5.16.2.17 hue	54
5.16.2.18 int	54
5.16.2.19 legend	55
5.16.2.20 lw	55
5.16.2.21 mean_psnr_1	55
5.16.2.22 mean_psnr_2	55
5.16.2.23 mean_ssim_1	55
5.16.2.24 mean_ssim_2	55
5.16.2.25 output_dir	55
5.16.2.26 palette	56
5.16.2.27 parents	56
5.16.2.28 parser	56
5.16.2.29 psnr_cols	56
5.16.2.30 psnr_diff_1_max	56
5.16.2.31 psnr_diff_1_min	56
5.16.2.32 psnr_diff_2_max	57

5.16.2.33 psnr_diff_2_min	57
5.16.2.34 range	57
5.16.2.35 required	57
5.16.2.36 se_psnr_1	57
5.16.2.37 se_psnr_2	57
5.16.2.38 se_ssim_1	58
5.16.2.39 se_ssim_2	58
5.16.2.40 ssim_cols	58
5.16.2.41 ssim_diff_1_max	58
5.16.2.42 ssim_diff_1_min	58
5.16.2.43 ssim_diff_2_max	59
5.16.2.44 ssim_diff_2_min	59
5.16.2.45 str	59
5.16.2.46 title	59
5.16.2.47 True	59
5.16.2.48 type	59
5.16.2.49 x	59
5.16.2.50 xlabel	60
5.16.2.51 y	60
5.17 synthetic_sim Namespace Reference	60
5.18 synthetic_sim.otf Namespace Reference	60
5.18.1 Function Documentation	60
5.18.1.1 calc_psf()	60
5.19 train Namespace Reference	61
5.19.1 Function Documentation	61
5.19.1.1 load_data_paths()	62
5.19.1.2 train()	62
5.19.2 Variable Documentation	62
5.19.2.1 args	62
5.19.2.2 ckpt	62
5.19.2.3 ckpt_path	62
5.19.2.4 config	63
5.19.2.5 device	63
5.19.2.6 exist_ok	63
5.19.2.7 input_shape	63
5.19.2.8 losses_train_epoch	63
5.19.2.9 losses_val_epoch	63
5.19.2.10 model	63
5.19.2.11 n_accumulations	64
5.19.2.12 ndim	64
5.19.2.13 nepoch	64
5.19.2.14 optimizer	64

5.19.2.15 output_dir	64
5.19.2.16 parents	64
5.19.2.17 parser	64
5.19.2.18 psnr_train_epoch	65
5.19.2.19 psnr_val_epoch	65
5.19.2.20 RCAN_hyperparameters	65
5.19.2.21 required	65
5.19.2.22 saveinterval	65
5.19.2.23 scheduler	65
5.19.2.24 schema	66
5.19.2.25 ssim_train_epoch	66
5.19.2.26 ssim_val_epoch	66
5.19.2.27 start_epoch	66
5.19.2.28 str	66
5.19.2.29 train_loader	66
5.19.2.30 True	67
5.19.2.31 type	67
5.19.2.32 val_loader	67
6 Class Documentation	69
6.1 rcnn.model._channel_attention_block Class Reference	69
6.1.1 Detailed Description	70
6.1.1.1 References	70
6.1.2 Constructor & Destructor Documentation	70
6.1.2.1 __init__()	70
6.1.3 Member Function Documentation	71
6.1.3.1 forward()	71
6.1.4 Member Data Documentation	71
6.1.4.1 conv_1	71
6.1.4.2 conv_2	71
6.1.4.3 global_average_pooling	71
6.2 rcnn.model._residual_channel_attention_blocks Class Reference	72
6.2.1 Detailed Description	73
6.2.1.1 References	73
6.2.2 Constructor & Destructor Documentation	73
6.2.2.1 __init__()	73
6.2.3 Member Function Documentation	73
6.2.3.1 forward()	73
6.2.4 Member Data Documentation	74
6.2.4.1 channel_attention_block_list	74
6.2.4.2 conv_list	74
6.2.4.3 repeat	74

6.2.4.4 residual_scaling	74
6.3 synthetic_sim.off.PsfParameters Class Reference	74
6.3.1 Detailed Description	75
6.3.2 Member Data Documentation	75
6.3.2.1 Callable	75
6.3.2.2 float	75
6.3.2.3 int	75
6.4 rcan.model.RCAN Class Reference	76
6.4.1 Detailed Description	77
6.4.1.1 References	77
6.4.2 Constructor & Destructor Documentation	77
6.4.2.1 __init__()	77
6.4.3 Member Function Documentation	78
6.4.3.1 forward()	78
6.4.4 Member Data Documentation	78
6.4.4.1 conv_input	78
6.4.4.2 conv_list	78
6.4.4.3 conv_output	78
6.4.4.4 num_residual_groups	79
6.4.4.5 rcab_list	79
6.5 rcan.data_generator.SIM_Dataset Class Reference	79
6.5.1 Detailed Description	80
6.5.2 Constructor & Destructor Documentation	80
6.5.2.1 __init__()	80
6.5.3 Member Function Documentation	81
6.5.3.1 __getitem__()	81
6.5.3.2 __len__()	82
6.5.3.3 _scale()	82
6.5.4 Member Data Documentation	82
6.5.4.1 _area_threshold	82
6.5.4.2 _intensity_threshold	82
6.5.4.3 _scale_factor	82
6.5.4.4 _shape	82
6.5.4.5 _transform_function	82
6.5.4.6 _y	83
6.5.4.7 output_shape	83
6.5.4.8 output_signature	83
6.5.4.9 p_max	83
6.5.4.10 p_min	83
6.5.4.11 steps_per_epoch	83
6.6 generate_sim.SimulationRunner Class Reference	83
6.6.1 Detailed Description	84

6.6.2 Constructor & Destructor Documentation	84
6.6.2.1 <code>__init__()</code>	84
6.6.3 Member Function Documentation	84
6.6.3.1 <code>do_sim()</code>	84
6.6.3.2 <code>run()</code>	85
6.6.4 Member Data Documentation	85
6.6.4.1 <code>input_dir</code>	85
6.6.4.2 <code>input_files</code>	85
6.6.4.3 <code>output_dir</code>	85
6.6.4.4 <code>range</code>	85
6.6.4.5 <code>z_offset</code>	85
6.7 <code>generate_sim.Simulator</code> Class Reference	86
6.7.1 Detailed Description	87
6.7.2 Constructor & Destructor Documentation	87
6.7.2.1 <code>__init__()</code>	87
6.7.3 Member Function Documentation	87
6.7.3.1 <code>add_noise()</code>	87
6.7.3.2 <code>illumination()</code>	87
6.7.3.3 <code>in_focus_plane()</code>	88
6.7.3.4 <code>params_dict()</code>	88
6.7.3.5 <code>psf()</code>	88
6.7.3.6 <code>psf_params()</code>	88
6.7.3.7 <code>randomise()</code>	88
6.7.3.8 <code>simulate_ideal_superres()</code>	88
6.7.3.9 <code>simulate_sim()</code>	89
6.7.3.10 <code>wavevectors()</code>	89
6.7.4 Member Data Documentation	89
6.7.4.1 <code>_illumination</code>	89
6.7.4.2 <code>_psf</code>	89
6.7.4.3 <code>_superres_psf</code>	89
6.7.4.4 <code>angle_error</code>	89
6.7.4.5 <code>beam_position</code>	90
6.7.4.6 <code>delta_z_p</code>	90
6.7.4.7 <code>k0</code>	90
6.7.4.8 <code>k_exc</code>	90
6.7.4.9 <code>lambda0</code>	90
6.7.4.10 <code>lambda_exc</code>	90
6.7.4.11 <code>n_angles</code>	90
6.7.4.12 <code>n_g</code>	90
6.7.4.13 <code>n_i</code>	91
6.7.4.14 <code>n_rotations</code>	91
6.7.4.15 <code>n_sample</code>	91

6.7.4.16 n_shifts	91
6.7.4.17 n_x	91
6.7.4.18 n_z	91
6.7.4.19 poisson_photons	91
6.7.4.20 res_axial	91
6.7.4.21 res_lateral	92
6.7.4.22 signal_to_noise	92
6.7.4.23 z	92
6.7.4.24 z_p	92
7 File Documentation	93
7.1 /home/jhughes2712/projects/sim_project/jh2284/src/analyse.py File Reference	93
7.1.1 Detailed Description	94
7.2 /home/jhughes2712/projects/sim_project/jh2284/src/apply.py File Reference	94
7.2.1 Detailed Description	95
7.3 /home/jhughes2712/projects/sim_project/jh2284/src/convert_omx_to_czxy.py File Reference	96
7.3.1 Detailed Description	96
7.4 /home/jhughes2712/projects/sim_project/jh2284/src/convert_omx_to_paz.py File Reference	97
7.4.1 Detailed Description	97
7.5 /home/jhughes2712/projects/sim_project/jh2284/src/convert_slices_to_volumes.py File Reference	97
7.5.1 Detailed Description	98
7.6 /home/jhughes2712/projects/sim_project/jh2284/src/generate_sim.py File Reference	98
7.6.1 Detailed Description	99
7.7 /home/jhughes2712/projects/sim_project/jh2284/src/image_noising.py File Reference	99
7.7.1 Detailed Description	100
7.8 /home/jhughes2712/projects/sim_project/jh2284/src/manage_stack.py File Reference	101
7.8.1 Detailed Description	101
7.9 /home/jhughes2712/projects/sim_project/jh2284/src/rcan/__init__.py File Reference	102
7.10 /home/jhughes2712/projects/sim_project/jh2284/src/synthetic_sim/__init__.py File Reference	102
7.11 /home/jhughes2712/projects/sim_project/jh2284/src/rcan/data_generator.py File Reference	102
7.11.1 Detailed Description	103
7.12 /home/jhughes2712/projects/sim_project/jh2284/src/rcan/model.py File Reference	103
7.12.1 Detailed Description	104
7.13 /home/jhughes2712/projects/sim_project/jh2284/src/rcan/plotting.py File Reference	104
7.13.1 Detailed Description	104
7.14 /home/jhughes2712/projects/sim_project/jh2284/src/rcan/utils.py File Reference	104
7.14.1 Detailed Description	105
7.15 /home/jhughes2712/projects/sim_project/jh2284/src/recon_postprocess.py File Reference	105
7.15.1 Detailed Description	106
7.16 /home/jhughes2712/projects/sim_project/jh2284/src/recon_preprocess.py File Reference	106
7.16.1 Detailed Description	107
7.17 /home/jhughes2712/projects/sim_project/jh2284/src/stats.py File Reference	107

7.18 /home/jhughes2712/projects/sim_project/jh2284/src/synthetic_sim/otf.py File Reference	108
7.19 /home/jhughes2712/projects/sim_project/jh2284/src/train.py File Reference	109
7.19.1 Detailed Description	110
Index	111

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

analyse	9
apply	15
convert_omx_to_czxy	20
convert_omx_to_paz	22
convert_slices_to_volumes	25
generate_sim	28
image_noising	30
manage_stack	35
rcan	39
rcan.data_generator	39
rcan.model	40
rcan.plotting	42
rcan.utils	44
recon_postprocess	46
recon_preprocess	47
stats	50
synthetic_sim	60
synthetic_sim.otf	60
train	61

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

torch.nn.Module	
rcan.model.RCAN	76
rcan.model._channel_attention_block	69
rcan.model._residual_channel_attention_blocks	72
synthetic_sim.otf.PsfParameters	74
generate_sim.SimulationRunner	83
generate_sim.Simulator	86
Dataset	
rcan.data_generator.SIM_Dataset	79

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

rcan.model._channel_attention_block	
Implements channel attention block/layer	69
rcan.model._residual_channel_attention_blocks	
Implements residual group based on [1]	72
synthetic_sim.otf.PsfParameters	
Class to store PSF parameters	74
rcan.model.RCAN	
Builds a residual channel attention network	76
rcan.data_generator.SIM_Dataset	
Generates batches of images with real-time data augmentation	79
generate_sim.SimulationRunner	
Class which performs a batch of simulations, either sequentially or in parallel	83
generate_sim.Simulator	
The Simulator class encapsulates the state of a 3D microscope simulation	86

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

/home/jhughes2712/projects/sim_project/jh2284/src/ analyse.py	
Script producing plots and small datasets that summarise the performance of models	93
/home/jhughes2712/projects/sim_project/jh2284/src/ apply.py	
Script producing restored images resulting from an RCAN denoiser being applied to low SNR images	94
/home/jhughes2712/projects/sim_project/jh2284/src/ convert_omx_to_czxy.py	
Script enabling .tif file conversion between OMX and CZXY	96
/home/jhughes2712/projects/sim_project/jh2284/src/ convert_omx_to_paz.py	
Script enabling .tif file conversion between OMX and PAZ	97
/home/jhughes2712/projects/sim_project/jh2284/src/ convert_slices_to_volumes.py	
Script enabling construction of 3D image volumes from large RGB 2D image slices	97
/home/jhughes2712/projects/sim_project/jh2284/src/ generate_sim.py	
Script simulating the acquisition of 3D SIM image volumes	98
/home/jhughes2712/projects/sim_project/jh2284/src/ image_noising.py	
Script which converts a directory of high-SNR SIM images into a training dataset	99
/home/jhughes2712/projects/sim_project/jh2284/src/ manage_stack.py	
Script handling the stacking and unstacking of groups of images, for the purpose of batch reconstructions	101
/home/jhughes2712/projects/sim_project/jh2284/src/ recon_postprocess.py	
Script handling the postprocessing of SIM reconstructions	105
/home/jhughes2712/projects/sim_project/jh2284/src/ recon_preprocess.py	
Script handling the preprocessing of images before SIM reconstruction	106
/home/jhughes2712/projects/sim_project/jh2284/src/ stats.py	107
/home/jhughes2712/projects/sim_project/jh2284/src/ train.py	
Script used to train RCAN	109
/home/jhughes2712/projects/sim_project/jh2284/src/rcan/ __init__.py	102
/home/jhughes2712/projects/sim_project/jh2284/src/rcan/ data_generator.py	
Module that handles processing and batching of data during training loop	102
/home/jhughes2712/projects/sim_project/jh2284/src/rcan/ model.py	
Module defining the RCAN model architecture	103
/home/jhughes2712/projects/sim_project/jh2284/src/rcan/ plotting.py	
Module providing helper functions for matplotlib plots	104
/home/jhughes2712/projects/sim_project/jh2284/src/rcan/ utils.py	
Contains utility functions for the training loop and inference	104
/home/jhughes2712/projects/sim_project/jh2284/src/synthetic_sim/ __init__.py	102
/home/jhughes2712/projects/sim_project/jh2284/src/synthetic_sim/ otf.py	108

Chapter 5

Namespace Documentation

5.1 analyse Namespace Reference

Functions

- def `reshape_to_bcwh` (data)

Variables

- `parser` = argparse.ArgumentParser()
- `type`
- `str`
- `required`
- `default`
- `int`
- `args` = parser.parse_args()
- `output_dir` = pathlib.Path(args.output_dir)
- `parents`
- `True`
- `exist_ok`
- tuple `device`
- `ckpt`
- `model`
- `RCAN_hyperparameters` = `ckpt`["hyperparameters"]
- `gt_dir` = pathlib.Path(args.gt_dir)
- `raw_dir` = pathlib.Path(args.raw_dir)
- `model_1_dir` = pathlib.Path(args.model_1_dir)
- `gt_files` = sorted(list(gt_dir.glob(args.glob_str)))
- `raw_files` = sorted(list(raw_dir.glob(args.glob_str)))
- `model_1_files` = sorted(list(model_1_dir.glob(args.glob_str)))
- `model_2_dir` = pathlib.Path(args.model_2_dir)
- `model_2_files` = sorted(list(model_2_dir.glob(args.glob_str)))
- `psnr` = PSNR(data_range=65536, `device`=`device`)
- `ssim`
- `df`
- `N` = len(`gt_files`)
- def `gt` = `reshape_to_bcwh`(tifffile.imread(`gt_files`[i]))

- `def raw = reshape_to_bcwh(tifffile.imread(raw_files[i]))`
- `def model_1 = reshape_to_bcwh(tifffile.imread(model_1_files[i]))`
- `def model_2 = reshape_to_bcwh(tifffile.imread(model_2_files[i]))`
- `rng = np.random.default_rng(seed=31052024)`
- `img_idx = list(range(N))`
- `list gt_samples = [np.squeeze(tifffile.imread(gt_files[i])) for i in img_idx]`
- `list raw_samples = [np.squeeze(tifffile.imread(raw_files[i])) for i in img_idx]`
- `list model_1_samples`
- `list model_2_samples`
- `cmap`

5.1.1 Function Documentation

5.1.1.1 reshape_to_bcwh()

```
def analyse.reshape_to_bcwh (
    data )
```

5.1.2 Variable Documentation

5.1.2.1 args

```
analyse.args = parser.parse_args()
```

5.1.2.2 ckpt

```
analyse.ckpt
```

5.1.2.3 cmap

```
analyse.cmap
```

5.1.2.4 default

```
analyse.default
```

5.1.2.5 device

tuple analyse.device

Initial value:

```
1 = (  
2     torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")  
3 )
```

5.1.2.6 df

analyse.df

Initial value:

```
1 = pd.DataFrame(  
2     columns=[  
3         "file",  
4         "psnr_raw",  
5         "psnr_model_1",  
6         "psnr_model_2",  
7         "ssim_raw",  
8         "ssim_model_1",  
9         "ssim_model_2",  
10    ]  
11 )
```

5.1.2.7 exist_ok

analyse.exist_ok

5.1.2.8 gt

```
def analyse.gt = reshape_to_bcwh(tifffile.imread(gt_files[i]))
```

5.1.2.9 gt_dir

```
analyse.gt_dir = pathlib.Path(args.gt_dir)
```

5.1.2.10 gt_files

```
analyse.gt_files = sorted(list(gt_dir.glob(args.glob_str)))
```

5.1.2.11 `gt_samples`

```
list analyse.gt_samples = [np.squeeze(tifffile.imread(gt_files[i])) for i in img_idx]
```

5.1.2.12 `img_idx`

```
analyse.img_idx = list(range(N))
```

5.1.2.13 `int`

```
analyse.int
```

5.1.2.14 `model`

```
analyse.model
```

5.1.2.15 `model_1`

```
def analyse.model_1 = reshape_to_bcwh(tifffile.imread(model_1_files[i]))
```

5.1.2.16 `model_1_dir`

```
analyse.model_1_dir = pathlib.Path(args.model_1_dir)
```

5.1.2.17 `model_1_files`

```
analyse.model_1_files = sorted(list(model_1_dir.glob(args.glob_str)))
```

5.1.2.18 model_1_samples

```
list analyse.model_1_samples
```

Initial value:

```
1 = [  
2     np.squeeze(tifffile.imread(model_1_files[i])) for i in img_idx  
3 ]
```

5.1.2.19 model_2

```
def analyse.model_2 = reshape_to_bcwh(tifffile.imread(model_2_files[i]))
```

5.1.2.20 model_2_dir

```
analyse.model_2_dir = pathlib.Path(args.model_2_dir)
```

5.1.2.21 model_2_files

```
list analyse.model_2_files = sorted(list(model_2_dir.glob(args.glob_str)))
```

5.1.2.22 model_2_samples

```
analyse.model_2_samples
```

Initial value:

```
1 = [  
2     np.squeeze(tifffile.imread(model_2_files[i])) for i in img_idx  
3 ]
```

5.1.2.23 N

```
analyse.N = len(gt_files)
```

5.1.2.24 output_dir

```
analyse.output_dir = pathlib.Path(args.output_dir)
```

5.1.2.25 parents

```
analyse.parents
```

5.1.2.26 parser

```
analyse.parser = argparse.ArgumentParser()
```

5.1.2.27 psnr

```
analyse.psnr = PSNR(data_range=65536, device=device)
```

5.1.2.28 raw

```
def analyse.raw = reshape_to_bcwh(tifffile.imread(raw_files[i]))
```

5.1.2.29 raw_dir

```
analyse.raw_dir = pathlib.Path(args.raw_dir)
```

5.1.2.30 raw_files

```
analyse.raw_files = sorted(list(raw_dir.glob(args.glob_str)))
```

5.1.2.31 raw_samples

```
list analyse.raw_samples = [np.squeeze(tifffile.imread(raw_files[i])) for i in img_idx]
```

5.1.2.32 RCAN_hyperparameters

```
analyse.RCAN_hyperparameters = ckpt["hyperparameters"]
```

5.1.2.33 required

```
analyse.required
```

5.1.2.34 rng

```
analyse.rng = np.random.default_rng(seed=31052024)
```

5.1.2.35 ssim

```
analyse.ssim
```

Initial value:

```
1 = SSIM(  
2     data_range=65536,  
3     kernel_size=(11, 11),  
4     sigma=(1.5, 1.5),  
5     k1=0.01,  
6     k2=0.03,  
7     gaussian=True,  
8     device=device,  
9 )
```

5.1.2.36 str

```
analyse.str
```

5.1.2.37 True

```
analyse.True
```

5.1.2.38 type

```
analyse.type
```

5.2 apply Namespace Reference

Functions

- def [normalize_between_zero_and_one](#) (m)

Variables

- `parser` = `argparse.ArgumentParser()`
- `type`
- `str`
- `required`
- `int`
- `choices`
- `default`
- `percentile`
- `action`
- `args` = `parser.parse_args()`
- `input_path` = `pathlib.Path(args.input)`
- `output_path` = `pathlib.Path(args.output)`
- `parents`
- `raw_files` = `sorted(input_path.glob("*.tif"))`
- `data` = `itertools.zip_longest(raw_files, [])`
- tuple `device`
- `ckpt`
- `model`
- `RCAN_hyperparameters` = `ckpt["hyperparameters"]`
- list `overlap_shape`
- `raw` = `normalize(tifffile.imread(raw_file), args.p_min, args.p_max)`
- `restored`
- `output_file` = `output_path / ("pred_" + raw_file.name)`
- `imagej`

5.2.1 Function Documentation

5.2.1.1 `normalize_between_zero_and_one()`

```
def apply.normalize_between_zero_and_one (
    m )
```

5.2.2 Variable Documentation

5.2.2.1 `action`

```
apply.action
```


5.2.2.2 args

```
apply.args = parser.parse_args()
```

5.2.2.3 choices

```
apply.choices
```

5.2.2.4 ckpt

```
apply.ckpt
```

5.2.2.5 data

```
list apply.data = itertools.zip_longest(raw_files, [])
```

5.2.2.6 default

```
apply.default
```

5.2.2.7 device

```
tuple apply.device
```

Initial value:

```
1 = (  
2     torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")  
3 )
```

5.2.2.8 imagej

```
apply.imagej
```

5.2.2.9 input_path

```
apply.input_path = pathlib.Path(args.input)
```

5.2.2.10 int

```
apply.int
```

5.2.2.11 model

```
apply.model
```

5.2.2.12 output_file

```
apply.output_file = output_path / ("pred_" + raw_file.name)
```

5.2.2.13 output_path

```
apply.output_path = pathlib.Path(args.output)
```

5.2.2.14 overlap_shape

```
apply.overlap_shape
```

Initial value:

```
1 = [  
2     max(1, x // 8) if x > 2 else 0  
3     for x in RCAN_hyperparameters["input_shape"]  
4 ]
```

5.2.2.15 parents

```
apply.parents
```

5.2.2.16 parser

```
apply.parser = argparse.ArgumentParser()
```

5.2.2.17 percentile

```
apply.percentile
```

5.2.2.18 raw

```
apply.raw = normalize(tifffile.imread(raw_file), args.p_min, args.p_max)
```

5.2.2.19 raw_files

```
apply.raw_files = sorted(input_path.glob("*.tif"))
```

5.2.2.20 RCAN_hyperparameters

```
apply.RCAN_hyperparameters = ckpt["hyperparameters"]
```

5.2.2.21 required

```
apply.required
```

5.2.2.22 restored

```
def apply.restored
```

Initial value:

```
1 = apply(  
2     model,  
3     raw,  
4     RCAN_hyperparameters["input_shape"],  
5     RCAN_hyperparameters["input_shape"],  
6     RCAN_hyperparameters["num_input_channels"],  
7     RCAN_hyperparameters["num_output_channels"],  
8     batch_size=1,  
9     device=device,  
10    overlap_shape=overlap_shape,  
11    verbose=True,  
12 )
```

5.2.2.23 str

`apply.str`

5.2.2.24 type

`apply.type`

5.3 convert_omx_to_czxy Namespace Reference

Variables

- `parser` = `argparse.ArgumentParser()`
- `type`
- `str`
- `required`
- `int`
- `action`
- `args` = `parser.parse_args()`
- `input_dir` = `pathlib.Path(args.input)`
- `input_files` = `sorted(input_dir.rglob("*.tif"))`
- `original` = `tiffimage.imread(input_file)`
- `n_phases` = `args.num_phases`
- `n_angles` = `args.num_angles`
- `converted`
- `imagej`

5.3.1 Variable Documentation

5.3.1.1 action

`convert_omx_to_czxy.action`

5.3.1.2 args

`convert_omx_to_czxy.args = parser.parse_args()`

5.3.1.3 converted

convert_omx_to_czxy.converted

Initial value:

```
1 = np.zeros(  
2     (  
3         n_phases * n_angles,  
4         original.shape[0] // (n_phases * n_angles),  
5         *original.shape[1:],  
6     )  
7 )
```

5.3.1.4 imagej

convert_omx_to_czxy.imagej

5.3.1.5 input_dir

convert_omx_to_czxy.input_dir = pathlib.Path(args.input)

5.3.1.6 input_files

convert_omx_to_czxy.input_files = sorted(input_dir.rglob("*.tif"))

5.3.1.7 int

convert_omx_to_czxy.int

5.3.1.8 n_angles

convert_omx_to_czxy.n_angles = args.num_angles

5.3.1.9 n_phases

convert_omx_to_czxy.n_phases = args.num_phases

5.3.1.10 original

```
convert_omx_to_czxy.original = tifffile.imread(input_file)
```

5.3.1.11 parser

```
convert_omx_to_czxy.parser = argparse.ArgumentParser()
```

5.3.1.12 required

```
convert_omx_to_czxy.required
```

5.3.1.13 str

```
convert_omx_to_czxy.str
```

5.3.1.14 type

```
convert_omx_to_czxy.type
```

5.4 convert_omx_to_paz Namespace Reference

Variables

- `parser` = `argparse.ArgumentParser()`
- `type`
- `str`
- `required`
- `int`
- `action`
- `args` = `parser.parse_args()`
- `input_dir` = `pathlib.Path(args.input)`
- `input_files` = `sorted(input_dir.rglob("*.tif"))`
- `original` = `tifffile.imread(input_file)`
- `n_phases` = `args.num_phases`
- `n_angles` = `args.num_angles`
- `converted` = `np.zeros_like(original)`
- `imagej`

5.4.1 Variable Documentation

5.4.1.1 action

```
convert_omx_to_paz.action
```

5.4.1.2 args

```
convert_omx_to_paz.args = parser.parse_args()
```

5.4.1.3 converted

```
convert_omx_to_paz.converted = np.zeros_like(original)
```

5.4.1.4 imagej

```
convert_omx_to_paz.imagej
```

5.4.1.5 input_dir

```
convert_omx_to_paz.input_dir = pathlib.Path(args.input)
```

5.4.1.6 input_files

```
convert_omx_to_paz.input_files = sorted(input_dir.rglob("*.tif"))
```

5.4.1.7 int

```
convert_omx_to_paz.int
```

5.4.1.8 n_angles

```
convert_omx_to_paz.n_angles = args.num_angles
```

5.4.1.9 n_phases

```
convert_omx_to_paz.n_phases = args.num_phases
```

5.4.1.10 original

```
convert_omx_to_paz.original = tiffimage.imread(input_file)
```

5.4.1.11 parser

```
convert_omx_to_paz.parser = argparse.ArgumentParser()
```

5.4.1.12 required

```
convert_omx_to_paz.required
```

5.4.1.13 str

```
convert_omx_to_paz.str
```

5.4.1.14 type

```
convert_omx_to_paz.type
```


5.5 convert_slices_to_volumes Namespace Reference

Variables

- `parser` = `argparse.ArgumentParser()`
- `type`
- `str`
- `required`
- `tuple_of_ints`
- `default`
- `args` = `parser.parse_args()`
- `input_dir` = `pathlib.Path(args.input)`
- `output_dir` = `pathlib.Path(args.output)`
- `input_files` = `sorted(input_dir.glob("*.tif"))`
- `parents`
- `True`
- `exist_ok`
- `volume` = `np.zeros((len(input_files), 3061, 4096), dtype=np.uint8)`
- `input_slice` = `tifffile.imread(file)`
- `subvolume`
- tuple `output_file`
- `imagej`

5.5.1 Variable Documentation

5.5.1.1 args

```
convert_slices_to_volumes.args = parser.parse_args()
```

5.5.1.2 default

```
convert_slices_to_volumes.default
```

5.5.1.3 exist_ok

```
convert_slices_to_volumes.exist_ok
```

5.5.1.4 imagej

```
convert_slices_to_volumes.imagej
```

5.5.1.5 input_dir

```
convert_slices_to_volumes.input_dir = pathlib.Path(args.input)
```

5.5.1.6 input_files

```
convert_slices_to_volumes.input_files = sorted(input_dir.glob("*.tif"))
```

5.5.1.7 input_slice

```
convert_slices_to_volumes.input_slice = tiffiffle.imread(file)
```

5.5.1.8 output_dir

```
convert_slices_to_volumes.output_dir = pathlib.Path(args.output)
```

5.5.1.9 output_file

```
tuple convert_slices_to_volumes.output_file
```

Initial value:

```
1 = (  
2     output_dir / f"{args.label}_{i*args.num_steps[1] + j:03}.tif"  
3 )
```

5.5.1.10 parents

```
convert_slices_to_volumes.parents
```

5.5.1.11 parser

```
convert_slices_to_volumes.parser = argparse.ArgumentParser()
```

5.5.1.12 required

```
convert_slices_to_volumes.required
```

5.5.1.13 str

```
convert_slices_to_volumes.str
```

5.5.1.14 subvolume

```
convert_slices_to_volumes.subvolume
```

Initial value:

```
1 = volume[
2     :,
3     args.start[0]
4     + args.step[0] * i : args.start[0]
5     + args.step[0] * (i + 1),
6     args.start[1]
7     + args.step[1] * j : args.start[1]
8     + args.step[1] * (j + 1),
9 ]
```

5.5.1.15 True

```
convert_slices_to_volumes.True
```

5.5.1.16 tuple_of_ints

```
convert_slices_to_volumes.tuple_of_ints
```

5.5.1.17 type

```
convert_slices_to_volumes.type
```

5.5.1.18 volume

```
convert_slices_to_volumes.volume = np.zeros((len(input_files), 3061, 4096), dtype=np.uint8)
```

5.6 generate_sim Namespace Reference

Classes

- class [Simulator](#)
The [Simulator](#) class encapsulates the state of a 3D microscope simulation.
- class [SimulationRunner](#)
Class which performs a batch of simulations, either sequentially or in parallel.

Functions

- def [arange_zero](#) (n, spacing=1)
- def [threshold_norm](#) (sample)
Applies a threshold and normalises the sample to improve contrast.

Variables

- [parser](#) = argparse.ArgumentParser()
- [type](#)
- [str](#)
- [required](#)
- [int](#)
- [default](#)
- [args](#) = parser.parse_args()
- [runner](#)

5.6.1 Function Documentation

5.6.1.1 arange_zero()

```
def generate_sim.arange_zero (
    n,
    spacing = 1 )
```

5.6.1.2 threshold_norm()

```
def generate_sim.threshold_norm (
    sample )
```

Applies a threshold and normalises the sample to improve contrast.

5.6.2 Variable Documentation

5.6.2.1 args

```
generate_sim.args = parser.parse_args()
```

5.6.2.2 default

```
generate_sim.default
```

5.6.2.3 int

```
generate_sim.int
```

5.6.2.4 parser

```
generate_sim.parser = argparse.ArgumentParser()
```

5.6.2.5 required

```
generate_sim.required
```

5.6.2.6 runner

```
generate_sim.runner
```

Initial value:

```
1 = SimulationRunner(  
2     args.input, args.output, range(args.start, args.end), args.z_offset  
3 )
```

5.6.2.7 str

`generate_sim.str`

5.6.2.8 type

`generate_sim.type`

5.7 image_noising Namespace Reference

Functions

- def `save_image_pair` (gt_img, `split`, name, channel_idx)

Variables

- `parser` = argparse.ArgumentParser()
- `type`
- `str`
- `required`
- `int`
- `choices`
- `float`
- `default`
- `args` = parser.parse_args()
- `input_path` = pathlib.Path(args.input)
- `output_path` = pathlib.Path(args.output)
- `parents`
- `output_train_gt_path` = output_path.joinpath("Training", "GT")
- `output_train_raw_path` = output_path.joinpath("Training", "Raw")
- `output_val_gt_path` = output_path.joinpath("Validation", "GT")
- `output_val_raw_path` = output_path.joinpath("Validation", "Raw")
- `output_test_gt_path` = output_path.joinpath("Testing", "GT")
- `output_test_raw_path` = output_path.joinpath("Testing", "Raw")
- `data` = sorted(input_path.glob("*.tif"))
- `n_acquisitions` = tifffile.imread(data[0]).shape[0] // args.channels
- `n_img` = len(data)
- `train_size` = int((1 - args.test_fraction) * n_img)
- `val_size` = int(args.val_fraction * train_size)
- `rng` = np.random.default_rng(seed=25042024)
- `img_idx_all` = list(range(n_img))
- `img_idx_test` = img_idx_all[train_size:]
- `img_idx_train` = img_idx_all[: train_size - val_size]
- `img_idx_val` = img_idx_all[train_size - val_size : train_size]
- `gt` = tifffile.imread(img_file)
- string `split` = "train"

5.7.1 Function Documentation

5.7.1.1 save_image_pair()

```
def image_noising.save_image_pair (
    gt_img,
    split,
    name,
    channel_idx )
```

5.7.2 Variable Documentation

5.7.2.1 args

```
image_noising.args = parser.parse_args()
```

5.7.2.2 choices

```
image_noising.choices
```

5.7.2.3 data

```
list image_noising.data = sorted(input_path.glob("*.tif"))
```

5.7.2.4 default

```
image_noising.default
```

5.7.2.5 float

```
image_noising.float
```

5.7.2.6 gt

```
image_noising.gt = tifffile.imread(img_file)
```

5.7.2.7 img_idx_all

```
image_noising.img_idx_all = list(range(n_img))
```

5.7.2.8 img_idx_test

```
image_noising.img_idx_test = img_idx_all[train_size:]
```

5.7.2.9 img_idx_train

```
image_noising.img_idx_train = img_idx_all[: train_size - val_size]
```

5.7.2.10 img_idx_val

```
image_noising.img_idx_val = img_idx_all[train_size - val_size : train_size]
```

5.7.2.11 input_path

```
image_noising.input_path = pathlib.Path(args.input)
```

5.7.2.12 int

```
image_noising.int
```

5.7.2.13 n_acquisitions

```
image_noising.n_acquisitions = tifffile.imread(data[0]).shape[0] // args.channels
```


5.7.2.14 n_img

```
image_noising.n_img = len(data)
```

5.7.2.15 output_path

```
image_noising.output_path = pathlib.Path(args.output)
```

5.7.2.16 output_test_gt_path

```
image_noising.output_test_gt_path = output_path.joinpath("Testing", "GT")
```

5.7.2.17 output_test_raw_path

```
image_noising.output_test_raw_path = output_path.joinpath("Testing", "Raw")
```

5.7.2.18 output_train_gt_path

```
image_noising.output_train_gt_path = output_path.joinpath("Training", "GT")
```

5.7.2.19 output_train_raw_path

```
image_noising.output_train_raw_path = output_path.joinpath("Training", "Raw")
```

5.7.2.20 output_val_gt_path

```
image_noising.output_val_gt_path = output_path.joinpath("Validation", "GT")
```

5.7.2.21 output_val_raw_path

```
image_noising.output_val_raw_path = output_path.joinpath("Validation", "Raw")
```

5.7.2.22 parents

```
image_noising.parents
```

5.7.2.23 parser

```
image_noising.parser = argparse.ArgumentParser()
```

5.7.2.24 required

```
image_noising.required
```

5.7.2.25 rng

```
image_noising.rng = np.random.default_rng(seed=25042024)
```

5.7.2.26 split

```
string image_noising.split = "train"
```

5.7.2.27 str

```
image_noising.str
```

5.7.2.28 train_size

```
image_noising.train_size = int((1 - args.test_fraction) * n_img)
```

5.7.2.29 type

```
image_noising.type
```

5.7.2.30 val_size

```
image_noising.val_size = int(args.val_fraction * train_size)
```

5.8 manage_stack Namespace Reference

Variables

- `parser` = `argparse.ArgumentParser()`
- `type`
- `str`
- `required`
- `int`
- `choices`
- `default`
- `action`
- `args` = `parser.parse_args()`
- `output_dir` = `pathlib.Path(args.output_dir)`
- `parents`
- `True`
- `exist_ok`
- `files` = `sorted(list(pathlib.Path(args.input_dir).glob(args.glob_str)))`
- `int stack_number` = `-1` else `args.stack_number`
- `int number_of_stacks` = `len(files) // stack_number`
- `sample` = `tiffimage.imread(files[0])`
- `stack`
- `img_data` = `tiffimage.imread(input_file)`
- tuple `filename`
- tuple `output_file` = `output_dir / filename`
- `n_acq` = `args.num_acquisitions`
- `n_z` = `sample.shape[0] // n_acq`
- `output_data`

5.8.1 Variable Documentation

5.8.1.1 action

```
manage_stack.action
```

5.8.1.2 args

```
manage_stack.args = parser.parse_args()
```

5.8.1.3 choices

`manage_stack.choices`

5.8.1.4 default

`manage_stack.default`

5.8.1.5 exist_ok

`manage_stack.exist_ok`

5.8.1.6 filename

`tuple manage_stack.filename`

Initial value:

```
1 = (  
2     args.output_name  
3     + f"_stack{stack_idx*stack_number:04d}"  
4     + f"_{(stack_idx+1)*stack_number:04d}"  
5 )
```

5.8.1.7 files

`manage_stack.files = sorted(list(pathlib.Path(args.input_dir).glob(args.glob_str)))`

5.8.1.8 img_data

`manage_stack.img_data = tifffile.imread(input_file)`

5.8.1.9 int

`manage_stack.int`

5.8.1.10 n_acq

```
manage_stack.n_acq = args.num_acquisitions
```

5.8.1.11 n_z

```
manage_stack.n_z = sample.shape[0] // n_acq
```

5.8.1.12 number_of_stacks

```
int manage_stack.number_of_stacks = len(files) // stack_number
```

5.8.1.13 output_data

```
manage_stack.output_data
```

Initial value:

```
1 = img_data[
2         j * args.num_acquisitions : (j + 1) * args.num_acquisitions
3     ]
```

5.8.1.14 output_dir

```
manage_stack.output_dir = pathlib.Path(args.output_dir)
```

5.8.1.15 output_file

```
string manage_stack.output_file = output_dir / filename
```

5.8.1.16 parents

```
manage_stack.parents
```

5.8.1.17 parser

```
manage_stack.parser = argparse.ArgumentParser()
```

5.8.1.18 required

```
manage_stack.required
```

5.8.1.19 sample

```
manage_stack.sample = tiffimage.imread(files[0])
```

5.8.1.20 stack

```
manage_stack.stack
```

Initial value:

```
1 = np.zeros(
2     (
3         len(
4             files[
5                 stack_idx
6                 * stack_number : (stack_idx + 1)
7                 * stack_number
8             ]
9         ),
10        *sample.shape,
11    )
12 ).astype(sample.dtype)
```

5.8.1.21 stack_number

```
int manage_stack.stack_number = -1 else args.stack_number
```

5.8.1.22 str

```
manage_stack.str
```

5.8.1.23 True

`manage_stack.True`

5.8.1.24 type

`manage_stack.type`

5.9 rcan Namespace Reference

Namespaces

- [data_generator](#)
- [model](#)
- [plotting](#)
- [utils](#)

5.10 rcan.data_generator Namespace Reference

Classes

- class [SIM_Dataset](#)
Generates batches of images with real-time data augmentation.

Functions

- def [load_SIM_dataset](#) (images, shape, batch_size, transform_function, intensity_threshold, area_threshold, scale_factor, steps_per_epoch, p_min, p_max)
Wraps [SIM_Dataset](#) object in a PyTorch Dataloader object to enable batch loading.

5.10.1 Function Documentation

5.10.1.1 load_SIM_dataset()

```
def rcan.data_generator.load_SIM_dataset (
    images,
    shape,
    batch_size,
    transform_function,
    intensity_threshold,
    area_threshold,
    scale_factor,
    steps_per_epoch,
    p_min,
    p_max )
```

Wraps [SIM_Dataset](#) object in a PyTorch Dataloader object to enable batch loading.

Parameters

<i>images</i>	(list[dict]) - List of dictionaries of data pairs with keys ["raw","gt"]. Images in CZXY format
<i>shape</i>	(tuple[int]) - Shape of batch images excluding the channel dimension
<i>batch_size</i>	(int) - Batch size
<i>transform_function</i>	(str or callable or None) - Function used for data augmentation. Typically you will set <code>transform_function='rotate_and_flip'</code> to apply combination of randomly selected image rotation and flipping. Alternatively, you can specify an arbitrary transformation function which takes two input images (source and target) and returns transformed images. If <code>transform_function=None</code> , no augmentation will be performed
<i>intensity_threshold</i>	(float) - If <code>intensity_threshold > 0</code> , pixels whose intensities are greater than this threshold will be considered as foreground
<i>area_ratio_threshold</i>	(float) - Threshold between 0 and 1. If <code>intensity_threshold > 0</code> , the generator calculates the ratio of foreground pixels in a target patch, and rejects the patch if the ratio is smaller than this threshold
<i>scale_factor</i>	(int) - Scale factor for the target patch size. Positive and negative values mean up- and down-scaling respectively.
<i>steps_per_epoch</i>	(int) - Determines how many times each image is used to generate a patch per batch
<i>p_min</i>	(float) - Minimum percentile used for scaling
<i>p_max</i>	(float) - Maximum percentile used for scaling

Returns

`torch.utils.data.DataLoader` object

5.11 rcan.model Namespace Reference

Classes

- class [_channel_attention_block](#)
Implements channel attention block/layer.
- class [_residual_channel_attention_blocks](#)
Implements residual group based on [1].
- class [RCAN](#)
Builds a residual channel attention network.

Functions

- def [_conv](#) (ndim, in_filters, out_filters, kernel_size, padding="same", **kwargs)
Returns the appropriate torch.nn convolution layer based on parameters.
- def [_global_average_pooling](#) (ndim)
Returns the appropriate torch.nn pooling layer based on parameters.
- def [_standardize](#) (x)
Standardises input data.
- def [_destandardize](#) (x)
Inverse of _standardize.

5.11.1 Function Documentation

5.11.1.1 `_conv()`

```
def rcan.model._conv (
    ndim,
    in_filters,
    out_filters,
    kernel_size,
    padding = "same",
    ** kwargs ) [private]
```

Returns the appropriate torch.nn convolution layer based on parameters.

Parameters

<i>ndim</i>	(int) - Specifies a 1, 2, or 3 dimensional convolution kernel
<i>in_filters</i>	(int) - Number of hidden input channels
<i>out_filters</i>	(int) - Number of hidden output channels
<i>kernel_size</i>	(int or tuple) Size of convolution kernel
<i>padding</i>	(str, optional) - Border padding strategy. Default: "same"

Returns

torch.nn.Module object of the specified type

5.11.1.2 `_destandardize()`

```
def rcan.model._destandardize (
    x ) [private]
```

Inverse of `_standardize`.

Parameters

<i>x</i>	(torch.Tensor) Input
----------	----------------------

Returns

torch.Tensor representing destandardised output.

5.11.1.3 `_global_average_pooling()`

```
def rcan.model._global_average_pooling (
    ndim ) [private]
```

Returns the appropriate torch.nn pooling layer based on parameters.

Parameters

<code>ndim</code>	(int) - Specifies a 2 or 3 dimensional convolution kernel
-------------------	---

Returns

torch.nn.Module object of the specified type

5.11.1.4 `_standardize()`

```
def rcan.model._standardize (
    x ) [private]
```

Standardises input data.

Standardize the signal so that the range becomes [-1, 1] (assuming the original range is [0, 1]).

Parameters

<code>x</code>	(torch.Tensor) Input
----------------	----------------------

Returns

torch.Tensor representing standardised output

5.12 `rcan.plotting` Namespace Reference

Functions

- def [plot_learning_curve](#) (losses_train, losses_val, psnr_train, psnr_val, ssim_train, ssim_val, figsize, output_path)

Plots the learning curve metrics from a model checkpoint according to loss, PSNR, and SSIM.
- def [plot_reconstructions](#) (device, output_path, dim, gt_imgs, raw_imgs, model_1_imgs, model_2_imgs, imgs=None, cmap="inferno")

Plots a sample of reconstructions comparing GT vs Raw vs Restored.

5.12.1 Function Documentation

5.12.1.1 `plot_learning_curve()`

```
def rcan.plotting.plot_learning_curve (
    losses_train,
    losses_val,
    psnr_train,
    psnr_val,
    ssim_train,
    ssim_val,
    figsize,
    output_path )
```

Plots the learning curve metrics from a model checkpoint according to loss, PSNR, and SSIM.

Parameters

<i>losses_train</i>	(list[float]) - List of training losses
<i>losses_val</i>	(list[float]) - List of validation losses
<i>psnr_train</i>	(list[float]) - List of training psnrs
<i>psnr_val</i>	(list[float]) - List of validation psnrs
<i>ssim_train</i>	(list[float]) - List of training ssims
<i>ssim_val</i>	(list[float]) - List of validation ssims
<i>figsize</i>	(tuple[int]) - Specifies matplotlib layout size
<i>output_path</i>	(str) - Determines where plot is saved

5.12.1.2 `plot_reconstructions()`

```
def rcan.plotting.plot_reconstructions (
    device,
    output_path,
    dim,
    gt_imgs,
    raw_imgs,
    model_1_imgs,
    model_2_imgs = None,
    cmap = "inferno" )
```

Plots a sample of reconstructions comparing GT vs Raw vs Restored.

Parameters

<i>device</i>	(torch.device) - Handles the processing unit for torch
<i>output_path</i>	(str) - Determines where the plot is saved
<i>dim</i>	(int) - Dimensionality of the images
<i>gt_imgs</i>	(list[np.ndarray]) - List containing GT image arrays
<i>raw_imgs</i>	(list[np.ndarray]) - List containing Raw image arrays
<i>model_1_imgs</i>	(list[np.ndarray]) - List containing Step 1 image arrays
<i>model_2_imgs</i>	(list[np.ndarray], optional) - List containing Step 2 image arrays. Default: None
<i>cmap</i>	(str) - Matplotlib colormap string

5.13 rcan.utils Namespace Reference

Functions

- def `normalize` (image, p_min=2, p_max=99.9, dtype="float32")
Normalizes the image intensity so that the `p_min`-th and the `p_max`-th percentiles are converted to 0 and 1 respectively.
- def `apply` (model, data, model_input_image_shape, model_output_image_shape, num_input_channels, num_output_channels, batch_size, device, overlap_shape=None, verbose=False)
Applies a model to an input image.
- def `load_rcan_checkpoint` (ckpt_path, device)
Enables loading of RCAN checkpointed model.
- def `tuple_of_ints` (string)
Defines behaviour of parsing tuples of ints (argparse).
- def `percentile` (x)
Defines behaviour of parsing percentiles (argparse).

5.13.1 Function Documentation

5.13.1.1 `apply()`

```
def rcan.utils.apply (
    model,
    data,
    model_input_image_shape,
    model_output_image_shape,
    num_input_channels,
    num_output_channels,
    batch_size,
    device,
    overlap_shape = None,
    verbose = False )
```

Applies a model to an input image.

The input image stack is split into sub-blocks with model's input size, then the model is applied block by block.

Parameters

<i>model</i>	(torch.nn.module) - PyTorch model
<i>data</i>	(array_like or list of array_like) - Input data. Either an image or a list of images
<i>batch_size</i>	(int) - Controls the batch size used to process image data
<i>device</i>	(torch.device) - PyTorch device object to specify processor to use
<i>overlap_shape</i>	(tuple of int or None) - Overlap size between sub-blocks in each dimension. If not specified, a default size ((32, 32) for 2D and (2, 32, 32) for 3D) is used. Results at overlapped areas are blended together linearly

Returns

np.ndarray Result image

5.13.1.2 load_rcan_checkpoint()

```
def rcan.utils.load_rcan_checkpoint (
    ckpt_path,
    device )
```

Enables loading of RCAN checkpointed model.

Uses the `hyperparameters` key saved in checkpoint file in order to avoid the need to know the architecture specifications in advance.

Parameters

<i>ckpt_path</i>	(str) - filepath for checkpoint, should end in .pth
<i>device</i>	(torch.device) - handles processing unit for torch

Returns

tuple of checkpoint, and model with weights loaded

5.13.1.3 normalize()

```
def rcan.utils.normalize (
    image,
    p_min = 2,
    p_max = 99.9,
    dtype = "float32" )
```

Normalizes the image intensity so that the `p_min`-th and the `p_max`-th percentiles are converted to 0 and 1 respectively.

Parameters

<i>image</i>	(np.ndarray) - Image to apply the normalization to
<i>p_min</i>	(float, optional) - Percentile that is mapped to zero. Default: 2
<i>p_max</i>	(float, optional) - Percentile that is mapped to one. Default: 99.9
<i>dtype</i>	(str) - Datatype to use for the output

Returns

np.ndarray Image with transformed pixel values

5.13.1.4 References

Content-Aware Image Restoration: Pushing the Limits of Fluorescence Microscopy <https://doi.org/10.1038/s41592-018-0216-7>

5.13.1.5 percentile()

```
def rcan.utils.percentile (
    x )
```

Defines behaviour of parsing percentiles (argparse).

5.13.1.6 tuple_of_ints()

```
def rcan.utils.tuple_of_ints (
    string )
```

Defines behaviour of parsing tuples of ints (argparse).

5.14 recon_postprocess Namespace Reference

Variables

- `parser` = `argparse.ArgumentParser()`
- `type`
- `str`
- `required`
- `args` = `parser.parse_args()`
- `files` = `sorted(list(pathlib.Path(args.input_dir).rglob("*.tif")))`
- `img_data` = `tiffimage.imread(input_file)`

5.14.1 Variable Documentation

5.14.1.1 args

```
recon_postprocess.args = parser.parse_args()
```

5.14.1.2 files

```
recon_postprocess.files = sorted(list(pathlib.Path(args.input_dir).rglob("*.tif")))
```

5.14.1.3 img_data

```
tuple recon_postprocess.img_data = tifffile.imread(input_file)
```

5.14.1.4 parser

```
recon_postprocess.parser = argparse.ArgumentParser()
```

5.14.1.5 required

```
recon_postprocess.required
```

5.14.1.6 str

```
recon_postprocess.str
```

5.14.1.7 type

```
recon_postprocess.type
```

5.15 recon_preprocess Namespace Reference

Functions

- def [normalize_acquisition_intensity](#) (data, dim)

Variables

- [parser](#) = argparse.ArgumentParser()
- [type](#)
- [str](#)
- [required](#)
- [int](#)
- [choices](#)
- [percentile](#)
- [default](#)
- [action](#)
- [args](#) = parser.parse_args()
- [output_dir](#) = pathlib.Path(args.output_dir)
- [parents](#)
- [True](#)
- [exist_ok](#)
- [files](#) = sorted(list(pathlib.Path(args.input_dir).glob("*.tif")))
- [img_data](#) = tifffile.imread(input_file).astype("float32")
- [output_file](#) = [output_dir](#) / input_file.name

5.15.1 Function Documentation

5.15.1.1 `normalize_acquisition_intensity()`

```
def recon_preprocess.normalize_acquisition_intensity (
    data,
    dim )
```

5.15.2 Variable Documentation

5.15.2.1 `action`

```
recon_preprocess.action
```

5.15.2.2 `args`

```
recon_preprocess.args = parser.parse_args()
```

5.15.2.3 `choices`

```
recon_preprocess.choices
```

5.15.2.4 `default`

```
recon_preprocess.default
```

5.15.2.5 `exist_ok`

```
recon_preprocess.exist_ok
```


5.15.2.6 files

```
recon_preprocess.files = sorted(list(pathlib.Path(args.input_dir).glob("*.tif")))
```

5.15.2.7 img_data

```
int recon_preprocess.img_data = tiffimage.imread(input_file).astype("float32")
```

5.15.2.8 int

```
recon_preprocess.int
```

5.15.2.9 output_dir

```
recon_preprocess.output_dir = pathlib.Path(args.output_dir)
```

5.15.2.10 output_file

```
recon_preprocess.output_file = output_dir / input_file.name
```

5.15.2.11 parents

```
recon_preprocess.parents
```

5.15.2.12 parser

```
recon_preprocess.parser = argparse.ArgumentParser()
```

5.15.2.13 percentile

```
recon_preprocess.percentile
```

5.15.2.14 required

`recon_preprocess.required`

5.15.2.15 str

`recon_preprocess.str`

5.15.2.16 True

`recon_preprocess.True`

5.15.2.17 type

`recon_preprocess.type`

5.16 stats Namespace Reference

Functions

- def [paired_t](#)(gt_data, [data](#))

Variables

- [parser](#) = argparse.ArgumentParser()
- [type](#)
- [str](#)
- [required](#)
- [int](#)
- [choices](#)
- [default](#)
- [args](#) = parser.parse_args()
- [output_dir](#) = pathlib.Path(args.output_dir)
- [parents](#)
- [True](#)
- [exist_ok](#)
- [df](#)
- [fig](#)
- [ax](#)
- [figsize](#)
- [psnr_diff_1_max](#)

- [psnr_diff_2_max](#)
- [psnr_diff_1_min](#)
- [psnr_diff_2_min](#)
- [tuple hist_range_psnr](#)
- [ssim_diff_1_max](#)
- [ssim_diff_2_max](#)
- [ssim_diff_1_min](#)
- [ssim_diff_2_min](#)
- [tuple hist_range_ssim](#)
- [xlabel](#)
- [title](#)
- [range](#)
- [color](#)
- [mean_psnr_1](#) = np.mean(np.array(df['psnr_model_1']) - np.array(df['psnr_raw']))
- [se_psnr_1](#)
- [mean_ssim_1](#) = np.mean(np.array(df['ssim_model_1']) - np.array(df['ssim_raw']))
- [se_ssim_1](#)
- [mean_psnr_2](#)
- [se_psnr_2](#)
- [mean_ssim_2](#)
- [se_ssim_2](#)
- [int psnr_cols](#) = 2 else df.columns[1:3]
- [int ssim_cols](#) = 2 else df.columns[3:5]
- [dflong](#)
- [dflongssim](#)
- [data](#)
- [x](#)
- [y](#)
- [hue](#)
- [dodge](#)
- [legend](#)
- [palette](#)
- [alpha](#)
- [lw](#)

5.16.1 Function Documentation

5.16.1.1 `paired_t()`

```
def stats.paired_t (  
    gt_data,  
    data )
```

5.16.2 Variable Documentation

5.16.2.1 alpha

`stats.alpha`

5.16.2.2 args

`stats.args = parser.parse_args()`

5.16.2.3 ax

`stats.ax`

5.16.2.4 choices

`stats.choices`

5.16.2.5 color

`stats.color`

5.16.2.6 data

`stats.data`

5.16.2.7 default

`stats.default`

5.16.2.8 df

stats.df

Initial value:

```
1 = pd.read_csv(  
2     pathlib.Path(args.dataset),  
3     index_col=False  
4 ).drop(columns="Unnamed: 0")
```

5.16.2.9 dflong

stats.dflong

Initial value:

```
1 = pd.melt(  
2     df,  
3     id_vars=["file"],  
4     value_vars=df.columns[1:4],  
5     var_name="type",  
6     value_name="psnr"  
7 )
```

5.16.2.10 dflongssim

stats.dflongssim

Initial value:

```
1 = pd.melt(  
2     df,  
3     id_vars=["file"],  
4     value_vars=df.columns[4:7],  
5     var_name="type",  
6     value_name="ssim"  
7 )
```

5.16.2.11 dodge

stats.dodge

5.16.2.12 exist_ok

stats.exist_ok

5.16.2.13 fig

`stats.fig`

5.16.2.14 figsize

`stats.figsize`

5.16.2.15 hist_range_psnr

`tuple stats.hist_range_psnr`

Initial value:

```
1 = (  
2     min(psnr_diff_1_min, psnr_diff_2_min),  
3     max(psnr_diff_1_max, psnr_diff_2_max)  
4 )
```

5.16.2.16 hist_range_ssim

`tuple stats.hist_range_ssim`

Initial value:

```
1 = (  
2     min(ssim_diff_1_min, ssim_diff_2_min),  
3     max(ssim_diff_1_max, ssim_diff_2_max)  
4 )
```

5.16.2.17 hue

`stats.hue`

5.16.2.18 int

`stats.int`

5.16.2.19 legend

```
stats.legend
```

5.16.2.20 lw

```
stats.lw
```

5.16.2.21 mean_psnr_1

```
stats.mean_psnr_1 = np.mean(np.array(df['psnr_model_1']) - np.array(df['psnr_raw']))
```

5.16.2.22 mean_psnr_2

```
stats.mean_psnr_2
```

Initial value:

```
1 = np.mean(  
2     np.array(df['psnr_model_2']) - np.array(df['psnr_raw'])  
3 )
```

5.16.2.23 mean_ssim_1

```
stats.mean_ssim_1 = np.mean(np.array(df['ssim_model_1']) - np.array(df['ssim_raw']))
```

5.16.2.24 mean_ssim_2

```
stats.mean_ssim_2
```

Initial value:

```
1 = np.mean(  
2     np.array(df['ssim_model_2']) - np.array(df['ssim_raw'])  
3 )
```

5.16.2.25 output_dir

```
stats.output_dir = pathlib.Path(args.output_dir)
```

5.16.2.26 palette

```
stats.palette
```

5.16.2.27 parents

```
stats.parents
```

5.16.2.28 parser

```
stats.parser = argparse.ArgumentParser()
```

5.16.2.29 psnr_cols

```
int stats.psnr_cols = 2 else df.columns[1:3]
```

5.16.2.30 psnr_diff_1_max

```
stats.psnr_diff_1_max
```

Initial value:

```
1 = np.max(  
2     np.array(df["psnr_model_1"]) - np.array(df["psnr_raw"])  
3 )
```

5.16.2.31 psnr_diff_1_min

```
stats.psnr_diff_1_min
```

Initial value:

```
1 = np.min(  
2     np.array(df["psnr_model_1"]) - np.array(df["psnr_raw"])  
3 )
```


5.16.2.32 psnr_diff_2_max

stats.psnr_diff_2_max

Initial value:

```
1 = np.max(  
2     np.array(df["psnr_model_2"]) - np.array(df["psnr_raw"])  
3 )
```

5.16.2.33 psnr_diff_2_min

stats.psnr_diff_2_min

Initial value:

```
1 = np.min(  
2     np.array(df["psnr_model_2"]) - np.array(df["psnr_raw"])  
3 )
```

5.16.2.34 range

stats.range

5.16.2.35 required

stats.required

5.16.2.36 se_psnr_1

stats.se_psnr_1

Initial value:

```
1 = np.std(  
2     np.array(df['psnr_model_1']) - np.array(df['psnr_raw']), ddof=1  
3 )/np.sqrt(len(df['psnr_model_1']))
```

5.16.2.37 se_psnr_2

stats.se_psnr_2

Initial value:

```
1 = np.std(  
2     np.array(df['psnr_model_2']) - np.array(df['psnr_raw']), ddof=1  
3 )/np.sqrt(len(df['psnr_model_2']))
```

5.16.2.38 se_ssim_1

stats.se_ssim_1

Initial value:

```
1 = np.std(  
2     np.array(df['ssim_model_1']) - np.array(df['ssim_raw']), ddof=1  
3 )/np.sqrt(len(df['ssim_model_1']))
```

5.16.2.39 se_ssim_2

stats.se_ssim_2

Initial value:

```
1 = np.std(  
2     np.array(df['ssim_model_2']) - np.array(df['ssim_raw']), ddof=1  
3     )/np.sqrt(len(df['ssim_model_2']))
```

5.16.2.40 ssim_cols

```
int stats.ssim_cols = 2 else df.columns[3:5]
```

5.16.2.41 ssim_diff_1_max

stats.ssim_diff_1_max

Initial value:

```
1 = np.max(  
2     np.array(df["ssim_model_1"]) - np.array(df["ssim_raw"])  
3 )
```

5.16.2.42 ssim_diff_1_min

stats.ssim_diff_1_min

Initial value:

```
1 = np.min(  
2     np.array(df["ssim_model_1"]) - np.array(df["ssim_raw"])  
3 )
```

5.16.2.43 ssim_diff_2_max

`stats.ssim_diff_2_max`

Initial value:

```
1 = np.max(  
2     np.array(df["ssim_model_2"]) - np.array(df["ssim_raw"])  
3 )
```

5.16.2.44 ssim_diff_2_min

`stats.ssim_diff_2_min`

Initial value:

```
1 = np.min(  
2     np.array(df["ssim_model_2"]) - np.array(df["ssim_raw"])  
3 )
```

5.16.2.45 str

`stats.str`

5.16.2.46 title

`stats.title`

5.16.2.47 True

`stats.True`

5.16.2.48 type

`stats.type`

5.16.2.49 x

`stats.x`

5.16.2.50 xlabel

`stats.xlabel`

5.16.2.51 y

`stats.y`

5.17 synthetic_sim Namespace Reference

Namespaces

- [otf](#)

5.18 synthetic_sim.otf Namespace Reference

Classes

- class [PsfParameters](#)
Class to store PSF parameters.

Functions

- def [calc_psf](#) (params)
Calculate an approximate Gibson-Lanni PSF based on the parameters provided.

5.18.1 Function Documentation

5.18.1.1 calc_psf()

```
def synthetic_sim.otf.calc_psf (
    params )
```

Calculate an approximate Gibson-Lanni PSF based on the parameters provided.

Code ported from MATLAB, original copyright Jizhou Li, 2016, The Chinese University of Hong Kong.

Parameters

<i>params</i>	(PsfParameters) - dataclass storing the PSF parameters
---------------	--

Returns

psf (np.ndarray) - array representing the PSF

5.19 train Namespace Reference

Functions

- def `load_data_paths` (`config`, `data_type`)
- def `train` (`train_loader`, `val_loader`, `optimizer`, `scheduler`, `net`, `batchsize`, `n_accumulations`, `saveinterval`, `nepoch`, `start_epoch=0`, `losses_train_epoch=[]`, `losses_val_epoch=[]`, `psnr_train_epoch=[]`, `psnr_val_epoch=[]`, `ssim_train_epoch=[]`, `ssim_val_epoch=[]`)

Variables

- `parser` = `argparse.ArgumentParser()`
- `type`
- `str`
- `required`
- `args` = `parser.parse_args()`
- dictionary `schema`
- `config` = `json.load(f)`
- int `ndim` = `tiffimage.imread(training_data[0]["raw"]).ndim - 1`
- `input_shape` = `config["input_shape"]`
- tuple `device`
- `ckpt_path` = `None` if `args.model_ckpt` is `None` else `pathlib.Path(args.model_ckpt)`
- `model`
- dictionary `RCAN_hyperparameters`
- `ckpt`
- `train_loader`
- `val_loader`
- `optimizer`
- `scheduler`
- `output_dir` = `pathlib.Path(args.output_dir)`
- `parents`
- `True`
- `exist_ok`
- `n_accumulations`
- `saveinterval`
- `nepoch`
- `start_epoch`
- `losses_train_epoch`
- `losses_val_epoch`
- `psnr_train_epoch`
- `psnr_val_epoch`
- `ssim_train_epoch`
- `ssim_val_epoch`

5.19.1 Function Documentation

5.19.1.1 load_data_paths()

```
def train.load_data_paths (
    config,
    data_type )
```

5.19.1.2 train()

```
def train.train (
    train_loader,
    val_loader,
    optimizer,
    scheduler,
    net,
    batchsize,
    n_accumulations,
    saveinterval,
    nepoch,
    start_epoch = 0,
    losses_train_epoch = [],
    losses_val_epoch = [],
    psnr_train_epoch = [],
    psnr_val_epoch = [],
    ssim_train_epoch = [],
    ssim_val_epoch = [] )
```

5.19.2 Variable Documentation

5.19.2.1 args

```
train.args = parser.parse_args()
```

5.19.2.2 ckpt

```
train.ckpt
```

5.19.2.3 ckpt_path

```
train.ckpt_path = None if args.model_ckpt is None else pathlib.Path(args.model_ckpt)
```

5.19.2.4 config

```
train.config = json.load(f)
```

5.19.2.5 device

```
tuple train.device
```

Initial value:

```
1 = (  
2     torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")  
3 )
```

5.19.2.6 exist_ok

```
train.exist_ok
```

5.19.2.7 input_shape

```
tuple train.input_shape = config["input_shape"]
```

5.19.2.8 losses_train_epoch

```
train.losses_train_epoch
```

5.19.2.9 losses_val_epoch

```
train.losses_val_epoch
```

5.19.2.10 model

```
train.model
```

Initial value:

```
1 = RCAN(  
2     input_shape,  
3     num_input_channels=config["num_input_channels"],  
4     num_hidden_channels=config["num_hidden_channels"],  
5     num_residual_blocks=config["num_residual_blocks"],  
6     num_residual_groups=config["num_residual_groups"],  
7     channel_reduction=config["channel_reduction"],  
8     residual_scaling=1.0,  
9     num_output_channels=config["num_output_channels"],  
10 )
```

5.19.2.11 n_accumulations

```
train.n_accumulations
```

5.19.2.12 ndim

```
int train.ndim = tiffimage.imread(training_data[0]["raw"]).ndim - 1
```

5.19.2.13 nepoch

```
train.nepoch
```

5.19.2.14 optimizer

```
train.optimizer
```

Initial value:

```
1 = torch.optim.Adam(  
2     model.parameters(), lr=config["initial_learning_rate"]  
3 )
```

5.19.2.15 output_dir

```
train.output_dir = pathlib.Path(args.output_dir)
```

5.19.2.16 parents

```
train.parents
```

5.19.2.17 parser

```
train.parser = argparse.ArgumentParser()
```


5.19.2.18 psnr_train_epoch

```
train.psnr_train_epoch
```

5.19.2.19 psnr_val_epoch

```
train.psnr_val_epoch
```

5.19.2.20 RCAN_hyperparameters

```
train.RCAN_hyperparameters
```

Initial value:

```
1 = {
2     "input_shape": input_shape,
3     "num_input_channels": config["num_input_channels"],
4     "num_hidden_channels": config["num_hidden_channels"],
5     "num_residual_blocks": config["num_residual_blocks"],
6     "num_residual_groups": config["num_residual_groups"],
7     "channel_reduction": config["channel_reduction"],
8     "residual_scaling": 1.0,
9     "num_output_channels": config["num_output_channels"],
10 }
```

5.19.2.21 required

```
train.required
```

5.19.2.22 saveinterval

```
train.saveinterval
```

5.19.2.23 scheduler

```
train.scheduler
```

Initial value:

```
1 = torch.optim.lr_scheduler.StepLR(
2     optimizer, step_size=config["epochs"] // 4, gamma=config["lr_decay"]
3 )
```

5.19.2.24 schema

dictionary train.schema

5.19.2.25 ssim_train_epoch

train.ssim_train_epoch

5.19.2.26 ssim_val_epoch

train.ssim_val_epoch

5.19.2.27 start_epoch

train.start_epoch

5.19.2.28 str

train.str

5.19.2.29 train_loader

train.train_loader

Initial value:

```
1 = load_SIM_dataset (
2     training_data,
3     input_shape,
4     batch_size=config["batch_size"],
5     transform_function=(
6         "rotate_and_flip" if config["data_augmentation"] else None
7     ),
8     intensity_threshold=config["intensity_threshold"],
9     area_threshold=config["area_ratio_threshold"],
10    scale_factor=1,
11    steps_per_epoch=config["steps_per_epoch"],
12    p_min=config["p_min"],
13    p_max=config["p_max"],
14 )
```

5.19.2.30 True

`train.True`

5.19.2.31 type

`train.type`

5.19.2.32 val_loader

`train.val_loader`

Initial value:

```
1 = load_SIM_dataset(  
2     validation_data,  
3     input_shape,  
4     batch_size=config["batch_size"],  
5     transform_function=(  
6         "rotate_and_flip" if config["data_augmentation"] else None  
7     ),  
8     intensity_threshold=config["intensity_threshold"],  
9     area_threshold=config["area_ratio_threshold"],  
10    scale_factor=1,  
11    steps_per_epoch=config["steps_per_epoch"],  
12    p_min=config["p_min"],  
13    p_max=config["p_max"],  
14 )
```

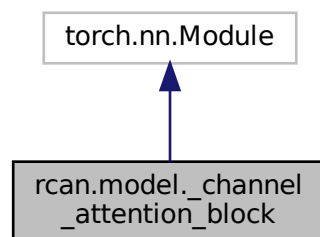

Chapter 6

Class Documentation

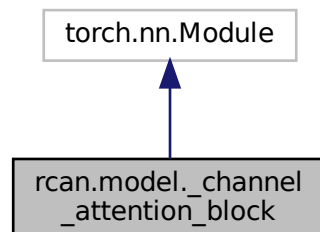
6.1 rcan.model._channel_attention_block Class Reference

Implements channel attention block/layer.

Inheritance diagram for rcan.model._channel_attention_block:



Collaboration diagram for rcan.model._channel_attention_block:



Public Member Functions

- def `__init__` (self, ndim, num_channels, reduction=16)
Initialises class.
- def `forward` (self, x)
Forward method for class.

Public Attributes

- `global_average_pooling`
- `conv_1`
- `conv_2`

6.1.1 Detailed Description

Implements channel attention block/layer.

Instantiates a simple attention mechanism which pools all spatial information in each channel, and computes channel attention weights through a series of linear transformations and activation layers. Builds part of the architecture originally presented in [1]. Software implementation based on [2].

6.1.1.1 References

[1] Image Super-Resolution Using Very Deep Residual Channel Attention Networks <https://arxiv.org/abs/1807.02758> [2] Fast, multicolour optical sectioning over extended fields of view by combining interferometric SIM with machine learning <https://doi.org/10.1364/BOE.510912> (Implementation based on CAlayer from the paper's source code: <https://github.com/edward-n-ward/ML-OS-SIM/blob/master/RCAN/Training%20code/models.py>)

6.1.2 Constructor & Destructor Documentation

6.1.2.1 `__init__()`

```
def rcnn.model._channel_attention_block.__init__ (
    self,
    ndim,
    num_channels,
    reduction = 16 )
```

Initialises class.

Parameters

<code>ndim</code>	(int) - Feature dimensionality
<code>num_channels</code>	(int) - Number of hidden channels
<code>reduction</code>	(int, optional) - Factor to reduce the number of channels by during the attention weight computation. Default: 16.

6.1.3 Member Function Documentation

6.1.3.1 forward()

```
def rcan.model._channel_attention_block.forward (
    self,
    x )
```

Forward method for class.

Parameters

<i>x</i>	(torch.Tensor) Input
----------	----------------------

Returns

torch.Tensor representing *x* multiplied by attention weights across channels.

6.1.4 Member Data Documentation

6.1.4.1 conv_1

```
rcan.model._channel_attention_block.conv_1
```

6.1.4.2 conv_2

```
rcan.model._channel_attention_block.conv_2
```

6.1.4.3 global_average_pooling

```
rcan.model._channel_attention_block.global_average_pooling
```

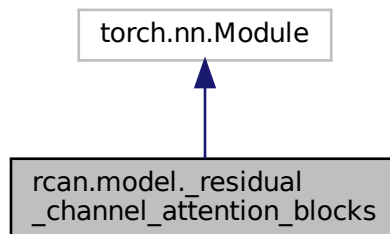
The documentation for this class was generated from the following file:

- /home/jhughes2712/projects/sim_project/jh2284/src/rcan/model.py

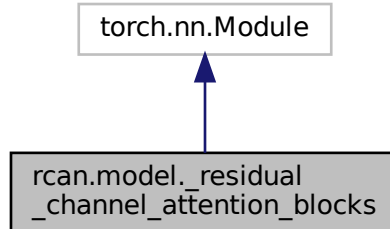
6.2 rcan.model._residual_channel_attention_blocks Class Reference

Implements residual group based on [1].

Inheritance diagram for rcan.model._residual_channel_attention_blocks:



Collaboration diagram for rcan.model._residual_channel_attention_blocks:



Public Member Functions

- def `__init__` (self, ndim, num_channels, `repeat`=1, channel_reduction=8, `residual_scaling`=1.0)
Initialises object.
- def `forward` (self, x)
Forward method for class.

Public Attributes

- `repeat`
- `residual_scaling`
- `conv_list`
- `channel_attention_block_list`

6.2.1 Detailed Description

Implements residual group based on [1].

6.2.1.1 References

[1] Fast, multicolour optical sectioning over extended fields of view by combining interferometric SIM with machine learning <https://doi.org/10.1364/BOE.510912> (Implementation based on ResidualGroup from the paper's source code: <https://github.com/edward-n-ward/ML-OS-SIM/blob/master/RCAN/Training%20code/models.py>)

6.2.2 Constructor & Destructor Documentation

6.2.2.1 __init__()

```
def rcan.model._residual_channel_attention_blocks.__init__ (
    self,
    ndim,
    num_channels,
    repeat = 1,
    channel_reduction = 8,
    residual_scaling = 1.0 )
```

Initialises object.

Parameters

<i>ndim</i>	(int) - Spatial dimension of input features
<i>num_channels</i>	(int) - Number of hidden channels
<i>repeat</i>	(int) - Number of residual blocks in group
<i>channel_reduction</i>	(int) - Channel reduction during attention mechanism
<i>residual_scaling</i>	(float) - output multiplier before residual connection

6.2.3 Member Function Documentation

6.2.3.1 forward()

```
def rcan.model._residual_channel_attention_blocks.forward (
    self,
    x )
```

Forward method for class.

Parameters

x	(torch.Tensor) - Input values
-----	-------------------------------

Returns

torch.Tensor representing output values

6.2.4 Member Data Documentation

6.2.4.1 channel_attention_block_list

```
rcan.model._residual_channel_attention_blocks.channel_attention_block_list
```

6.2.4.2 conv_list

```
rcan.model._residual_channel_attention_blocks.conv_list
```

6.2.4.3 repeat

```
rcan.model._residual_channel_attention_blocks.repeat
```

6.2.4.4 residual_scaling

```
rcan.model._residual_channel_attention_blocks.residual_scaling
```

The documentation for this class was generated from the following file:

- /home/jhughes2712/projects/sim_project/jh2284/src/rcan/[model.py](#)

6.3 synthetic_sim.otf.PsfParameters Class Reference

Class to store PSF parameters.

Static Public Attributes

- [int](#)
- [float](#)
- [Callable](#)

6.3.1 Detailed Description

Class to store PSF parameters.

```
@details Class to store the parameters used to evaluate an approximate  
Gibson-Lanni PSF. Default values are provided except for the PSF size.
```

6.3.2 Member Data Documentation

6.3.2.1 Callable

```
synthetic_sim.otf.PsfParameters.Callable [static]
```

6.3.2.2 float

```
synthetic_sim.otf.PsfParameters.float [static]
```

6.3.2.3 int

```
synthetic_sim.otf.PsfParameters.int [static]
```

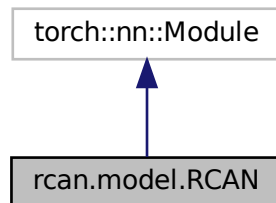
The documentation for this class was generated from the following file:

- [/home/jhughes2712/projects/sim_project/jh2284/src/synthetic_sim/otf.py](#)

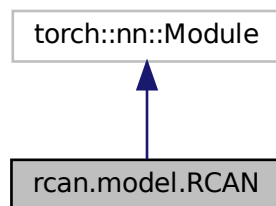
6.4 rcan.model.RCAN Class Reference

Builds a residual channel attention network.

Inheritance diagram for rcan.model.RCAN:



Collaboration diagram for rcan.model.RCAN:



Public Member Functions

- `def __init__ (self, input_shape=(16, 256, 256), *num_input_channels=9, num_hidden_channels=32, num_residual_blocks=3, num_residual_groups=5, channel_reduction=8, residual_scaling=1.0, num_output_channels=-1)`
Initialises object.
- `def forward (self, x)`
Forward method for class.

Public Attributes

- `num_residual_groups`
- `rcab_list`
- `conv_input`
- `conv_list`
- `conv_output`

6.4.1 Detailed Description

Builds a residual channel attention network.

Note that the upscale module at the end of the network is omitted so that the input and output of the model have the same size.

6.4.1.1 References

[1] Image Super-Resolution Using Very Deep Residual Channel Attention Networks <https://arxiv.org/abs/1807.02758> [2] Fast, multicolour optical sectioning over extended fields of view by combining interferometric SIM with machine learning <https://doi.org/10.1364/BOE.510912> (Implementation based on RCAN from the paper's source code: <https://github.com/edward-n-ward/ML-OS-SIM/blob/master/RCAN/Training%20code/models.py>)

6.4.2 Constructor & Destructor Documentation

6.4.2.1 __init__()

```
def rcan.model.RCAN.__init__(
    self,
    input_shape = (16, 256, 256),
    * num_input_channels = 9,
    num_hidden_channels = 32,
    num_residual_blocks = 3,
    num_residual_groups = 5,
    channel_reduction = 8,
    residual_scaling = 1.0,
    num_output_channels = -1 )
```

Initialises object.

Builds a residual channel attention network. Note that the upscale module at the end of the network is omitted so that the input and output of the model have the same size.

Parameters

<i>input_shape</i>	(tuple[int]) - Input shape of the model.
<i>num_channels</i>	(int) - Number of feature channels.
<i>num_residual_blocks</i>	(int) - Number of residual channel attention blocks in each residual group.
<i>num_residual_groups</i>	(int) - Number of residual groups.
<i>channel_reduction</i>	(int) - Channel reduction ratio for channel attention.
<i>residual_scaling</i>	(float) - Scaling factor applied to the residual component in the residual channel attention block.
<i>num_output_channels</i>	(int) - Number of channels in the output image. if negative, it is set to the same number as the input.

Returns

`torch.nn.Module` PyTorch model instance.

6.4.3 Member Function Documentation

6.4.3.1 `forward()`

```
def rcan.model.RCAN.forward (
    self,
    x )
```

Forward method for class.

Parameters

<code>x</code>	(<code>torch.Tensor</code>) - Input
----------------	---------------------------------------

Returns

`torch.Tensor` Output

6.4.4 Member Data Documentation

6.4.4.1 `conv_input`

`rcan.model.RCAN.conv_input`

6.4.4.2 `conv_list`

`rcan.model.RCAN.conv_list`

6.4.4.3 `conv_output`

`rcan.model.RCAN.conv_output`

6.4.4.4 num_residual_groups

`rcan.model.RCAN.num_residual_groups`

6.4.4.5 rcab_list

`rcan.model.RCAN.rcab_list`

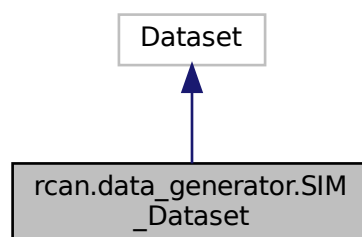
The documentation for this class was generated from the following file:

- `/home/jhughes2712/projects/sim_project/jh2284/src/rcan/model.py`

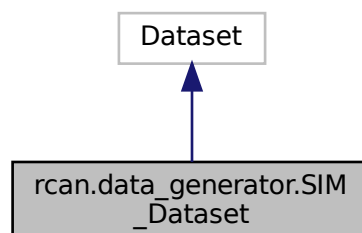
6.5 rcan.data_generator.SIM_Dataset Class Reference

Generates batches of images with real-time data augmentation.

Inheritance diagram for `rcan.data_generator.SIM_Dataset`:



Collaboration diagram for `rcan.data_generator.SIM_Dataset`:



Public Member Functions

- `def __init__ (self, images, shape, transform_function="rotate_and_flip", intensity_threshold=0.0, area_ratio_threshold=0.0, scale_factor=1, steps_per_epoch=1, p_min=2.0, p_max=99.9)`
Initialises object.
- `def __getitem__ (self, j)`
Method used during batch loading.
- `def __len__ (self)`

Public Attributes

- `steps_per_epoch`
- `p_min`
- `p_max`
- `output_shape`
- `output_signature`

Private Member Functions

- `def _scale (self, shape)`

Private Attributes

- `_shape`
- `_transform_function`
- `_intensity_threshold`
- `_area_threshold`
- `_scale_factor`
- `_y`

6.5.1 Detailed Description

Generates batches of images with real-time data augmentation.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 __init__()

```
def rcan.data_generator.SIM_Dataset.__init__ (
    self,
    images,
    shape,
    transform_function = "rotate_and_flip",
    intensity_threshold = 0.0,
    area_ratio_threshold = 0.0,
    scale_factor = 1,
    steps_per_epoch = 1,
    p_min = 2.0,
    p_max = 99.9 )
```

Initialises object.

Parameters

<i>images</i>	(list[dict]) - List of dictionaries of data pairs with keys ["raw","gt"]. Images in CZXY format
<i>shape</i>	(tuple[int]) - Shape of batch images excluding the channel dimension
<i>transform_function</i>	(str or callable, optional) - Function used for data augmentation. Typically you will set <code>transform_function='rotate_and_flip'</code> to apply combination of randomly selected image rotation and flipping. Alternatively, you can specify an arbitrary transformation function which takes two input images (source and target) and returns transformed images. If <code>transform_function=None</code> , no augmentation will be performed. Default: "rotate_and_flip"
<i>intensity_threshold</i>	(float, optional) - If <code>intensity_threshold > 0</code> , pixels whose intensities are greater than this threshold will be considered as foreground. Default: 0.0
<i>area_ratio_threshold</i>	(float, optional) - Threshold between 0 and 1. If <code>intensity_threshold > 0</code> , the generator calculates the ratio of foreground pixels in a target patch, and rejects the patch if the ratio is smaller than this threshold. Default: 0.0
<i>scale_factor</i>	(int, optional) - Scale factor for the target patch size. Positive and negative values mean up- and down-scaling respectively. Default: 1
<i>steps_per_epoch</i>	(int, optional) - Determines how many times each image is used to generate a patch per batch. Default: 1
<i>p_min</i>	(float, optional) - Minimum percentile used for scaling. Default: 2.0
<i>p_max</i>	(float, optional) - Maximum percentile used for scaling. Default: 99.9

6.5.3 Member Function Documentation

6.5.3.1 `__getitem__()`

```
def rcan.data_generator.SIM_Dataset.__getitem__ (
    self,
    j )
```

Method used during batch loading.

Standardises pixel values and takes patches from the image pair. Also implements the rejection of patches based on area/intensity threshold, if `self._intensity_threshold > 0`. Augments data pair.

Parameters

<i>j</i>	(int) - Index of data to be loaded. Note that if <code>self.steps_per_epoch > 1</code> , this can be more than the dataset size, in which case it is interpreted modulo the dataset size.
----------	--

Returns

tuple(torch.Tensor) raw-gt image pair

6.5.3.2 `__len__()`

```
def rcan.data_generator.SIM_Dataset.__len__ (
    self )
```

6.5.3.3 `_scale()`

```
def rcan.data_generator.SIM_Dataset._scale (
    self,
    shape ) [private]
```

6.5.4 Member Data Documentation

6.5.4.1 `_area_threshold`

```
rcan.data_generator.SIM_Dataset._area_threshold [private]
```

6.5.4.2 `_intensity_threshold`

```
rcan.data_generator.SIM_Dataset._intensity_threshold [private]
```

6.5.4.3 `_scale_factor`

```
rcan.data_generator.SIM_Dataset._scale_factor [private]
```

6.5.4.4 `_shape`

```
rcan.data_generator.SIM_Dataset._shape [private]
```

6.5.4.5 `_transform_function`

```
rcan.data_generator.SIM_Dataset._transform_function [private]
```

6.5.4.6 `_y`

`rca.data_generator.SIM_Dataset._y` [private]

6.5.4.7 `output_shape`

`rca.data_generator.SIM_Dataset.output_shape`

6.5.4.8 `output_signature`

`rca.data_generator.SIM_Dataset.output_signature`

6.5.4.9 `p_max`

`rca.data_generator.SIM_Dataset.p_max`

6.5.4.10 `p_min`

`rca.data_generator.SIM_Dataset.p_min`

6.5.4.11 `steps_per_epoch`

`rca.data_generator.SIM_Dataset.steps_per_epoch`

The documentation for this class was generated from the following file:

- `/home/jhughes2712/projects/sim_project/jh2284/src/rca/data_generator.py`

6.6 generate_sim.SimulationRunner Class Reference

Class which performs a batch of simulations, either sequentially or in parallel.

Public Member Functions

- `def __init__ (self, input_dir, output_dir, index_range, z_offset)`
- `def do_sim (self, i, sim, vol)`
Creates a new random virtual microscope simulator, takes a new sample from the VHP dataset, runs the simulation on the sample, and saves the results, along with the ground truth, in a single TIFF file.
- `def run (self)`
Runs a series of simulations sequentially.

Public Attributes

- `input_dir`
- `input_files`
- `output_dir`
- `range`
- `z_offset`

6.6.1 Detailed Description

Class which performs a batch of simulations, either sequentially or in parallel.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 __init__()

```
def generate_sim.SimulationRunner.__init__ (
    self,
    input_dir,
    output_dir,
    index_range,
    z_offset )
```

6.6.3 Member Function Documentation

6.6.3.1 do_sim()

```
def generate_sim.SimulationRunner.do_sim (
    self,
    i,
    sim,
    vol )
```

Creates a new random virtual microscope simulator, takes a new sample from the VHP dataset, runs the simulation on the sample, and saves the results, along with the ground truth, in a single TIFF file.

The parameters are saved in an accompanying JSON file.

6.6.3.2 run()

```
def generate_sim.SimulationRunner.run (
    self )
```

Runs a series of simulations sequentially.

6.6.4 Member Data Documentation

6.6.4.1 input_dir

generate_sim.SimulationRunner.input_dir

6.6.4.2 input_files

generate_sim.SimulationRunner.input_files

6.6.4.3 output_dir

generate_sim.SimulationRunner.output_dir

6.6.4.4 range

generate_sim.SimulationRunner.range

6.6.4.5 z_offset

generate_sim.SimulationRunner.z_offset

The documentation for this class was generated from the following file:

- /home/jhughes2712/projects/sim_project/jh2284/src/[generate_sim.py](#)

6.7 generate_sim.Simulator Class Reference

The [Simulator](#) class encapsulates the state of a 3D microscope simulation.

Public Member Functions

- `def __init__ (self, **kwargs)`
- `def randomise (self)`
- `def params_dict (self)`
- `def psf_params (self)`
- `def wavevectors (self)`
Calculates wavevectors inside the sample for the three beams, for a given number of rotations of those beams.
- `def illumination (self)`
Calculates the illumination intensity in the sample; returns ndarray of shape (n_rotations, n_shifts, n_x, n_x, n_z)
- `def in_focus_plane (self, sample)`
Returns the designated 'ground truth' plane.
- `def psf (self)`
Calculates a PSF if it has not been done already.
- `def simulate_sim (self, sample)`
Calculates the 15 simulated SIM images for a given sample.
- `def simulate_ideal_superres (self, sample)`
Simulates the best-case scenario for a 3D SIM reconstruction, by convolving the in-focus plane with a small PSF.
- `def add_noise (self, image)`
Adds a combination of Gaussian and Poissonian noise to the image.

Public Attributes

- `n_shifts`
- `n_angles`
- `n_x`
- `n_z`
- `n_rotations`
- `res_axial`
- `res_lateral`
- `delta_z_p`
- `n_sample`
- `n_i`
- `n_g`
- `z`
- `z_p`
- `angle_error`
- `poisson_photons`
- `signal_to_noise`
- `lambda0`
- `k0`
- `lambda_exc`
- `k_exc`
- `beam_position`

Private Attributes

- [_psf](#)
- [_superres_psf](#)
- [_illumination](#)

6.7.1 Detailed Description

The [Simulator](#) class encapsulates the state of a 3D microscope simulation.

A single instance of this class corresponds to a specific set of microscope parameters. These parameters are randomly chosen upon object creation.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 `__init__()`

```
def generate_sim.Simulator.__init__ (
    self,
    ** kwargs )
```

6.7.3 Member Function Documentation

6.7.3.1 `add_noise()`

```
def generate_sim.Simulator.add_noise (
    self,
    image )
```

Adds a combination of Gaussian and Poissonian noise to the image.

6.7.3.2 `illumination()`

```
def generate_sim.Simulator.illumination (
    self )
```

Calculates the illumination intensity in the sample; returns ndarray of shape (n_rotations, n_shifts, n_x, n_x, n_z)

6.7.3.3 in_focus_plane()

```
def generate_sim.Simulator.in_focus_plane (
    self,
    sample )
```

Returns the designated 'ground truth' plane.

6.7.3.4 params_dict()

```
def generate_sim.Simulator.params_dict (
    self )
```

6.7.3.5 psf()

```
def generate_sim.Simulator.psf (
    self )
```

Calculates a PSF if it has not been done already.

6.7.3.6 psf_params()

```
def generate_sim.Simulator.psf_params (
    self )
```

6.7.3.7 randomise()

```
def generate_sim.Simulator.randomise (
    self )
```

6.7.3.8 simulate_ideal_superres()

```
def generate_sim.Simulator.simulate_ideal_superres (
    self,
    sample )
```

Simulates the best-case scenario for a 3D SIM reconstruction, by convolving the in-focus plane with a small PSF.

6.7.3.9 simulate_sim()

```
def generate_sim.Simulator.simulate_sim (
    self,
    sample )
```

Calculates the 15 simulated SIM images for a given sample.

6.7.3.10 wavevectors()

```
def generate_sim.Simulator.wavevectors (
    self )
```

Calculates wavevectors inside the sample for the three beams, for a given number of rotations of those beams.

Returns ndarray of shape (n_rotations, n_beams, 3), where n_beams = 3

6.7.4 Member Data Documentation

6.7.4.1 _illumination

```
generate_sim.Simulator._illumination [private]
```

6.7.4.2 _psf

```
generate_sim.Simulator._psf [private]
```

6.7.4.3 _superres_psf

```
generate_sim.Simulator._superres_psf [private]
```

6.7.4.4 angle_error

```
generate_sim.Simulator.angle_error
```

6.7.4.5 beam_position

`generate_sim.Simulator.beam_position`

6.7.4.6 delta_z_p

`generate_sim.Simulator.delta_z_p`

6.7.4.7 k0

`generate_sim.Simulator.k0`

6.7.4.8 k_exc

`generate_sim.Simulator.k_exc`

6.7.4.9 lambda0

`generate_sim.Simulator.lambda0`

6.7.4.10 lambda_exc

`generate_sim.Simulator.lambda_exc`

6.7.4.11 n_angles

`generate_sim.Simulator.n_angles`

6.7.4.12 n_g

`generate_sim.Simulator.n_g`

6.7.4.13 n_i

`generate_sim.Simulator.n_i`

6.7.4.14 n_rotations

`generate_sim.Simulator.n_rotations`

6.7.4.15 n_sample

`generate_sim.Simulator.n_sample`

6.7.4.16 n_shifts

`generate_sim.Simulator.n_shifts`

6.7.4.17 n_x

`generate_sim.Simulator.n_x`

6.7.4.18 n_z

`generate_sim.Simulator.n_z`

6.7.4.19 poisson_photons

`generate_sim.Simulator.poisson_photons`

6.7.4.20 res_axial

`generate_sim.Simulator.res_axial`

6.7.4.21 res_lateral

`generate_sim.Simulator.res_lateral`

6.7.4.22 signal_to_noise

`generate_sim.Simulator.signal_to_noise`

6.7.4.23 z

`generate_sim.Simulator.z`

6.7.4.24 z_p

`generate_sim.Simulator.z_p`

The documentation for this class was generated from the following file:

- [/home/jhughes2712/projects/sim_project/jh2284/src/generate_sim.py](#)

Chapter 7

File Documentation

7.1 /home/jhughes2712/projects/sim_project/jh2284/src/analyse.py File Reference

Script producing plots and small datasets that summarise the performance of models.

Namespaces

- [analyse](#)

Functions

- def [analyse.reshape_to_bcwh](#) (data)

Variables

- [analyse.parser](#) = argparse.ArgumentParser()
- [analyse.type](#)
- [analyse.str](#)
- [analyse.required](#)
- [analyse.default](#)
- [analyse.int](#)
- [analyse.args](#) = parser.parse_args()
- [analyse.output_dir](#) = pathlib.Path(args.output_dir)
- [analyse.parents](#)
- [analyse.True](#)
- [analyse.exist_ok](#)
- tuple [analyse.device](#)
- [analyse.ckpt](#)
- [analyse.model](#)
- [analyse.RCAN_hyperparameters](#) = ckpt["hyperparameters"]
- [analyse.gt_dir](#) = pathlib.Path(args.gt_dir)
- [analyse.raw_dir](#) = pathlib.Path(args.raw_dir)
- [analyse.model_1_dir](#) = pathlib.Path(args.model_1_dir)
- [analyse.gt_files](#) = sorted(list(gt_dir.glob(args.glob_str)))

- `analyse.raw_files` = sorted(list(raw_dir.glob(args.glob_str)))
- `analyse.model_1_files` = sorted(list(model_1_dir.glob(args.glob_str)))
- `analyse.model_2_dir` = pathlib.Path(args.model_2_dir)
- `analyse.model_2_files` = sorted(list(model_2_dir.glob(args.glob_str)))
- `analyse.psnr` = PSNR(data_range=65536, device=device)
- `analyse.ssim`
- `analyse.df`
- `analyse.N` = len(gt_files)
- `def analyse.gt` = reshape_to_bcwh(tifffile.imread(gt_files[i]))
- `def analyse.raw` = reshape_to_bcwh(tifffile.imread(raw_files[i]))
- `def analyse.model_1` = reshape_to_bcwh(tifffile.imread(model_1_files[i]))
- `def analyse.model_2` = reshape_to_bcwh(tifffile.imread(model_2_files[i]))
- `analyse.rng` = np.random.default_rng(seed=31052024)
- `analyse.img_idx` = list(range(N))
- `list analyse.gt_samples` = [np.squeeze(tifffile.imread(gt_files[i])) for i in img_idx]
- `list analyse.raw_samples` = [np.squeeze(tifffile.imread(raw_files[i])) for i in img_idx]
- `list analyse.model_1_samples`
- `list analyse.model_2_samples`
- `analyse.cmap`

7.1.1 Detailed Description

Script producing plots and small datasets that summarise the performance of models.

This script reads directories of reconstructed images, and compares raw versus model reconstructions versus ground truth. The script then produces summary statistics, saves relevant metrics to a .csv file, and produces samples of cropped image regions for comparison.

Arguments:

- g: directory path for ground-truth images
- r: directory path for raw images
- a: directory path for model-1-restored images
- b: directory path for model-2-restored images
- o: output directory for analysis plots, default "figures/"
- x: filepath for model 1 checkpoint (plots learning curve)
- y: filepath for model 2 checkpoint (plots learning curve)
- s: globbing string, to analyse a subset of images
- n: number of sample crops to display, default 0.

7.2 /home/jhughes2712/projects/sim_project/jh2284/src/apply.py File Reference

Script producing restored images resulting from an RCAN denoiser being applied to low SNR images.

Namespaces

- `apply`

Functions

- `def apply.normalize_between_zero_and_one (m)`

Variables

- `apply.parser = argparse.ArgumentParser()`
- `apply.type`
- `apply.str`
- `apply.required`
- `apply.int`
- `apply.choices`
- `apply.default`
- `apply.percentile`
- `apply.action`
- `apply.args = parser.parse_args()`
- `apply.input_path = pathlib.Path(args.input)`
- `apply.output_path = pathlib.Path(args.output)`
- `apply.parents`
- `apply.raw_files = sorted(input_path.glob("*.tif"))`
- `apply.data = itertools.zip_longest(raw_files, [])`
- tuple `apply.device`
- `apply.ckpt`
- `apply.model`
- `apply.RCAN_hyperparameters = ckpt["hyperparameters"]`
- list `apply.overlap_shape`
- `apply.raw = normalize(tifffile.imread(raw_file), args.p_min, args.p_max)`
- `apply.restored`
- `apply.output_file = output_path / ("pred_" + raw_file.name)`
- `apply.imagej`

7.2.1 Detailed Description

Script producing restored images resulting from an RCAN denoiser being applied to low SNR images.

This script takes directories of raw images, and a model checkpoint file, and applies the model to the image in a patched fashion. The details of this patching, and the output datatype, can be configured.

Arguments:

- m: model checkpoint filepath
- i: low SNR image directory path
- o: output directory path
- b: specifies pixel bit depth to save for output (8 or 16)
- O: block overlap shape (by default input_shape / 8)

- p_min: input normalization parameter, percentile maps to zero
- p_max: input normalization parameter, percentile maps to one
- normalize_output_range_between_zero_and_one: scaling for output

Adapted from <https://github.com/AiviaCommunity/3D-RCAN/blob/TF2/apply.py>

Copyright 2021 SVision Technologies LLC. Copyright 2021-2022 Leica Microsystems, Inc. Creative Commons Attribution-NonCommercial 4.0 International Public License (CC BY-NC 4.0) <https://creativecommons.org/licenses/by-nc/4.0/>

7.3 /home/jhughes2712/projects/sim_project/jh2284/src/convert_omx_↔ to_czxy.py File Reference

Script enabling .tif file conversion between OMX and CZXY.

Namespaces

- [convert_omx_to_czxy](#)

Variables

- [convert_omx_to_czxy.parser](#) = argparse.ArgumentParser()
- [convert_omx_to_czxy.type](#)
- [convert_omx_to_czxy.str](#)
- [convert_omx_to_czxy.required](#)
- [convert_omx_to_czxy.int](#)
- [convert_omx_to_czxy.action](#)
- [convert_omx_to_czxy.args](#) = parser.parse_args()
- [convert_omx_to_czxy.input_dir](#) = pathlib.Path(args.input)
- [convert_omx_to_czxy.input_files](#) = sorted(input_dir.rglob("*.tif"))
- [convert_omx_to_czxy.original](#) = tifffile.imread(input_file)
- [convert_omx_to_czxy.n_phases](#) = args.num_phases
- [convert_omx_to_czxy.n_angles](#) = args.num_angles
- [convert_omx_to_czxy.converted](#)
- [convert_omx_to_czxy.imagej](#)

7.3.1 Detailed Description

Script enabling .tif file conversion between OMX and CZXY.

This script takes directories of image volumes as input, and converts, in place, between the OMX and CZXY formats (in either direction). In the OMX format, the first dimension is of size n_phases x n_z x n_angles; moving along this dimension, the phase changes first, then the z-value, then the angle. The CZXY format is the same, but the z-dimension of the image is separated into the 2nd dimension, so that the first dimension is just n_phases x n_angles.

Arguments:

- i: image directory
- p: number of phases
- a: number of angles
- b: specifies conversion - if not used it will be OMX to CZXY, the b flag reverses this direction.

7.4 /home/jhughes2712/projects/sim_project/jh2284/src/convert_omx_to_paz.py File Reference

Script enabling .tif file conversion between OMX and PAZ.

Namespaces

- [convert_omx_to_paz](#)

Variables

- [convert_omx_to_paz.parser](#) = argparse.ArgumentParser()
- [convert_omx_to_paz.type](#)
- [convert_omx_to_paz.str](#)
- [convert_omx_to_paz.required](#)
- [convert_omx_to_paz.int](#)
- [convert_omx_to_paz.action](#)
- [convert_omx_to_paz.args](#) = parser.parse_args()
- [convert_omx_to_paz.input_dir](#) = pathlib.Path(args.input)
- [convert_omx_to_paz.input_files](#) = sorted(input_dir.rglob("*.tif"))
- [convert_omx_to_paz.original](#) = tifffile.imread(input_file)
- [convert_omx_to_paz.n_phases](#) = args.num_phases
- [convert_omx_to_paz.n_angles](#) = args.num_angles
- [convert_omx_to_paz.converted](#) = np.zeros_like(original)
- [convert_omx_to_paz.imagej](#)

7.4.1 Detailed Description

Script enabling .tif file conversion between OMX and PAZ.

This script takes directories of image volumes as input, and converts, in place, between the OMX and PAZ formats (in either direction). In the OMX format, the first dimension is of size `n_phases` x `n_z` x `n_angles`; moving along this dimension, the phase changes first, then the z-value, then the angle. The PAZ format is the same except the order is changed so that z-values and angels are swapped.

Arguments:

- `i`: image directory
- `p`: number of phases
- `a`: number of angles
- `b`: specifies conversion - if not used it will be OMX to PAZ, the `b` flag reverses this direction.

7.5 /home/jhughes2712/projects/sim_project/jh2284/src/convert_slices_to_volumes.py File Reference

Script enabling construction of 3D image volumes from large RGB 2D image slices.

Namespaces

- [convert_slices_to_volumes](#)

Variables

- [convert_slices_to_volumes.parser](#) = argparse.ArgumentParser()
- [convert_slices_to_volumes.type](#)
- [convert_slices_to_volumes.str](#)
- [convert_slices_to_volumes.required](#)
- [convert_slices_to_volumes.tuple_of_ints](#)
- [convert_slices_to_volumes.default](#)
- [convert_slices_to_volumes.args](#) = parser.parse_args()
- [convert_slices_to_volumes.input_dir](#) = pathlib.Path(args.input)
- [convert_slices_to_volumes.output_dir](#) = pathlib.Path(args.output)
- [convert_slices_to_volumes.input_files](#) = sorted(input_dir.glob("*.tif"))
- [convert_slices_to_volumes.parents](#)
- [convert_slices_to_volumes.True](#)
- [convert_slices_to_volumes.exist_ok](#)
- [convert_slices_to_volumes.volume](#) = np.zeros((len(input_files), 3061, 4096), dtype=np.uint8)
- [convert_slices_to_volumes.input_slice](#) = tiffimage.imread(file)
- [convert_slices_to_volumes.subvolume](#)
- tuple [convert_slices_to_volumes.output_file](#)
- [convert_slices_to_volumes.imagej](#)

7.5.1 Detailed Description

Script enabling construction of 3D image volumes from large RGB 2D image slices.

Takes a directory of 2D image slices as input, and converts to 3D volumes. The 2D images are assumed to be ordered z-axially; the number of images is the number of voxels in the z-direction of the 3D volumes. The lateral cross-sections of the 3D images are determined by script arguments. Saves in uint16 depth.

Arguments:

- i: directory path for 2D images
- o: directory path for 3D image volumes
- s: start pixel coordinates (x, y)
- j: crop size for image volume (crop_x, crop_y)
- n: number of crops to take in each direction (steps_x, steps_y)
- l: filename prefix, default "volume"

7.6 /home/jhughes2712/projects/sim_project/jh2284/src/generate_sim.py File Reference

Script simulating the acquisition of 3D SIM image volumes.

Classes

- class `generate_sim.Simulator`
The `Simulator` class encapsulates the state of a 3D microscope simulation.
- class `generate_sim.SimulationRunner`
Class which performs a batch of simulations, either sequentially or in parallel.

Namespaces

- `generate_sim`

Functions

- def `generate_sim.arange_zero` (n, spacing=1)
- def `generate_sim.threshold_norm` (sample)
Applies a threshold and normalises the sample to improve contrast.

Variables

- `generate_sim.parser` = `argparse.ArgumentParser()`
- `generate_sim.type`
- `generate_sim.str`
- `generate_sim.required`
- `generate_sim.int`
- `generate_sim.default`
- `generate_sim.args` = `parser.parse_args()`
- `generate_sim.runner`

7.6.1 Detailed Description

Script simulating the acquisition of 3D SIM image volumes.

Takes a directory of 3D image volumes as input, and produces synthetic 3-beam SIM volumes of size (15, 32, 256, 256).

Arguments:

- i: directory path of input volumes
- o: directory path of output volumes
- s: start index of sorted input files to process
- e: end index of sorted input files to process
- z: z_offset, used to specify the region of the input volume to use.

7.7 /home/jhughes2712/projects/sim_project/jh2284/src/image_noising.py File Reference ↩

Script which converts a directory of high-SNR SIM images into a training dataset.

Namespaces

- [image_noising](#)

Functions

- [def image_noising.save_image_pair](#) (gt_img, split, name, channel_idx)

Variables

- [image_noising.parser](#) = argparse.ArgumentParser()
- [image_noising.type](#)
- [image_noising.str](#)
- [image_noising.required](#)
- [image_noising.int](#)
- [image_noising.choices](#)
- [image_noising.float](#)
- [image_noising.default](#)
- [image_noising.args](#) = parser.parse_args()
- [image_noising.input_path](#) = pathlib.Path(args.input)
- [image_noising.output_path](#) = pathlib.Path(args.output)
- [image_noising.parents](#)
- [image_noising.output_train_gt_path](#) = output_path.joinpath("Training", "GT")
- [image_noising.output_train_raw_path](#) = output_path.joinpath("Training", "Raw")
- [image_noising.output_val_gt_path](#) = output_path.joinpath("Validation", "GT")
- [image_noising.output_val_raw_path](#) = output_path.joinpath("Validation", "Raw")
- [image_noising.output_test_gt_path](#) = output_path.joinpath("Testing", "GT")
- [image_noising.output_test_raw_path](#) = output_path.joinpath("Testing", "Raw")
- [image_noising.data](#) = sorted(input_path.glob("*.tif"))
- [image_noising.n_acquisitions](#) = tiffimage.imread(data[0]).shape[0] // args.channels
- [image_noising.n_img](#) = len(data)
- [image_noising.train_size](#) = int((1 - args.test_fraction) * n_img)
- [image_noising.val_size](#) = int(args.val_fraction * train_size)
- [image_noising.rng](#) = np.random.default_rng(seed=25042024)
- [image_noising.img_idx_all](#) = list(range(n_img))
- [image_noising.img_idx_test](#) = img_idx_all[train_size:]
- [image_noising.img_idx_train](#) = img_idx_all[: train_size - val_size]
- [image_noising.img_idx_val](#) = img_idx_all[train_size - val_size : train_size]
- [image_noising.gt](#) = tiffimage.imread(img_file)
- string [image_noising.split](#) = "train"

7.7.1 Detailed Description

Script which converts a directory of high-SNR SIM images into a training dataset.

Each image is duplicated so that a low SNR counterpart is produced, simulating the same sample imaged with a lower illumination intensity. The data is then randomly split into train, validation, and testing subsets.

Arguments:

- i: directory path of input image
- o: directory path of output
- d: dimension
- s: scale factor used to simulate the low SNR images.
- tf: the fraction of the full dataset used for the hold-out test set.
- vf: the fraction of the *training* dataset that is reserved for validation during training.

7.8 /home/jhughes2712/projects/sim_project/jh2284/src/manage_stack.py File Reference

Script handling the stacking and unstacking of groups of images, for the purpose of batch reconstructions.

Namespaces

- [manage_stack](#)

Variables

- [manage_stack.parser](#) = argparse.ArgumentParser()
- [manage_stack.type](#)
- [manage_stack.str](#)
- [manage_stack.required](#)
- [manage_stack.int](#)
- [manage_stack.choices](#)
- [manage_stack.default](#)
- [manage_stack.action](#)
- [manage_stack.args](#) = parser.parse_args()
- [manage_stack.output_dir](#) = pathlib.Path(args.output_dir)
- [manage_stack.parents](#)
- [manage_stack.True](#)
- [manage_stack.exist_ok](#)
- [manage_stack.files](#) = sorted(list(pathlib.Path(args.input_dir).glob(args.glob_str)))
- [int manage_stack.stack_number](#) = -1 else args.stack_number
- [int manage_stack.number_of_stacks](#) = len(files) // stack_number
- [manage_stack.sample](#) = tifffile.imread(files[0])
- [manage_stack.stack](#)
- [manage_stack.img_data](#) = tifffile.imread(input_file)
- [tuple manage_stack.filename](#)
- [tuple manage_stack.output_file](#) = output_dir / filename
- [manage_stack.n_acq](#) = args.num_acquisitions
- [manage_stack.n_z](#) = sample.shape[0] // n_acq
- [manage_stack.output_data](#)

7.8.1 Detailed Description

Script handling the stacking and unstacking of groups of images, for the purpose of batch reconstructions.

Takes a directory of images as input, and either stacks or unstacks the images there according to the configuration. 3D Image Volumes are expected to be in PAZ format.

Arguments:

- i: directory path of input images
- o: directory path of output images
- n: output image name prefix - only applies in 'stack' mode

- d: dimension
- q: number of SIM acquisitions per image - currently also used to set the number of z-planes per image when unstacking reconstructions
- g: glob string used to choose images from input directory
- u: if used, sets mode to 'unstack'
- s: start index of sorted input files to process
- e: end index of sorted input files to process
- t: number of images to stack together - only applies in 'stack' mode

7.9 /home/jhughes2712/projects/sim_project/jh2284/src/rcan/__init__.py File Reference

Namespaces

- [rcan](#)

7.10 /home/jhughes2712/projects/sim_project/jh2284/src/synthetic_sim/__init__.py File Reference

Namespaces

- [synthetic_sim](#)

7.11 /home/jhughes2712/projects/sim_project/jh2284/src/rcan/data_generator.py File Reference

Module that handles processing and batching of data during training loop.

Classes

- class [rcan.data_generator.SIM_Dataset](#)
Generates batches of images with real-time data augmentation.

Namespaces

- [rcan.data_generator](#)

Functions

- def [rcan.data_generator.load_SIM_dataset](#) (images, shape, batch_size, transform_function, intensity_threshold, area_threshold, scale_factor, steps_per_epoch, p_min, p_max)
Wraps [SIM_Dataset](#) object in a PyTorch Dataloader object to enable batch loading.

7.11.1 Detailed Description

Module that handles processing and batching of data during training loop.

This module primarily defines the SIM_Dataset class which handles image cropping, normalization, augmentation, and intensity-threshold-area based rejection.

Migrated from https://github.com/AiviaCommunity/3D-RCAN/blob/TF2/rcan/data_generator.py

Copyright 2021 SVision Technologies LLC. Copyright 2021-2022 Leica Microsystems, Inc. Creative Commons Attribution-NonCommercial 4.0 International Public License (CC BY-NC 4.0) <https://creativecommons.org/licenses/by-nc/4.0/>

7.12 /home/jhughes2712/projects/sim_project/jh2284/src/rcan/model.py File Reference

Module defining the RCAN model architecture.

Classes

- class [rcan.model._channel_attention_block](#)
Implements channel attention block/layer.
- class [rcan.model._residual_channel_attention_blocks](#)
Implements residual group based on [1].
- class [rcan.model.RCAN](#)
Builds a residual channel attention network.

Namespaces

- [rcan.model](#)

Functions

- def [rcan.model._conv](#) (ndim, in_filters, out_filters, kernel_size, padding="same", **kwargs)
Returns the appropriate torch.nn convolution layer based on parameters.
- def [rcan.model._global_average_pooling](#) (ndim)
Returns the appropriate torch.nn pooling layer based on parameters.
- def [rcan.model._standardize](#) (x)
Standardises input data.
- def [rcan.model._destandardize](#) (x)
Inverse of _standardize.

7.12.1 Detailed Description

Module defining the RCAN model architecture.

Module that defines a number of classes inheriting from `nn.Module`, implementing different levels of the RCAN architecture. This includes the channel attention layer, residual channel attention block, and RCAN itself.

Migrated from <https://github.com/AiviaCommunity/3D-RCAN/blob/TF2/rcan/model.py>

Copyright 2021 SVision Technologies LLC. Copyright 2021-2022 Leica Microsystems, Inc. Creative Commons Attribution-NonCommercial 4.0 International Public License (CC BY-NC 4.0) <https://creativecommons.org/licenses/by-nc/4.0/>

7.13 /home/jhughes2712/projects/sim_↵ project/jh2284/src/rcan/plotting.py File Reference

Module providing helper functions for matplotlib plots.

Namespaces

- [rcan.plotting](#)

Functions

- def [rcan.plotting.plot_learning_curve](#) (losses_train, losses_val, psnr_train, psnr_val, ssim_train, ssim_val, fig-size, output_path)
Plots the learning curve metrics from a model checkpoint according to loss, PSNR, and SSIM.
- def [rcan.plotting.plot_reconstructions](#) (device, output_path, dim, gt_imgs, raw_imgs, model_1_imgs, model_↵_2_imgs=None, cmap="inferno")
Plots a sample of reconstructions comparing GT vs Raw vs Restored.

7.13.1 Detailed Description

Module providing helper functions for matplotlib plots.

Provides tools to assist with analysis of trained networks, including samples of restored reconstructions, metrics, and model progress during training.

7.14 /home/jhughes2712/projects/sim_project/jh2284/src/rcan/utils.py File Reference

Contains utility functions for the training loop and inference.

Namespaces

- [rcan.utils](#)

Functions

- def [rcan.utils.normalize](#) (image, p_min=2, p_max=99.9, dtype="float32")
Normalizes the image intensity so that the p_{min} -th and the p_{max} -th percentiles are converted to 0 and 1 respectively.
- def [rcan.utils.apply](#) (model, data, model_input_image_shape, model_output_image_shape, num_input_channels, num_output_channels, batch_size, device, overlap_shape=None, verbose=False)
Applies a model to an input image.
- def [rcan.utils.load_rcan_checkpoint](#) (ckpt_path, device)
Enables loading of RCAN checkpointed model.
- def [rcan.utils.tuple_of_ints](#) (string)
Defines behaviour of parsing tuples of ints (argparse).
- def [rcan.utils.percentile](#) (x)
Defines behaviour of parsing percentiles (argparse).

7.14.1 Detailed Description

Contains utility functions for the training loop and inference.

Migrated from <https://github.com/AiviaCommunity/3D-RCAN/blob/TF2/rcan/utils.py>

Copyright 2021 SVision Technologies LLC. Creative Commons Attribution-NonCommercial 4.0 International Public License (CC BY-NC 4.0) <https://creativecommons.org/licenses/by-nc/4.0/>

7.15 /home/jhughes2712/projects/sim_project/jh2284/src/recon_postprocess.py File Reference

Script handling the postprocessing of SIM reconstructions.

Namespaces

- [recon_postprocess](#)

Variables

- [recon_postprocess.parser](#) = argparse.ArgumentParser()
- [recon_postprocess.type](#)
- [recon_postprocess.str](#)
- [recon_postprocess.required](#)
- [recon_postprocess.args](#) = parser.parse_args()
- [recon_postprocess.files](#) = sorted(list(pathlib.Path(args.input_dir).rglob("*.tif")))
- [recon_postprocess.img_data](#) = tiffimage.imread(input_file)

7.15.1 Detailed Description

Script handling the postprocessing of SIM reconstructions.

Takes a directory of images as input, clips zero values, and scales to the full 16-bit depth range. Operates in-place.

Arguments:

- `i`: directory path of input images

7.16 `/home/jhughes2712/projects/sim_project/jh2284/src/recon_preprocess.py` File Reference

Script handling the preprocessing of images before SIM reconstruction.

Namespaces

- `recon_preprocess`

Functions

- `def recon_preprocess.normalize_acquisition_intensity (data, dim)`

Variables

- `recon_preprocess.parser` = `argparse.ArgumentParser()`
- `recon_preprocess.type`
- `recon_preprocess.str`
- `recon_preprocess.required`
- `recon_preprocess.int`
- `recon_preprocess.choices`
- `recon_preprocess.percentile`
- `recon_preprocess.default`
- `recon_preprocess.action`
- `recon_preprocess.args` = `parser.parse_args()`
- `recon_preprocess.output_dir` = `pathlib.Path(args.output_dir)`
- `recon_preprocess.parents`
- `recon_preprocess.True`
- `recon_preprocess.exist_ok`
- `recon_preprocess.files` = `sorted(list(pathlib.Path(args.input_dir).glob("*.tif")))`
- `recon_preprocess.img_data` = `tiffimage.imread(input_file).astype("float32")`
- `recon_preprocess.output_file` = `output_dir / input_file.name`

7.16.1 Detailed Description

Script handling the preprocessing of images before SIM reconstruction.

Takes a directory of images as input, equalizes the total acquisition, intensities within each image, subtracts background and extreme pixels on a percentile basis, then scales to the full 16-bit depth range.

Arguments:

- i: directory path of input images
- o: directory path of output images
- d: dimension
- l: lower percentile used for clipping (background)
- u: upper percentile used for clipping (bright values)
- n: turns on normalization of acquisition intensity

7.17 /home/jhughes2712/projects/sim_project/jh2284/src/stats.py File Reference

Namespaces

- [stats](#)

Functions

- def [stats.paired_t](#) (gt_data, data)

Variables

- [stats.parser](#) = argparse.ArgumentParser()
- [stats.type](#)
- [stats.str](#)
- [stats.required](#)
- [stats.int](#)
- [stats.choices](#)
- [stats.default](#)
- [stats.args](#) = parser.parse_args()
- [stats.output_dir](#) = pathlib.Path(args.output_dir)
- [stats.parents](#)
- [stats.True](#)
- [stats.exist_ok](#)
- [stats.df](#)
- [stats.fig](#)
- [stats.ax](#)
- [stats.figsize](#)
- [stats.psnr_diff_1_max](#)
- [stats.psnr_diff_2_max](#)

- `stats.psnr_diff_1_min`
- `stats.psnr_diff_2_min`
- `tuple stats.hist_range_psnr`
- `stats.ssim_diff_1_max`
- `stats.ssim_diff_2_max`
- `stats.ssim_diff_1_min`
- `stats.ssim_diff_2_min`
- `tuple stats.hist_range_ssim`
- `stats.xlabel`
- `stats.title`
- `stats.range`
- `stats.color`
- `stats.mean_psnr_1 = np.mean(np.array(df['psnr_model_1']) - np.array(df['psnr_raw']))`
- `stats.se_psnr_1`
- `stats.mean_ssim_1 = np.mean(np.array(df['ssim_model_1']) - np.array(df['ssim_raw']))`
- `stats.se_ssim_1`
- `stats.mean_psnr_2`
- `stats.se_psnr_2`
- `stats.mean_ssim_2`
- `stats.se_ssim_2`
- `int stats.psnr_cols = 2 else df.columns[1:3]`
- `int stats.ssim_cols = 2 else df.columns[3:5]`
- `stats.dflong`
- `stats.dflongssim`
- `stats.data`
- `stats.x`
- `stats.y`
- `stats.hue`
- `stats.dodge`
- `stats.legend`
- `stats.palette`
- `stats.alpha`
- `stats.lw`

7.18 `/home/jhughes2712/projects/sim_project/jh2284/src/synthetic_sim/otf.py` File Reference

Classes

- class `synthetic_sim.otf.PsfParameters`
Class to store PSF parameters.

Namespaces

- `synthetic_sim.otf`

Functions

- def `synthetic_sim.otf.calc_psf` (params)
Calculate an approximate Gibson-Lanni PSF based on the parameters provided.

7.19 /home/jhughes2712/projects/sim_project/jh2284/src/train.py File Reference

Script used to train RCAN.

Namespaces

- [train](#)

Functions

- def [train.load_data_paths](#) (config, data_type)
- def [train.train](#) (train_loader, val_loader, optimizer, scheduler, net, batchsize, n_accumulations, saveinterval, nepoch, start_epoch=0, losses_train_epoch=[], losses_val_epoch=[], psnr_train_epoch=[], psnr_val_epoch=[], ssim_train_epoch=[], ssim_val_epoch=[])

Variables

- [train.parser](#) = argparse.ArgumentParser()
- [train.type](#)
- [train.str](#)
- [train.required](#)
- [train.args](#) = parser.parse_args()
- dictionary [train.schema](#)
- [train.config](#) = json.load(f)
- int [train.ndim](#) = tiffimage.imread(training_data[0]["raw"]).ndim - 1
- [train.input_shape](#) = config["input_shape"]
- tuple [train.device](#)
- [train.ckpt_path](#) = None if args.model_ckpt is None else pathlib.Path(args.model_ckpt)
- [train.model](#)
- dictionary [train.RCAN_hyperparameters](#)
- [train.ckpt](#)
- [train.train_loader](#)
- [train.val_loader](#)
- [train.optimizer](#)
- [train.scheduler](#)
- [train.output_dir](#) = pathlib.Path(args.output_dir)
- [train.parents](#)
- [train.True](#)
- [train.exist_ok](#)
- [train.n_accumulations](#)
- [train.saveinterval](#)
- [train.nepoch](#)
- [train.start_epoch](#)
- [train.losses_train_epoch](#)
- [train.losses_val_epoch](#)
- [train.psnr_train_epoch](#)
- [train.psnr_val_epoch](#)
- [train.ssim_train_epoch](#)
- [train.ssim_val_epoch](#)

7.19.1 Detailed Description

Script used to train RCAN.

Reads the specified config.json file, and trains an RCAN model accordingly. Intermediate training progress is saved using model checkpoints. Can handle resumed model training if a previous checkpoint is provided.

Arguments:

- c: filepath for config JSON file
- o: path of model checkpoint directory
- m: filepath of intermediate model checkpoint (if given, training resumes from this checkpoint)

Adapted from <https://github.com/AiviaCommunity/3D-RCAN/blob/TF2/train.py>

Copyright 2021 SVision Technologies LLC. Copyright 2021-2022 Leica Microsystems, Inc. Creative Commons Attribution-NonCommercial 4.0 International Public License (CC BY-NC 4.0) <https://creativecommons.org/licenses/by-nc/4.0/>

Index

/home/jhughes2712/projects/sim_project/jh2284/src/analyse.py,rcan.data_generator.SIM_Dataset, 81
93
_area_threshold
/home/jhughes2712/projects/sim_project/jh2284/src/apply.py, rcan.data_generator.SIM_Dataset, 82
94
_conv
/home/jhughes2712/projects/sim_project/jh2284/src/convert_omx_to_model.py, 41
96
_destandardize
/home/jhughes2712/projects/sim_project/jh2284/src/convert_omx_to_model.py, 41
97
_global_average_pooling
/home/jhughes2712/projects/sim_project/jh2284/src/convert_slice_to_model.py, 45
97
_illumination
/home/jhughes2712/projects/sim_project/jh2284/src/generate_sim.py,generate_sim.Simulator, 89
98
_intensity_threshold
/home/jhughes2712/projects/sim_project/jh2284/src/image_noise.py,rcan.data_generator.SIM_Dataset, 82
99
_psf
/home/jhughes2712/projects/sim_project/jh2284/src/manage_stack.py,generate_sim.Simulator, 89
101
_scale
/home/jhughes2712/projects/sim_project/jh2284/src/rcan/_init_rcan.py,rcan.data_generator.SIM_Dataset, 82
102
_scale_factor
/home/jhughes2712/projects/sim_project/jh2284/src/rcan/data_generator.py,rcan.data_generator.SIM_Dataset, 82
102
_shape
/home/jhughes2712/projects/sim_project/jh2284/src/rcan/model.py,rcan.data_generator.SIM_Dataset, 82
103
_standardize
/home/jhughes2712/projects/sim_project/jh2284/src/rcan/plotting.py,rcan.model, 42
104
_superres_psf
/home/jhughes2712/projects/sim_project/jh2284/src/rcan/utils.py,generate_sim.Simulator, 89
104
_transform_function
/home/jhughes2712/projects/sim_project/jh2284/src/recon_postprocess.py,rcan.data_generator.SIM_Dataset, 82
105
_y
/home/jhughes2712/projects/sim_project/jh2284/src/recon_preprocess.py,rcan.data_generator.SIM_Dataset, 82
106
/home/jhughes2712/projects/sim_project/jh2284/src/stats.py,action
107
apply, 16
/home/jhughes2712/projects/sim_project/jh2284/src/synthetic_sim.py,convert_omx_to_czxy, 20
102
convert_omx_to_paz, 23
/home/jhughes2712/projects/sim_project/jh2284/src/synthetic_sim.py,manage_stack, 35
108
recon_preprocess, 48
/home/jhughes2712/projects/sim_project/jh2284/src/train.py,add_noise
109
generate_sim.Simulator, 87
__getitem__
alpha
rcan.data_generator.SIM_Dataset, 81
stats, 51
__init__
analyse, 9
generate_sim.SimulationRunner, 84
args, 10
generate_sim.Simulator, 87
ckpt, 10
rcan.data_generator.SIM_Dataset, 80
cmap, 10
rcan.model._channel_attention_block, 70
default, 10
rcan.model._residual_channel_attention_blocks,
device, 10
73
df, 11
rcan.model.RCAN, 77
exist_ok, 11
__len__
gt, 11
gt_dir, 11

- gt_files, 11
- gt_samples, 11
- img_idx, 12
- int, 12
- model, 12
- model_1, 12
- model_1_dir, 12
- model_1_files, 12
- model_1_samples, 12
- model_2, 13
- model_2_dir, 13
- model_2_files, 13
- model_2_samples, 13
- N, 13
- output_dir, 13
- parents, 13
- parser, 14
- psnr, 14
- raw, 14
- raw_dir, 14
- raw_files, 14
- raw_samples, 14
- RCAN_hyperparameters, 14
- required, 14
- reshape_to_bcwh, 10
- rng, 15
- ssim, 15
- str, 15
- True, 15
- type, 15
- angle_error
 - generate_sim.Simulator, 89
- apply, 15
 - action, 16
 - args, 16
 - choices, 17
 - ckpt, 17
 - data, 17
 - default, 17
 - device, 17
 - imagej, 17
 - input_path, 17
 - int, 18
 - model, 18
 - normalize_between_zero_and_one, 16
 - output_file, 18
 - output_path, 18
 - overlap_shape, 18
 - parents, 18
 - parser, 18
 - percentile, 19
 - raw, 19
 - raw_files, 19
 - rcan.utils, 44
 - RCAN_hyperparameters, 19
 - required, 19
 - restored, 19
 - str, 19
 - type, 20
- arange_zero
 - generate_sim, 28
- args
 - analyse, 10
 - apply, 16
 - convert_omx_to_czxy, 20
 - convert_omx_to_paz, 23
 - convert_slices_to_volumes, 25
 - generate_sim, 29
 - image_noising, 31
 - manage_stack, 35
 - recon_postprocess, 46
 - recon_preprocess, 48
 - stats, 52
 - train, 62
- ax
 - stats, 52
- beam_position
 - generate_sim.Simulator, 89
- calc_psf
 - synthetic_sim.otf, 60
- Callable
 - synthetic_sim.otf.PsfParameters, 75
- channel_attention_block_list
 - rcan.model._residual_channel_attention_blocks, 74
- choices
 - apply, 17
 - image_noising, 31
 - manage_stack, 35
 - recon_preprocess, 48
 - stats, 52
- ckpt
 - analyse, 10
 - apply, 17
 - train, 62
- ckpt_path
 - train, 62
- cmap
 - analyse, 10
- color
 - stats, 52
- config
 - train, 62
- conv_1
 - rcan.model._channel_attention_block, 71
- conv_2
 - rcan.model._channel_attention_block, 71
- conv_input
 - rcan.model.RCAN, 78
- conv_list
 - rcan.model._residual_channel_attention_blocks, 74
 - rcan.model.RCAN, 78
- conv_output
 - rcan.model.RCAN, 78

- convert_omx_to_czxy, 20
 - action, 20
 - args, 20
 - converted, 20
 - imagej, 21
 - input_dir, 21
 - input_files, 21
 - int, 21
 - n_angles, 21
 - n_phases, 21
 - original, 21
 - parser, 22
 - required, 22
 - str, 22
 - type, 22
- convert_omx_to_paz, 22
 - action, 23
 - args, 23
 - converted, 23
 - imagej, 23
 - input_dir, 23
 - input_files, 23
 - int, 23
 - n_angles, 23
 - n_phases, 24
 - original, 24
 - parser, 24
 - required, 24
 - str, 24
 - type, 24
- convert_slices_to_volumes, 25
 - args, 25
 - default, 25
 - exist_ok, 25
 - imagej, 25
 - input_dir, 26
 - input_files, 26
 - input_slice, 26
 - output_dir, 26
 - output_file, 26
 - parents, 26
 - parser, 26
 - required, 27
 - str, 27
 - subvolume, 27
 - True, 27
 - tuple_of_ints, 27
 - type, 27
 - volume, 27
- converted
 - convert_omx_to_czxy, 20
 - convert_omx_to_paz, 23
- data
 - apply, 17
 - image_noising, 31
 - stats, 52
- default
 - analyse, 10
- apply, 17
 - convert_slices_to_volumes, 25
 - generate_sim, 29
 - image_noising, 31
 - manage_stack, 36
 - recon_preprocess, 48
 - stats, 52
- delta_z_p
 - generate_sim.Simulator, 90
- device
 - analyse, 10
 - apply, 17
 - train, 63
- df
 - analyse, 11
 - stats, 52
- dflong
 - stats, 53
- dflongssim
 - stats, 53
- do_sim
 - generate_sim.SimulationRunner, 84
- dodge
 - stats, 53
- exist_ok
 - analyse, 11
 - convert_slices_to_volumes, 25
 - manage_stack, 36
 - recon_preprocess, 48
 - stats, 53
 - train, 63
- fig
 - stats, 53
- figsize
 - stats, 54
- filename
 - manage_stack, 36
- files
 - manage_stack, 36
 - recon_postprocess, 46
 - recon_preprocess, 48
- float
 - image_noising, 31
 - synthetic_sim.otf.PsfParameters, 75
- forward
 - rcan.model._channel_attention_block, 71
 - rcan.model._residual_channel_attention_blocks, 73
 - rcan.model.RCAN, 78
- generate_sim, 28
 - arange_zero, 28
 - args, 29
 - default, 29
 - int, 29
 - parser, 29
 - required, 29

- runner, 29
- str, 29
- threshold_norm, 28
- type, 30
- generate_sim.SimulationRunner, 83
 - __init__, 84
 - do_sim, 84
 - input_dir, 85
 - input_files, 85
 - output_dir, 85
 - range, 85
 - run, 84
 - z_offset, 85
- generate_sim.Simulator, 86
 - __init__, 87
 - _illumination, 89
 - _psf, 89
 - _superres_psf, 89
 - add_noise, 87
 - angle_error, 89
 - beam_position, 89
 - delta_z_p, 90
 - illumination, 87
 - in_focus_plane, 87
 - k0, 90
 - k_exc, 90
 - lambda0, 90
 - lambda_exc, 90
 - n_angles, 90
 - n_g, 90
 - n_i, 90
 - n_rotations, 91
 - n_sample, 91
 - n_shifts, 91
 - n_x, 91
 - n_z, 91
 - params_dict, 88
 - poisson_photons, 91
 - psf, 88
 - psf_params, 88
 - randomise, 88
 - res_axial, 91
 - res_lateral, 91
 - signal_to_noise, 92
 - simulate_ideal_superres, 88
 - simulate_sim, 88
 - wavevectors, 89
 - z, 92
 - z_p, 92
- global_average_pooling
 - rca.model.channel_attention_block, 71
- gt
 - analyse, 11
 - image_noising, 31
- gt_dir
 - analyse, 11
- gt_files
 - analyse, 11
- gt_samples
 - analyse, 11
- hist_range_psnr
 - stats, 54
- hist_range_ssim
 - stats, 54
- hue
 - stats, 54
- illumination
 - generate_sim.Simulator, 87
- image_noising, 30
 - args, 31
 - choices, 31
 - data, 31
 - default, 31
 - float, 31
 - gt, 31
 - img_idx_all, 32
 - img_idx_test, 32
 - img_idx_train, 32
 - img_idx_val, 32
 - input_path, 32
 - int, 32
 - n_acquisitions, 32
 - n_img, 32
 - output_path, 33
 - output_test_gt_path, 33
 - output_test_raw_path, 33
 - output_train_gt_path, 33
 - output_train_raw_path, 33
 - output_val_gt_path, 33
 - output_val_raw_path, 33
 - parents, 33
 - parser, 34
 - required, 34
 - rng, 34
 - save_image_pair, 31
 - split, 34
 - str, 34
 - train_size, 34
 - type, 34
 - val_size, 34
- imagej
 - apply, 17
 - convert_omx_to_czxy, 21
 - convert_omx_to_paz, 23
 - convert_slices_to_volumes, 25
- img_data
 - manage_stack, 36
 - recon_postprocess, 46
 - recon_preprocess, 49
- img_idx
 - analyse, 12
- img_idx_all
 - image_noising, 32
- img_idx_test
 - image_noising, 32

- img_idx_train
 - image_noising, [32](#)
- img_idx_val
 - image_noising, [32](#)
- in_focus_plane
 - generate_sim.Simulator, [87](#)
- input_dir
 - convert_omx_to_czxy, [21](#)
 - convert_omx_to_paz, [23](#)
 - convert_slices_to_volumes, [26](#)
 - generate_sim.SimulationRunner, [85](#)
- input_files
 - convert_omx_to_czxy, [21](#)
 - convert_omx_to_paz, [23](#)
 - convert_slices_to_volumes, [26](#)
 - generate_sim.SimulationRunner, [85](#)
- input_path
 - apply, [17](#)
 - image_noising, [32](#)
- input_shape
 - train, [63](#)
- input_slice
 - convert_slices_to_volumes, [26](#)
- int
 - analyse, [12](#)
 - apply, [18](#)
 - convert_omx_to_czxy, [21](#)
 - convert_omx_to_paz, [23](#)
 - generate_sim, [29](#)
 - image_noising, [32](#)
 - manage_stack, [36](#)
 - recon_preprocess, [49](#)
 - stats, [54](#)
 - synthetic_sim.otf.PsfParameters, [75](#)
- k0
 - generate_sim.Simulator, [90](#)
- k_exc
 - generate_sim.Simulator, [90](#)
- lambda0
 - generate_sim.Simulator, [90](#)
- lambda_exc
 - generate_sim.Simulator, [90](#)
- legend
 - stats, [54](#)
- load_data_paths
 - train, [61](#)
- load_rcan_checkpoint
 - rcan.utils, [45](#)
- load_SIM_dataset
 - rcan.data_generator, [39](#)
- losses_train_epoch
 - train, [63](#)
- losses_val_epoch
 - train, [63](#)
- lw
 - stats, [55](#)
- manage_stack, [35](#)
 - action, [35](#)
 - args, [35](#)
 - choices, [35](#)
 - default, [36](#)
 - exist_ok, [36](#)
 - filename, [36](#)
 - files, [36](#)
 - img_data, [36](#)
 - int, [36](#)
 - n_acq, [36](#)
 - n_z, [37](#)
 - number_of_stacks, [37](#)
 - output_data, [37](#)
 - output_dir, [37](#)
 - output_file, [37](#)
 - parents, [37](#)
 - parser, [37](#)
 - required, [38](#)
 - sample, [38](#)
 - stack, [38](#)
 - stack_number, [38](#)
 - str, [38](#)
 - True, [38](#)
 - type, [39](#)
- mean_psnr_1
 - stats, [55](#)
- mean_psnr_2
 - stats, [55](#)
- mean_ssim_1
 - stats, [55](#)
- mean_ssim_2
 - stats, [55](#)
- model
 - analyse, [12](#)
 - apply, [18](#)
 - train, [63](#)
- model_1
 - analyse, [12](#)
- model_1_dir
 - analyse, [12](#)
- model_1_files
 - analyse, [12](#)
- model_1_samples
 - analyse, [12](#)
- model_2
 - analyse, [13](#)
- model_2_dir
 - analyse, [13](#)
- model_2_files
 - analyse, [13](#)
- model_2_samples
 - analyse, [13](#)
- N
 - analyse, [13](#)
- n_accumulations
 - train, [63](#)
- n_acq

- manage_stack, 36
- n_acquisitions
 - image_noising, 32
- n_angles
 - convert_omx_to_czxy, 21
 - convert_omx_to_paz, 23
 - generate_sim.Simulator, 90
- n_g
 - generate_sim.Simulator, 90
- n_i
 - generate_sim.Simulator, 90
- n_img
 - image_noising, 32
- n_phases
 - convert_omx_to_czxy, 21
 - convert_omx_to_paz, 24
- n_rotations
 - generate_sim.Simulator, 91
- n_sample
 - generate_sim.Simulator, 91
- n_shifts
 - generate_sim.Simulator, 91
- n_x
 - generate_sim.Simulator, 91
- n_z
 - generate_sim.Simulator, 91
 - manage_stack, 37
- ndim
 - train, 64
- nepoch
 - train, 64
- normalize
 - rcan.utils, 45
- normalize_acquisition_intensity
 - recon_preprocess, 48
- normalize_between_zero_and_one
 - apply, 16
- num_residual_groups
 - rcan.model.RCAN, 78
- number_of_stacks
 - manage_stack, 37
- optimizer
 - train, 64
- original
 - convert_omx_to_czxy, 21
 - convert_omx_to_paz, 24
- output_data
 - manage_stack, 37
- output_dir
 - analyse, 13
 - convert_slices_to_volumes, 26
 - generate_sim.SimulationRunner, 85
 - manage_stack, 37
 - recon_preprocess, 49
 - stats, 55
 - train, 64
- output_file
 - apply, 18
- convert_slices_to_volumes, 26
- manage_stack, 37
- recon_preprocess, 49
- output_path
 - apply, 18
 - image_noising, 33
- output_shape
 - rcan.data_generator.SIM_Dataset, 83
- output_signature
 - rcan.data_generator.SIM_Dataset, 83
- output_test_gt_path
 - image_noising, 33
- output_test_raw_path
 - image_noising, 33
- output_train_gt_path
 - image_noising, 33
- output_train_raw_path
 - image_noising, 33
- output_val_gt_path
 - image_noising, 33
- output_val_raw_path
 - image_noising, 33
- overlap_shape
 - apply, 18
- p_max
 - rcan.data_generator.SIM_Dataset, 83
- p_min
 - rcan.data_generator.SIM_Dataset, 83
- paired_t
 - stats, 51
- palette
 - stats, 55
- params_dict
 - generate_sim.Simulator, 88
- parents
 - analyse, 13
 - apply, 18
 - convert_slices_to_volumes, 26
 - image_noising, 33
 - manage_stack, 37
 - recon_preprocess, 49
 - stats, 56
 - train, 64
- parser
 - analyse, 14
 - apply, 18
 - convert_omx_to_czxy, 22
 - convert_omx_to_paz, 24
 - convert_slices_to_volumes, 26
 - generate_sim, 29
 - image_noising, 34
 - manage_stack, 37
 - recon_postprocess, 47
 - recon_preprocess, 49
 - stats, 56
 - train, 64
- percentile
 - apply, 19

- rcan.utils, 46
- recon_preprocess, 49
- plot_learning_curve
 - rcan.plotting, 42
- plot_reconstructions
 - rcan.plotting, 43
- poisson_photons
 - generate_sim.Simulator, 91
- psf
 - generate_sim.Simulator, 88
- psf_params
 - generate_sim.Simulator, 88
- psnr
 - analyse, 14
- psnr_cols
 - stats, 56
- psnr_diff_1_max
 - stats, 56
- psnr_diff_1_min
 - stats, 56
- psnr_diff_2_max
 - stats, 56
- psnr_diff_2_min
 - stats, 57
- psnr_train_epoch
 - train, 64
- psnr_val_epoch
 - train, 65
- randomise
 - generate_sim.Simulator, 88
- range
 - generate_sim.SimulationRunner, 85
 - stats, 57
- raw
 - analyse, 14
 - apply, 19
- raw_dir
 - analyse, 14
- raw_files
 - analyse, 14
 - apply, 19
- raw_samples
 - analyse, 14
- rcab_list
 - rcan.model.RCAN, 79
- rcan, 39
- rcan.data_generator, 39
 - load_SIM_dataset, 39
- rcan.data_generator.SIM_Dataset, 79
 - __getitem__, 81
 - __init__, 80
 - __len__, 81
 - _area_threshold, 82
 - _intensity_threshold, 82
 - _scale, 82
 - _scale_factor, 82
 - _shape, 82
 - _transform_function, 82
 - _y, 82
 - output_shape, 83
 - output_signature, 83
 - p_max, 83
 - p_min, 83
 - steps_per_epoch, 83
- rcan.model, 40
 - _conv, 41
 - _destandardize, 41
 - _global_average_pooling, 41
 - _standardize, 42
- rcan.model_channel_attention_block, 69
 - __init__, 70
 - conv_1, 71
 - conv_2, 71
 - forward, 71
 - global_average_pooling, 71
- rcan.model_residual_channel_attention_blocks, 72
 - __init__, 73
 - channel_attention_block_list, 74
 - conv_list, 74
 - forward, 73
 - repeat, 74
 - residual_scaling, 74
- rcan.model.RCAN, 76
 - __init__, 77
 - conv_input, 78
 - conv_list, 78
 - conv_output, 78
 - forward, 78
 - num_residual_groups, 78
 - rcab_list, 79
- rcan.plotting, 42
 - plot_learning_curve, 42
 - plot_reconstructions, 43
- rcan.utils, 44
 - apply, 44
 - load_rcan_checkpoint, 45
 - normalize, 45
 - percentile, 46
 - tuple_of_ints, 46
- RCAN_hyperparameters
 - analyse, 14
 - apply, 19
 - train, 65
- recon_postprocess, 46
 - args, 46
 - files, 46
 - img_data, 46
 - parser, 47
 - required, 47
 - str, 47
 - type, 47
- recon_preprocess, 47
 - action, 48
 - args, 48
 - choices, 48
 - default, 48

- exist_ok, 48
- files, 48
- img_data, 49
- int, 49
- normalize_acquisition_intensity, 48
- output_dir, 49
- output_file, 49
- parents, 49
- parser, 49
- percentile, 49
- required, 49
- str, 50
- True, 50
- type, 50
- repeat
 - rca.model._residual_channel_attention_blocks, 74
- required
 - analyse, 14
 - apply, 19
 - convert_omx_to_czxy, 22
 - convert_omx_to_paz, 24
 - convert_slices_to_volumes, 27
 - generate_sim, 29
 - image_noising, 34
 - manage_stack, 38
 - recon_postprocess, 47
 - recon_preprocess, 49
 - stats, 57
 - train, 65
- res_axial
 - generate_sim.Simulator, 91
- res_lateral
 - generate_sim.Simulator, 91
- reshape_to_bcwh
 - analyse, 10
- residual_scaling
 - rca.model._residual_channel_attention_blocks, 74
- restored
 - apply, 19
- rng
 - analyse, 15
 - image_noising, 34
- run
 - generate_sim.SimulationRunner, 84
- runner
 - generate_sim, 29
- sample
 - manage_stack, 38
- save_image_pair
 - image_noising, 31
- saveinterval
 - train, 65
- scheduler
 - train, 65
- schema
 - train, 65
- se_psnr_1
 - stats, 57
- se_psnr_2
 - stats, 57
- se_ssim_1
 - stats, 57
- se_ssim_2
 - stats, 58
- signal_to_noise
 - generate_sim.Simulator, 92
- simulate_ideal_superres
 - generate_sim.Simulator, 88
- simulate_sim
 - generate_sim.Simulator, 88
- split
 - image_noising, 34
- ssim
 - analyse, 15
- ssim_cols
 - stats, 58
- ssim_diff_1_max
 - stats, 58
- ssim_diff_1_min
 - stats, 58
- ssim_diff_2_max
 - stats, 58
- ssim_diff_2_min
 - stats, 59
- ssim_train_epoch
 - train, 66
- ssim_val_epoch
 - train, 66
- stack
 - manage_stack, 38
- stack_number
 - manage_stack, 38
- start_epoch
 - train, 66
- stats, 50
 - alpha, 51
 - args, 52
 - ax, 52
 - choices, 52
 - color, 52
 - data, 52
 - default, 52
 - df, 52
 - dflong, 53
 - dflongssim, 53
 - dodge, 53
 - exist_ok, 53
 - fig, 53
 - figsize, 54
 - hist_range_psnr, 54
 - hist_range_ssim, 54
 - hue, 54
 - int, 54
 - legend, 54

- lw, 55
- mean_psnr_1, 55
- mean_psnr_2, 55
- mean_ssim_1, 55
- mean_ssim_2, 55
- output_dir, 55
- paired_t, 51
- palette, 55
- parents, 56
- parser, 56
- psnr_cols, 56
- psnr_diff_1_max, 56
- psnr_diff_1_min, 56
- psnr_diff_2_max, 56
- psnr_diff_2_min, 57
- range, 57
- required, 57
- se_psnr_1, 57
- se_psnr_2, 57
- se_ssim_1, 57
- se_ssim_2, 58
- ssim_cols, 58
- ssim_diff_1_max, 58
- ssim_diff_1_min, 58
- ssim_diff_2_max, 58
- ssim_diff_2_min, 59
- str, 59
- title, 59
- True, 59
- type, 59
- x, 59
- xlabel, 59
- y, 60
- steps_per_epoch
 - rcan.data_generator.SIM_Dataset, 83
- str
 - analyse, 15
 - apply, 19
 - convert_omx_to_czxy, 22
 - convert_omx_to_paz, 24
 - convert_slices_to_volumes, 27
 - generate_sim, 29
 - image_noising, 34
 - manage_stack, 38
 - recon_postprocess, 47
 - recon_preprocess, 50
 - stats, 59
 - train, 66
- subvolume
 - convert_slices_to_volumes, 27
- synthetic_sim, 60
- synthetic_sim.otf, 60
 - calc_psf, 60
- synthetic_sim.otf.PsfParameters, 74
 - Callable, 75
 - float, 75
 - int, 75
- threshold_norm
 - generate_sim, 28
- title
 - stats, 59
- train, 61
 - args, 62
 - ckpt, 62
 - ckpt_path, 62
 - config, 62
 - device, 63
 - exist_ok, 63
 - input_shape, 63
 - load_data_paths, 61
 - losses_train_epoch, 63
 - losses_val_epoch, 63
 - model, 63
 - n_accumulations, 63
 - ndim, 64
 - nepoch, 64
 - optimizer, 64
 - output_dir, 64
 - parents, 64
 - parser, 64
 - psnr_train_epoch, 64
 - psnr_val_epoch, 65
 - RCAN_hyperparameters, 65
 - required, 65
 - saveinterval, 65
 - scheduler, 65
 - schema, 65
 - ssim_train_epoch, 66
 - ssim_val_epoch, 66
 - start_epoch, 66
 - str, 66
 - train, 62
 - train_loader, 66
 - True, 66
 - type, 67
 - val_loader, 67
- train_loader
 - train, 66
- train_size
 - image_noising, 34
- True
 - analyse, 15
 - convert_slices_to_volumes, 27
 - manage_stack, 38
 - recon_preprocess, 50
 - stats, 59
 - train, 66
- tuple_of_ints
 - convert_slices_to_volumes, 27
 - rcan.utils, 46
- type
 - analyse, 15
 - apply, 20
 - convert_omx_to_czxy, 22
 - convert_omx_to_paz, 24
 - convert_slices_to_volumes, 27

- [generate_sim](#), [30](#)
 - [image_noising](#), [34](#)
 - [manage_stack](#), [39](#)
 - [recon_postprocess](#), [47](#)
 - [recon_preprocess](#), [50](#)
 - [stats](#), [59](#)
 - [train](#), [67](#)
- [val_loader](#)
 - [train](#), [67](#)
- [val_size](#)
 - [image_noising](#), [34](#)
- [volume](#)
 - [convert_slices_to_volumes](#), [27](#)
- [wavevectors](#)
 - [generate_sim.Simulator](#), [89](#)
- [x](#)
 - [stats](#), [59](#)
- [xlabel](#)
 - [stats](#), [59](#)
- [y](#)
 - [stats](#), [60](#)
- [z](#)
 - [generate_sim.Simulator](#), [92](#)
- [z_offset](#)
 - [generate_sim.SimulationRunner](#), [85](#)
- [z_p](#)
 - [generate_sim.Simulator](#), [92](#)