Contents

Demo **1**
- **Introduction to the demos and corresponding paper results** **1**
- **System requirements** **2**
- **Installation Guide** **2**
- **Instructions** **2**

Reproduction Introduction **3**
- Data Generation 3
- LRR, GSR, & UTV 3
- VSNR 4
- WNN 4
- DeStripe2 5
- Pygwy 5

Demo

Introduction to the demos and corresponding paper results

Demos and their Corresponding sections	Corresponding results	Methods	locations
Demo 1: Natural noise removal Section 'Visual Comparison'	Figure 2 (b) LRR noise (n) VSNR (o) UTV1 (p) UTV2 (q) GSR (r) LRR	LRR, VSNR	Run file: Demo/Visual Comparison/Demo.m Data: JR38_CoCr_zoom_ne0020.ibw
Demo 2: Noise removal on one simulated noisy image Section 'Quantitative Image Quality Comparison'	The 17 SSIM results in Figure 2: Simulated Noisy Image, SSIM=39.06 Median, SSIM=49.04 MD, SSIM=49.04 Modus, SSIM=48.15 Matching SSIM=50.71 FLT, SSIM=20.33 Poly, SSIM=53.6 TMD, SSIM=50.34 RS, SSIM=40.27 WNN, SSIM=39.06 DS2, SSIM=54.97 VSNR, SSIM=38.9 UTV1, SSIM=37.89 UTV2, SSIM=37.7 GSR, SSIM=43.61 LRR, SSIM=71.99	The whole 16 methods including 9 Gwyddion methods, WNN, Destripe2, VSNR, UTV vertical, UTV horizontal, GSR, LLR	Run file: Demo/five_pairs_comparison Data: Demo/Quantitative Image Quality Comparison/Different Strengths/Results_bmps Demo/Quantitative Image Quality Comparison/Different Strengths/ground_truth.bmp

	The SSIM and PSNR results on five simulated noisy images in Figure 4.		
Demo 3: Noise removal on a simulated noisy dataset Section 'Quantitative Image Quality Comparison'	Figure 5. Box plots for SSIM and PSNR results for the 16 denoising methods on the simulated noisy dataset		Run files: Demo/batch_PSNR_SSIM_values Demo/draw Data: Demo/Quantitative Image Quality Comparison/Simulated Dataset/

System requirements

Demo	Dependencies	Operating systems (including version numbers)	Versions the software has been tested on	Any required non-standard hardware
Demo/Visual Comparison/Demo.m	Run by Matlab	Ubuntu 20.04 Windows	Ubuntu 20.04 Windows 10 Enterprise	None
five_pairs_comparison	Linux-vdso.so.1 libdl.so.2 libz.so.1 libpthread.so.0 libc.so.6 /lib64/ld-linux-x86-64.so.2	Ubuntu 20.04	Ubuntu 20.04	None
batch_PSNR_SSIM_values		Ubuntu 20.04	Ubuntu 20.04	None
draw		Ubuntu 20.04	Ubuntu 20.04	None

Installation Guide

Demo	Installation instruction	Install Time
Demo/Visual Comparison/Demo.m	The same steps to install Matlab: https://ww2.mathworks.cn/en/?s_tid=gn_logo	The same with the install time of Matlab
five_pairs_comparison	None	Ubuntu 20.04
batch_PSNR_SSIM_values		Ubuntu 20.04
draw		Ubuntu 20.04

Instructions

Demo	How to run	Expected output	Expected run time
Demo/Visual Comparison/Demo.m	Run by Matlab	7 image results which combine ‘natural_’ and the method name as	2.98s

		their names	
five_pairs_comparison	Run the command: ./five_pairs_comparison	results_5_pairs.txt The SSIM and PSNR results on five simulated noisy images in Figure 4. The results with '2_' as beginnings are the 17 SSIM results in Figure 2.	6,768s
batch_PSNR_SSIM_values	./batch_PSNR_SSIM_values	results.txt SSIM/PSNR average and deviation results of different methods Data_ssim.npy and data_psnr.npy are for Demo draw	3m19,901s
draw	./draw	Two figures: PSNR_Boxplot.png SSIM_Boxplot.png The figure of SSIM and PSNR box plots results in five simulated noisy images.	0m6,380s

Reproduction Introduction

Data Generation

The simulated noise is generated randomly which means the reproduction results will be different with the paper.

Folders	Files	The Result Locations & Figures in the paper
Reproduction Code/Data_Generation	**Read_Save_RAW_Image.m**: Read and save raw image data	../Noisy_Data/ ../Ground_Truth_Data/ Fig. 1 conduction maps
	Generate_Noisy_Images.m: Add generated noise on the ground truth for a simulated noisy image	../Noisy_Data/Sim_Noisy_Images/ Fig. 4(b)-(f) Figure 4. The images with variable noise strengths.
	Generate_Dataset.m: 1. Create ground truth image dataset (800 images) by using random geometric transformation on the ground truth image 2. Create noisy image dataset (800 images) by adding random generated noise on the created ground truth image dataset	..\Ground_Truth_Data\Ground_truth_Dataset\ ..\Noisy_Data\Sim_Noisy_Dataset\
	Demo.m: Generate all the data by calling the 3 programs above.	

LRR, GSR, & UTV

The installation time and steps: consistent with the time of installation Matlab

Files (Reproduction Code/Comparison_Methods/LRR_GSR_UTV)	Result Locations
Demo.m: use LRR, GSR and UTV remove the noise from the 'Noisy_Data', which includes natural noisy image, simulated noisy image set and simulated noisy image dataset.	After running Demo.m you can find the files in: ../../Results/natural_noisy_image/ ../../Results/Sim_Noisy_Dataset ../../Results/Sim_Noisy_Images/
One_image_processing.m: A function that batch processes all data in the input database by using these 3 models.	
Images_batch_processing.m: A function that removes the noise of the input image using these 3 models. **JR38_CoCr_zoom_ne0020.ibw**: The 'CurrentRetrace' image in this file is the raw natural stripe noisy image. (often opened by Gwyddion software)	
./tools: ● **DenoiseGroupSparse.m, DenoiseL2ATV.m, DenoiseL2TV.m,DenoiseL2UTV1.m, DenoiseL2UTV2.m, DenoiseLowRank.m**: The functions for using these 3 noise removal algorithms. ● **IBWread.m, readIBWbinheader.m and readIBWheaders.m**: The functions for using IBW image. (Jakub Bialek (2023). Igor Pro file format (ibw) to matlab variable (from https://www.mathworks.com/matlabcentral/fileexchange/42679-igor-pro-file-format-ibw-to-matlab-variable), MATLAB Central File Exchange. Retrieved September 19, 2023.)	

VSNR

(Reproduction Code/Comparison_Methods/VSNR)

The installation time and steps: consistent with the time of installation Matlab

FIles instructions

For this method, you just need to change the directory in the Matlab scripts.

Files	Result Locations
Demo.m: use VSNR to remove the noise from the 'Noisy_Data', which includes natural noisy image, simulated noisy image set and simulated noisy image dataset. **One_image_processing.m**: A function that batch processes all data in the input database by using VSNR. **Images_batch_processing.m**: A function that removes the noise of the input image by using VSNR.	After running Demo.m you can find the files in: ../../Results/natural_noisy_image/VSNR ../../Results/Sim_Noisy_Dataset/VSNR ../../Results/Sim_Noisy_Images/VSNR

WNN

(Reproduction Code/SNRWDNN/)

The installation time and steps: consistent with the time of the deep learning environment installation

Files and running Instructions

1. Follow the official deep learning environment and make sure the machine can run programs by its GPU
2. Install pywt package
   pip install PyWavelets

3. Change the directory names *out_dir, test_dir* and *noise_dir* in the TEST part in ‘main’ python scripts and run it

DeStripe2

Reproduction Code/DeStripe2

FIles instructions

destripe2_Linux/destripe2_Linux64: The Demo files for using Destripe2

bat_png2txt.py**:** The script to transfer the png. Images into txt. Files with the required headers

bat_sh.py: The file generates the sh command line for running the *deStripe2* demo for all images in the same folders.

batch_generate_png.m: the script for transferring the txt results to png image files in the same folder.

README: the original instruction for Destripe2. For the details of other files are in this file.

Running instructions

Step 1: Put the noisy image dataset in this folder and change 'folder_names', 'output_folders' to bat_png2txt.py and then run it. You will then get the txt version of the png dataset.

Step 2, Generate the sh. file by bat_sh.py.

Step 3, Put the sh file and the demo file *destripe2_Linux/destripe2_Linux64* in the database, run the sh file and get the txt results in the same folders.

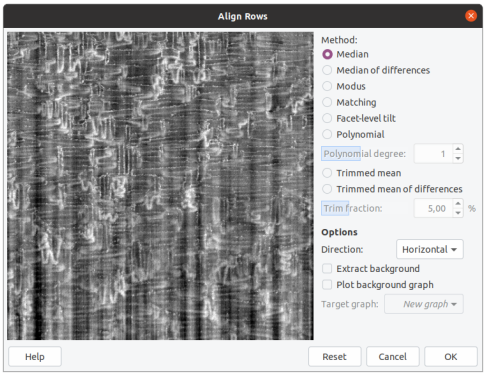
Step 4, Change the folder names to *batch_generate_png.m* and use it to transfer the TXT result to png.

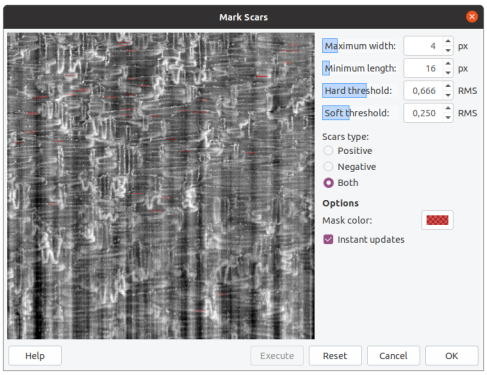
Pygwy

File instruction:

The directory of the project is Reproduction Code/pygwy/. which is for using the 9 destriping methods in Gwyddion methods:

- 8 destriping methods in ‘Align Rows’
- Another method ‘Mark Scar’ in the module ‘Data Process’





Installation instructions==========

Install time: 10+ hours

Pygwy is a Python binding to Gwyddion's objects, methods and functions. It is suitable for writing modules (process and import/export) and simplifying repetitive or complex tasks. This document contains described examples of Pygwy usage.

http://gwyddion.net/documentation/user-guide-en/pygwy.html

The Requirements==========

Python >= 2.4 (but actually Python <= 2.7 is more important)

Pygtk >= 2.10, PyGObject >= 2.12, PyCairo >= 1.2: http://www.pygtk.org/
Step-by-step Guide==========
1, Set python 2.7 to be default version to use
Changing the version of the command 'python2' is enough to handle it. Do not change command 'python3', otherwise the Linux system might be broken.

1.1 Get the entire latest version of Python:
sudo apt update
sudo apt upgrade
To list all python versions in default locations
ls /usr/bin/python*

Check:
~$ ls /usr/bin/python*

1.2 install Python 2.7 and do not remove python3!
~$ sudo apt install python 2.7

1.2 Set python 2.7 to be default version to use
Check the present default version:
~$ python -V
Python
change the python default version python 2.7:
~$ sudo update-alternatives --install /usr/bin/python python /usr/bin/python2.7 1
Check:
~$ python -V
Python 2.7.18

2, install gcc if you still didn't do yet
~$sudo apt install -y make
~$sudo apt install build-essential
https://linuxize.com/post/how-to-install-gcc-compiler-on-ubuntu-18-04/

3. Install the out-of-date packages.
Install these versions of packages which are not supported anymore. In this step you need to make sure if all the packages are correctly installed and the path of GTK is already added with correct versions. Some bugs seem like C++ grammar mistakes or file import problems that happen in Step 4 & 5 but actually it is because of the problems of this step.
But the thing is, the versions can not be too old, also, cannot be too new. You can see https://sourceforge.net/projects/gwyddion/files/pygtk-win32/ for more info about the version but the packages are for Win32.

3.1 install GTK 2.0
(Other versions cannot work!)
sudo apt-get install gtk2.0
sudo apt-get install build-essential libgtk2.0-dev
Don't forget this: export PATH="/usr/lib/x86_64-linux-gnu/pkgconfig/:$PATH"

3.2 Install Pygtk

sudo apt-get install python-dev
sudo apt-get install python-gtk2
sudo apt-get install python-gtk2-dev (still didn't install it)

3.3 ONLY one available previous PyCairo packages for Ubuntu 20.04 (cairo-1.8.10, pycairo-1.8.8):

build cairo from sources
wget http://www.cairographics.org/releases/cairo-1.8.10.tar.gz 1
tar xvf cairo-1.8.10.tar.gz
cd cairo-1.8.10
pixman_LIBS="/usr/local/lib" CPPFLAGS="-I/usr/local/include/pixman-1" LDFLAGS="-L/usr/local/lib"
./configure --prefix=/usr/local
make
make install

wget http://www.cairographics.org/releases/pycairo-1.8.8.tar.gz
tar xvf pycairo-1.8.8.tar.gz
cd pycairo-1.8.8
./configure

3.4 For PyGObject
https://download.gnome.org/sources/pygobject/2.12/
pygobject-2.12.3.tar.bz2
./configure' `make'`make check' make install'

4. Compile Gwyddion project
The simplest way to compile this package is:
./configure' `make'`make check' make install'

Try to open Gwyddion:
~$ ./gwyddion

In this step for more details you can read 'INSTALL' file in 'gwyddion-2.62'

5. Compile Pygwy

Still in the same directory:
~$ ./configure --enable-pygwy

If there are some problems then you can remove the program binaries and object files from the source code directory by typing `make clean' and try again.

6. Check and run
6.1 Pygwy Console
~$ gwyddion

If you are already successfully installed then you can see 'Pygwy' in 'Data Process'.

6.2 in the folder of 'gwy', run the demo

python Demo_gwy.py

if you see 'Failed to load module "canberra-gtk-module" from python',
then:
sudo apt install libcanberra-gtk-module libcanberra-gtk-module
sudo apt-get install libcanberra-gtk-module
Demo Files

To simplify the process, two files have been created and saved in our project:

• ‘Demo_gwy.py’: a simple demo to show successful importing of the Gwy package and opening image files.

The Destriping Files

Loading a file into a gwy container is different from other Pygwy programs, particularly if the files are not in the gwy format, which is the case with our simulated data. Understanding the gwy container and the different file formats it can read is crucial to effectively utilize Gwyddion. Additionally, this knowledge will be useful for processing data in our project in the future.

1. In the batch_code/gwy/module/pygwy directory, the following code files are useful for the tasks related to the paper:

• ’one_pair_align_rows.py‘: A simple demo processes the images in the folder ‘Demo_Data’ with only one Align Rows method and saves the result with the method name as the file name in the subfolder ‘Results_gwy’.
• ’one_pair_scar_remove.py‘: A simple demo processes the images in the folder ‘Demo_Data’ with the Scar Removal method and saves the result with the method name as the file name in the subfolder ‘Results_gwy’.
• ’batch_align_rows.py‘: This code file can align multiple single-channel image files in a folder using 8 different Align Rows methods and generate eight folders with the method name appended with "_gwy" (in .gwy format).
• ’batch_scar_remove.py‘: This code file can remove scars from multiple image files in a folder using the Scar Removal method and generate a folder named "scar_remove_gwy" (in .gwy format).
• ’find_the_gwy2png_lost_file.py‘: This code file can help recover the lost generated images in .png format.
• ’batch_gwy2png.py‘: This code file can convert the result files in .gwy format to .png format and generate 9 folders results with the method name appended with "_png" (in .gwy format).

2. Although some code was written for the paper but ultimately not used, it could potentially be useful for data processing in the future:

• datafied_operation.py allows for direct processing of two-dimensional images by specifying the channel in the datafied. The file can be saved using the gwyfile package, but for higher accuracy, it requires saving in the gwy format. In cases where the files are not in the gwy format, they must be read into a gwy container to solve the problem.
• pure_Python.py uses pure Python (gwyfile package) to read and save gwy files, which can also be saved as png. However, it was not used as it couldn't be applied to existing data_field structures. Nevertheless, the gwy2png code referenced this method.

In addition to the above, some tools that are not currently being used but may be useful in the future are also included, such as the gwyfile package and gwybatch.c (a simple standalone SPM batch data processing program written in C using Gwyddion libraries). The latter is located in the gwy_batch_c folder.

All the code used in this report can be found in the "batch_code" directory, while the resulting images and metrics (PSNR, SSIM) are stored in the "batch_results" directory alongside the corresponding code. This report will also be updated and placed in the same directory.

References

gwyddion-user-guide-en.pdf
README.pygwy
Why it is so tricky: The core problem is that the Ubuntu packages are not built with pygwy.
https://sourceforge.net/p/gwyddion/discussion/pygwy/thread/24a071efea/
Python gwy_process_func_run Examples
(https://python.hotexamples.com/examples/gwy/-/gwy_process_func_run/python-gwy_process_func_run-function-examples.html).