Install OPERA DISP-S1 data access and preparation environment

Instructions derived and modified from https://github.com/nisar-solid/ATBD/blob/main/docs/installation.md and https://github.com/OPERA-Cal-Val/calval-DISP/blob/main/docs/installation.md/

1. Install conda

```
mkdir -p /path/to/your/folder/; cd /path/to/your/folder/

# download, install and setup (mini/ana)conda
# for Linux, use Miniconda3-latest-Linux-x86_64.sh
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh -b -p
/path/to/your/folder/miniconda3
# for macOS:
curl https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-
x86_64.sh -o Miniconda3-latest-MacOSX-x86_64.sh
bash Miniconda3-latest-MacOSX-x86_64.sh -b -p
/path/to/your/folder/miniconda3
# Initialize conda on shell:
/path/to/your/folder/miniconda3/bin/conda init bash
```

Close and restart the shell for changes to take effect.

```
conda config --add channels conda-forge
conda install git mamba --yes
```

2. Install OPERA DISP-S1 tools to `opera_disp` environment

Download source code

```
cd /path/to/your/folder/
git clone TBD ## Should we put the codes on a git or just share a
compressed archive?
```

Create `opera_disp` environment and install pre-requisites

```
cd opera_disp
# create new environment
# install dependencies with mamba by using `environment.yml`
mamba env create -f environment.yml
# load the environnement disp
conda activate opera_disp
```

Source your installation

Create a file (*e.g.*: config.rc) for easy activation and loading of the paths to your files:

```
# creation of a empty file
touch /path/to/your/folder/opera_disp/config.rc
```

Add the following paths within the config.rc file:

```
##-------------- OPERA DISP ---------------------------------##
# add repo tools to your path
if [ -z ${PYTHONPATH+x} ]; then export PYTHONPATH=""; fi
export PATH="${PATH}:/path/to/your/folder/opera_disp"
export DISP_HOME=/path/to/your/folder/opera_disp
export PYTHONPATH=${PYTHONPATH}:${DISP_HOME}
```

Create an alias `load_disp` in `~/.bash_profile` file for easy activation, that call a config.rc file *e.g.*:

```
alias load_disp='conda activate opera_disp; source
/path/to/your/folder/opera_disp/config.rc'
#Close and restart the terminal or source your .bash_profile for changes
to take effect
```

3. Update the `opera_disp` environment MintPy packages

Install MintPy from source

```
# Load your environnement and paths
load_disp
cd /path/to/your/folder/
git clone https://github.com/insarlab/MintPy.git
python -m pip install -e MintPy
```

Test the installation

Run the following to test the installation:

```
load_disp
run1_download_DISP_S1_Static.py --h
smallbaselineApp.py -h
```

4. Prepare credentials or register for NASA Earthdata access

1. Register for an account with NASA Earthdata at https://urs.earthdata.nasa.gov/users/new
2. After creating the username and confirming your email, store your username/password in a `~/.netrc` file with the hostname `urs.earthdata.nasa.gov`:

```
machine urs.earthdata.nasa.gov
  login MYUSERNAME
  password MYPASSWORD
```

5. Available frames on OPERA AWS S3 bucket:

```
#FrameID,   location,       reference_lalo,
08882,     Houston,        29.692 -095.635,
11115,     Central CA,     37.104 -121.651,
11116,     Central CA,     36.612 -121.064,
12640,     Florida,        29.056 -081.263,
18903,     Rosamond,       35.039 -118.006,
28486,     Oklahoma,       35.483 -098.971,
33039,     Hawaii,         19.450 -155.525,
33065,     Unimak Isl.,    54.831 -163.781,
36542,     Central CA,     36.516 -120.853,
42779,     Alaska,         61.550 -149.327,
25018,     Alaska,         65.117 -147.433,
08622,     New York,       40.703 -073.979,
09156,     South SF,       36.293 -121.403,
```

6. Run the OPERA data downloading script:

For example, here is a sample run for the Central Valley, California case study for descending Sentinel-1 track 042. The lastest preliminary version is v0.9.

For the Frame 11116, the size of the entire dataset is ~102Gb, ~340Mb for a file. By default, the script processes all available dates, which may require substantial storage and processing time. To reduce the dataset size, you can select a specific date range using the --startDate and --endDate arguments.

```
#Args:
# --frameID --> Frame number
# --version -->
# --staticDir -->
# --geomDir -->
# --dispDir -->
# --startDate 20190101 -->
# --endDate 20200101 -->

run1_download_DISP_S1_Static.py \
      --frameID 11116 \
```

```
        --version 0.9 \
        --staticDir /path/to/your/data/directory/static_lyrs \
        --geomDir /path/to/your/data/directory/geometry \
        --dispDir /path/to/your/data/directory/data
```

7. Run the Mintpy output script

For example, here is a sample run for the Central Valley, California case study for descending Sentinel-1 track 042.

```
## Example Command to Run `run2_prep_mintpy_opera.py`
# Args:
# -m -->
# -u -->
# -o -->
# --water-mask-file -->
# --dem-file -->
# --ref-lalo -->
# --apply-mask -->

run2_prep_mintpy_opera.py \
        -m "/path/to/your/data/directory/static_lyrs" \
        -u "/path/to/your/data/directory/data/*.nc" \
        --geom-dir /path/to/your/data/directory/geometry \
        -o /path/to/your/data/directory/mintpy_output \
        --water-mask-file esa_world_cover_2021 \
        --dem-file glo_30 \
        --ref-lalo '36.612 -121.064' \
        --apply-mask
```

8. How to view the data?

In a terminal, you can visualize the timeseries.h5 newly created with the MintPy tools.

```
## Need help with the arguments: tsview.py -h
tsview.py \
    /path/to/your/timeseries.h5 \
    -m /path/to/your/recommended_mask90threshold.h5 \
```

9. Quick view of a .nc file

In the following, you can open a interactive python window within the environnement to see some

```
# Open the NetCDF file
file_path = "/path/to/your/.nc"  # Replace with your file path

# Open the NetCDF file
```

```
data = rxr.open_rasterio(file_path, masked=True)

# Inspect the dataset
data
![image](files:/Users/Desktop/Image.png)
```

```
data.connected_component_labels.plot(cmap='gray', vmin=0)
```

```
data.displacement.where(data.water_mask > 0).plot(cmap='RdBu', vmin=-.1,
vmax=.1)
```

```
# List all data variables
variables = list(data.data_vars.keys())
print(f"Variables in dataset: {variables}")

# Determine grid layout
num_vars = len(variables)
grid_cols = int(num_vars ** 0.5)
grid_rows = (num_vars // grid_cols) + (num_vars % grid_cols > 0)

# Create a figure with subplots
fig, axes = plt.subplots(grid_rows, grid_cols, figsize=(15, 10))
axes = axes.flatten()

# Loop through variables and plot
for i, var_name in enumerate(variables):
    data_ = data[var_name]
    data_.plot(ax=axes[i], cmap="viridis", add_colorbar=True)
    axes[i].set_title(var_name)

# Hide unused subplots
for j in range(num_vars, len(axes)):
    axes[j].set_visible(False)

# Adjust layout and show
plt.tight_layout()
plt.show()
```