SAR Tomography of Tropical Forests

Xiao Liu

2025-12-15

Table of contents

1 Overview

This tutorial demonstrates SAR tomography (TomoSAR) analysis of tropical forests using P-band data from the AfriSAR 2016 campaign. We'll process multi-baseline SAR data to extract 3D forest structure information and compare results with LiDAR measurements. The analysis covers data preprocessing, covariance matrix computation, tomographic inversion, and validation against airborne LiDAR forest height and biomass data.

> **i** Tutorial Background
>
> This script is developed based on the TomoSAR tutorial from EO-College.
> **Data**: F-SAR P-band TomoSAR data from AfriSAR 2016 campaign **Study site**: Lopé National Park, Gabon **Contact**: xiao.liu@mailbox.tu-dresden.de

2 Prerequisites

Before starting, you'll need to install the required Python packages and set up your working environment.

```
# Install required python packages (uncomment if needed)
# !pip install -r requirements.txt

import os
```

```
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm
from sys import exit

import warnings
warnings.filterwarnings("ignore")
```

3 Part 1: Data Preparation

3.1 Setting Up the Working Directory

Define the project paths for input data, code, and output storage:

```
# Set the project path (UPDATE THIS to your actual path)
project_path = 'E:/TomoSAR/'

inpath = os.path.join(project_path, 'data')
workspace = os.path.join(project_path, 'code')
os.chdir(workspace)
```

3.2 Importing TomoSAR Functions

Load the specialized functions for tomographic processing and visualization:

```
from tomosar_toolbox import tomobox, normalize, topo_residual_correction, \
                            covmat_downsampling
from tomosar_plotting import cov_mat_plot, tomo_plot, \
                              grouped_tomosar_profiles, tomosar_phase_centre, \
                              tomosar_layerd_reflectivity, quick_look, insar_quick_look
```

> **!** Required Functions
>
> These custom functions handle the core TomoSAR processing pipeline. Ensure the `tomosar_toolbox.py` and `tomosar_plotting.py` modules are in your workspace directory.

4 Part 2: Parameter Configuration

4.1 Defining Processing Parameters

Configure the key parameters for TomoSAR processing:

```
# Pixel spacing
ps_rg = 1.19  # Pixel spacing in range (meters)
ps_az = 0.9   # Pixel spacing in azimuth (meters)

# Define the boxcar smoothing dimension (in meters)
multi_look = 25  # Options: 25, 50

# Define the max height for the inversion
height = 65
z_vector = np.arange(-height, height + 1, 1)

# Select polarization
pol = 'hh'  # Options: 'hh', 'hv', 'vv'

# Select TomoSAR algorithm
tomo_method = 'capon'  # Options: 'capon', 'beamforming'

# Flag for terrain normalization
terrain_cor_flag = 0  # 0: without terrain normalization, 1: with terrain normalization

if terrain_cor_flag == 0:
    outpath = os.path.join(project_path, 'out')  # Save results without terrain normalizatio
else:
    outpath = os.path.join(project_path, 'out/terrain_normalised')  # Save results with terra
```

Parameter Selection

- **Multi-look size**: Larger values (50m) provide more stable estimates but lower spatial resolution
- **Polarization**: HH is most sensitive to forest structure; HV to volume scattering
- **Capon beamformer**: Provides better vertical resolution than conventional beamforming
- **Terrain correction**: Essential for accurate height estimation in sloped terrain

5 Part 3: Data Loading and Visualization

5.1 Loading SAR Data

Read the pre-processed SAR data stack:

```
outname = os.path.join(inpath, '{}_sample_data.npz'.format(pol))
data = np.load(outname)

slc_stack = data['slc_stack']      # Single Look Complex (SLC) image
kz_stack = data['kz_stack']        # Vertical wavenumber
phase_stack = data['phase_stack']  # Topographical phase

# Optional: use subset for testing
# slc_stack = slc_stack[:,:,:5]
# kz_stack = kz_stack[:,:,:5]
# phase_stack = phase_stack[:,:,:5]

del data

n_row, n_col, n_track = slc_stack.shape

print(f"Data dimensions: {slc_stack.shape}")
print(f"Number of tracks: {n_track}")
```

```
Data dimensions: (1100, 1000, 10)
Number of tracks: 10
```

> **Data Structure**
>
> The data contains:
>
> - **slc_stack**: Complex SAR image stack (rows × columns × tracks)
> - **kz_stack**: Vertical wavenumber for each baseline
> - **phase_stack**: Topographic phase for each acquisition

5.2 Quick Look at SAR Data

Visualize the SAR intensity, phase, vertical wavenumber, and topographic phase:

```
slc_id = 3
img_path = os.path.join(outpath, 'Track_{}_quick_look.png'.format(slc_id))
quick_look(slc_id, slc_stack, kz_stack, phase_stack, img_path,
           figsize=(10, 8), dpi=150, fontsize=12)
```

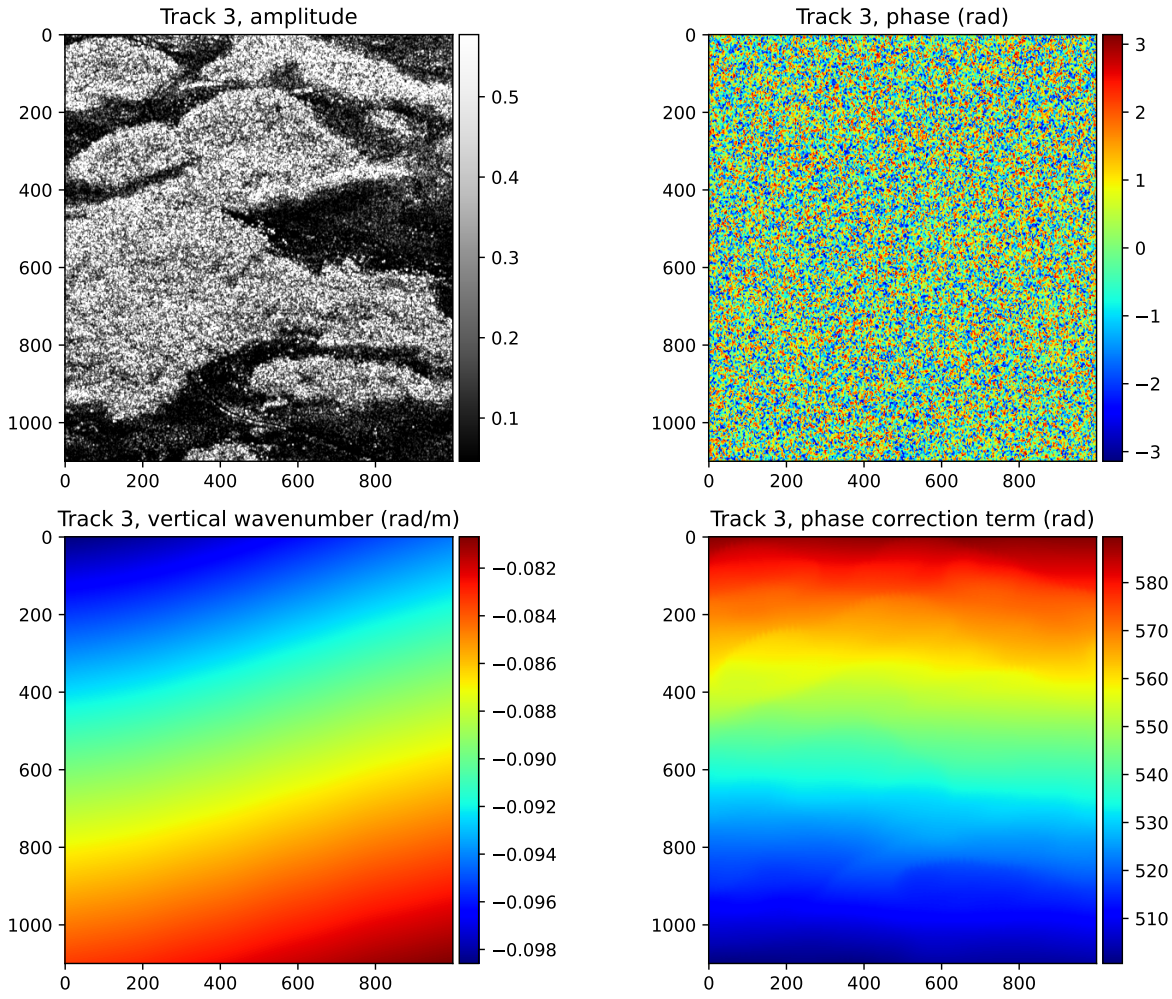

Figure 1: Quick look at SAR data: intensity, phase, kz, and topographic phase

6 Part 4: Phase Calibration

6.1 Removing Flat-Earth and Topography Phase

Calibrate the SAR data by removing the flat-earth and topographic phase contributions:

```
print('Remove flat-earth and topography phase')
outname = os.path.join(outpath, '{}_normalized_stack.npy'.format(pol))

normalized_stack = slc_stack.copy()
# Start from second SLC (kz and dem_phase of master SLC are 0)
for n in tqdm(range(1, n_track)):
    dem_phase = phase_stack[:,:,n].squeeze()
    slc_uncal = slc_stack[:,:,n].squeeze()
    normalized_stack[:,:,n] = slc_uncal * np.exp(1j * dem_phase)  # Phase calibration
del dem_phase, slc_uncal

np.save(outname, normalized_stack)
```

```
Remove flat-earth and topography phase
```

> **!** Why Phase Calibration Matters
>
> The flat-earth and topographic phases dominate the interferometric phase and must be removed to isolate the vertical structure information. This step is crucial for accurate TomoSAR reconstruction.

6.2 Removing Residual Topography Phase

For terrain-corrected processing, remove any remaining topographic phase:

```
if terrain_cor_flag == 1:
    hh_tomo_path = os.path.join(project_path + 'out/',
                                'hh_tomo_ml{}_h{}_capon.npy'.format(multi_look, height))

    if os.path.isfile(hh_tomo_path) == False:
        print('Please first calculate the HH tomography without terrain correction.')
        terrain_cor_flag = 0
        exit(0)
    else:
        print('Remove residual topography phase')
        terrain_path = os.path.join(project_path + 'out/',
                                    'tomo_ml{}_h{}'.format(multi_look, height) + '_hh_capon_t

        normalized_stack, terrain = topo_residual_correction(normalized_stack, kz_stack, z_ve
                                                             hh_tomo_path, terrain_path)
```

```
        # Plot terrain residual
        plt.figure(dpi=300)
        plt.imshow(terrain, cmap='jet', vmax=30, vmin=-30)
        plt.colorbar()
        plt.title('Topography residual (m)')
        plt.show()
```

6.3 Visualizing Phase Calibration Results

Compare the interferometric phase before and after calibration:

```
slc_1_id, slc_2_id = 1, 3
insar_quick_look(slc_1_id, slc_2_id, slc_stack, normalized_stack,
                 figsize=(12, 5), dpi=150)
```

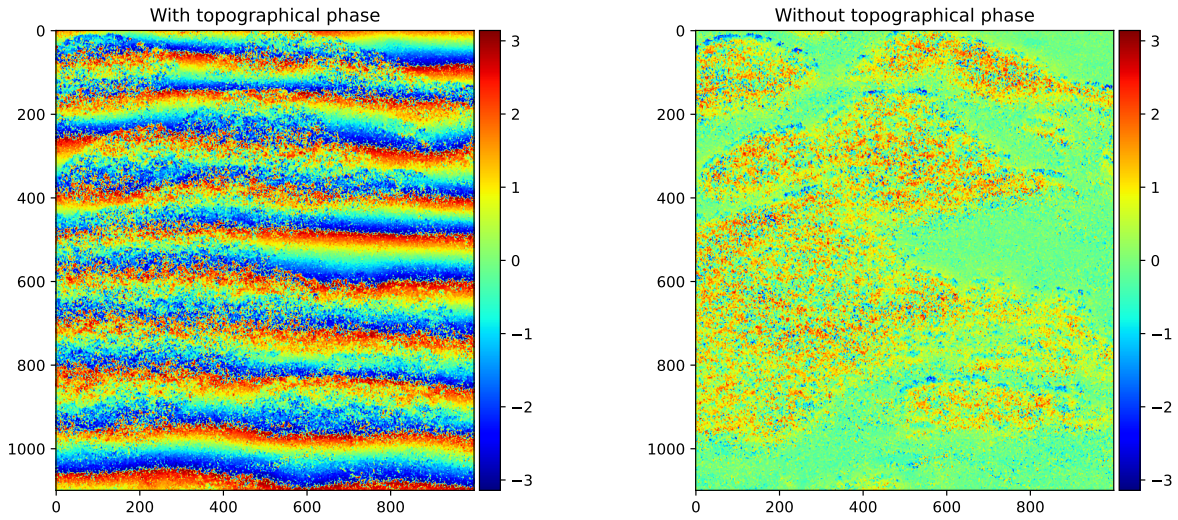

Figure 2: InSAR phase before and after removing flat-earth and topography phase

7 Part 5: Covariance Matrix Computation

7.1 Computing Covariance Matrix with Multi-looking

Calculate the covariance matrix with spatial averaging to improve signal-to-noise ratio:

```
print('Calculate covariance matrix')
covariance_matrix, r_out_smpl, x_out_smpl = covmat_downsampling(normalized_stack, multi_look
outname = os.path.join(outpath, '{}_cov_matrix_ml{}.npy'.format(pol, multi_look))
np.save(outname, covariance_matrix)

# Update kz using downsampling index along azimuth and range
kz_stack_down = kz_stack[r_out_smpl,:,:][:,x_out_smpl,:]

print(f"Covariance matrix shape: {covariance_matrix.shape}")
```

```
Calculate covariance matrix

Covariance matrix shape: (180, 139, 10, 10)
```

> **Multi-looking Trade-off**
>
> Multi-looking reduces speckle noise but decreases spatial resolution. The 25m window size balances these considerations for forest structure analysis.

7.2 Visualizing Covariance Matrix

Display the covariance matrix structure:

```
img_path = os.path.join(outpath, '{}_covariance_matrix_quick_look.png'.format(pol))
cov_mat_plot(covariance_matrix, img_path,
             figsize=(10, 10), dpi=150, fontsize=18)
```

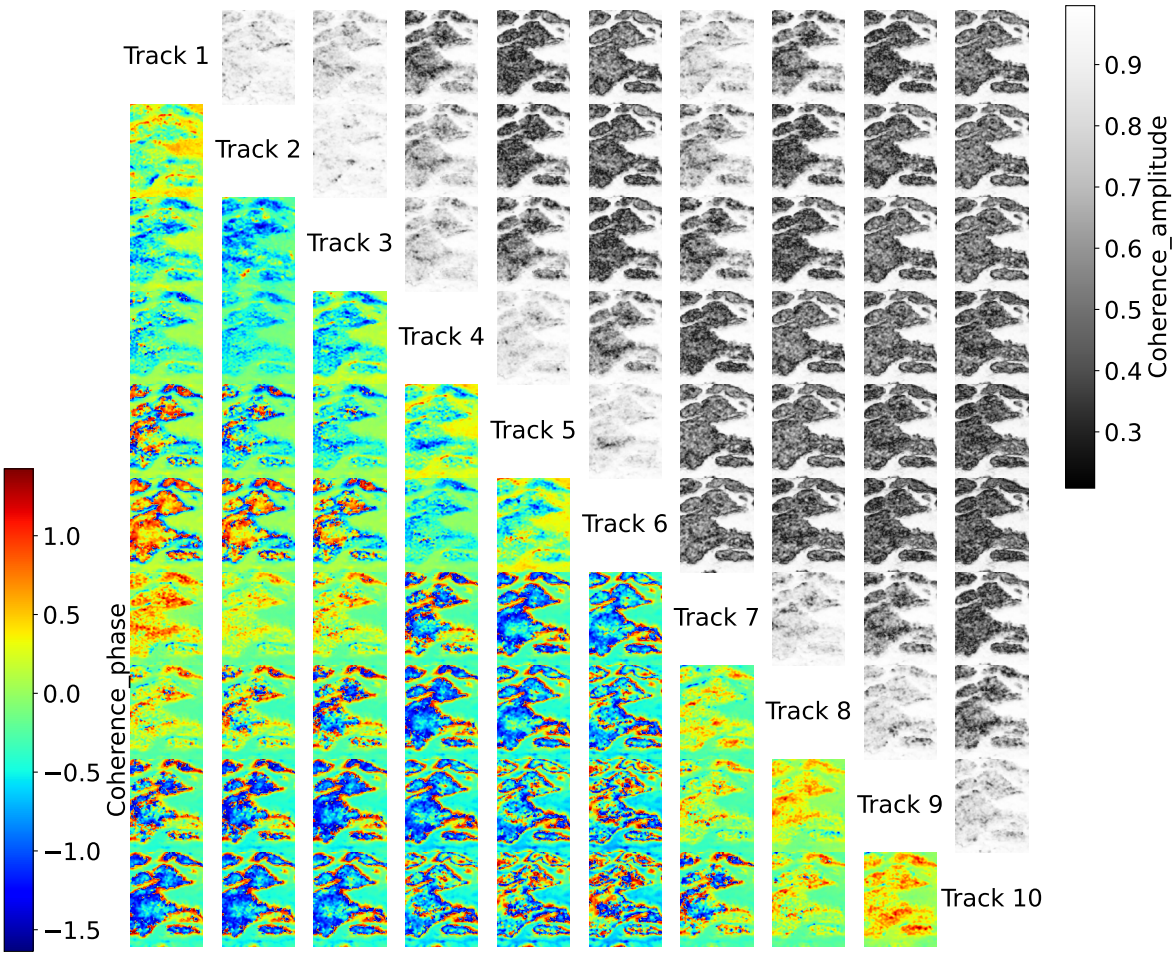

Figure 3: Covariance matrix visualization

8 Part 6: Tomographic Inversion

8.1 Computing TomoSAR Reflectivity Profiles

Perform the tomographic inversion to estimate vertical backscatter profiles:

```
tomo = tomobox(covariance_matrix, kz_stack_down, z_vector, outname, tomo_method)

outname = os.path.join(outpath, '{}_tomo_ml{}_h{}_{}.npy'.format(pol, multi_look, height, tom
np.save(outname, tomo)

print(f"TomoSAR result shape: {tomo.shape}")
```

```
TomoSAR result shape: (180, 139, 131)
```

> TomoSAR Methods
>
> - **Capon (MVDR)**: Minimum Variance Distortionless Response beamformer, provides better resolution
> - **Beamforming**: Conventional delay-and-sum beamformer, more robust but lower resolution

8.2 Normalizing Reflectivity Profiles

Normalize the tomographic reflectivity to [0, 1] for each pixel:

```
tomo_norm = np.apply_along_axis(normalize, 2, tomo)
```

8.3 Visualizing Tomographic Results

Plot example vertical reflectivity profiles:

```
rg_ratio, az_ratio = 0.5, 0.6

rg, az = int(rg_ratio * tomo_norm.shape[1]), int(az_ratio * tomo_norm.shape[0])
img_path = os.path.join(outpath, '{}_tomosar_example_az_{}_rg_{}_{}.png'.format(pol, az, rg,
tomo_plot(rg, az, slc_stack, tomo_norm, height, pol, img_path,
          figsize=(10, 8), dpi=150, fontsize=12)
```

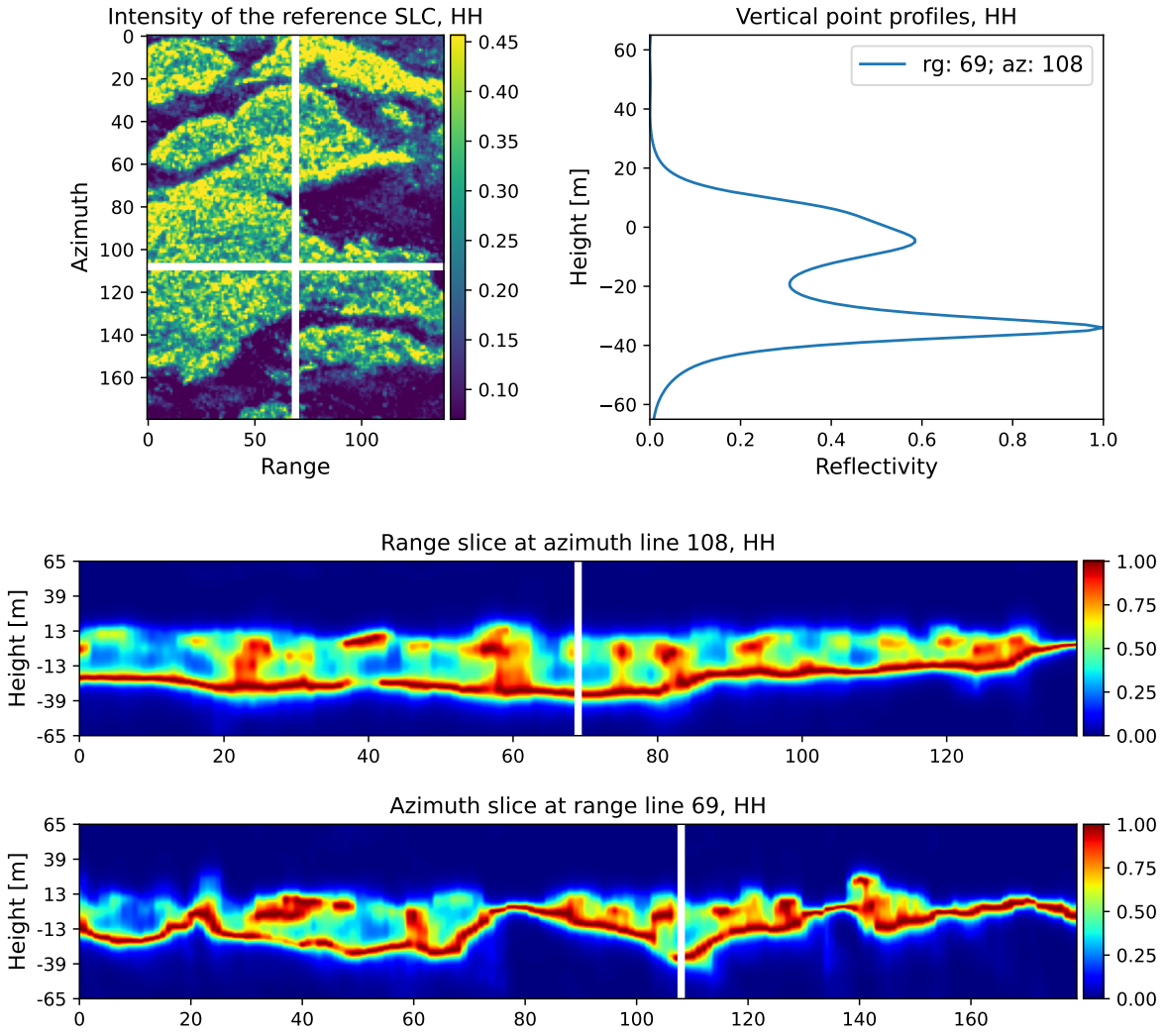

Figure 4: Example TomoSAR vertical reflectivity profile

! Interpreting Reflectivity Profiles

The vertical reflectivity profile shows:

- **Ground peak**: Strong backscatter from the ground surface
- **Canopy volume**: Distributed scattering from vegetation layers
- **Canopy top**: Upper extent of forest structure

9 Part 7: Validation with LiDAR Data

9.1 Loading LiDAR Reference Data

Load airborne LiDAR forest height and above-ground biomass data:

```
if terrain_cor_flag == 1:
    lvis_rh_path = os.path.join(inpath, 'lidar_rh100_25m.npy')
    lvis_agb_path = os.path.join(inpath, 'lidar_agb_50m.npy')

    lvis_rh = np.load(lvis_rh_path)
    lvis_agb = np.load(lvis_agb_path)

    from skimage.transform import resize
    lvis_rh = resize(lvis_rh, (tomo_norm.shape[0], tomo_norm.shape[1]))
    lvis_agb = resize(lvis_agb, (tomo_norm.shape[0], tomo_norm.shape[1]))

    print(f"LiDAR height range: {lvis_rh.min():.2f} - {lvis_rh.max():.2f} m")
    print(f"LiDAR AGB range: {lvis_agb.min():.2f} - {lvis_agb.max():.2f} Mg/ha")
```

i LiDAR Data Resolution

- **Forest height (RH100)**: 25 m resolution
- **Above-ground biomass (AGB)**: 50 m resolution

Both are resampled to match the TomoSAR grid.

9.2 Comparing TomoSAR with LiDAR Height

Overlay LiDAR forest height on TomoSAR profiles:

```
if terrain_cor_flag == 1:
    rg_ratio, az_ratio = np.random.rand(), np.random.rand()
    rg, az = int(rg_ratio * tomo_norm.shape[1]), int(az_ratio * tomo_norm.shape[0])

    img_path = os.path.join(outpath, '{}_tomosar_example_az_{}_rg_{}_lidar.png'.format(pol, a
    tomo_plot(rg, az, slc_stack, tomo_norm, height, pol, img_path, lvis_rh,
              figsize=(10, 8), dpi=150, fontsize=12)
```

9.3 Aggregating Profiles by Forest Height and Biomass

Analyze TomoSAR reflectivity patterns across different forest height and biomass classes:

```
if terrain_cor_flag == 1:
    img_path = os.path.join(outpath, '{}_tomosar_aggregated_profiles.png'.format(pol))
    grouped_tomosar_profiles(tomo, lvis_rh, lvis_agb, z_vector, img_path,
                             figsize=(11, 6), dpi=150, fontsize=12)
```

9.4 TomoSAR Phase Center vs. LiDAR Height

Compare the TomoSAR phase center with LiDAR forest height:

```
if terrain_cor_flag == 1:
    img_path = os.path.join(outpath, 'lidar_height_{}_tomosar_phase_centre.png'.format(pol))
    tomosar_phase_centre(tomo, lvis_rh, z_vector, img_path,
                         figsize=(12, 4), dpi=150)
```

> **Phase Center Height**
>
> The phase center represents the weighted center of the vertical backscatter distribution. For forests, it typically falls between the ground and canopy top, depending on penetration depth and canopy structure.

9.5 Layer-wise Reflectivity vs. Biomass

Examine TomoSAR reflectivity at different height layers in relation to biomass:

```
if terrain_cor_flag == 1:
    min_agb = 50  # Minimum AGB threshold (Mg/ha)
    img_path = os.path.join(outpath, 'lidar_agb_{}_tomosar_reflectivity_at'.format(pol))
    tomosar_layerd_reflectivity(tomo, lvis_agb, min_agb, height, img_path,
                                figsize=(12, 4), dpi=150)
```

10 Summary and Exercises

11 Key Takeaways

Data Preprocessing - Multi-baseline SAR data requires careful phase calibration - Terrain correction is essential for accurate height estimation - Multi-looking balances noise reduction and spatial resolution

Tomographic Inversion - Capon beamformer provides better vertical resolution than conventional methods - Vertical profiles reveal ground and canopy structure - Phase center height correlates with forest structure

Validation Results - TomoSAR profiles show strong agreement with LiDAR measurements - Penetration depth varies with polarization and forest density - Biomass estimation benefits from multi-layer information

12 Best Practices

- Always perform terrain correction for sloped terrain
- Use HH polarization for maximum ground penetration
- Validate results with independent reference data (LiDAR, field measurements)
- Consider multiple polarizations for comprehensive analysis
- Document all processing parameters for reproducibility

13 Exercises

Compare results using different:

1. **Polarizations**: HH, HV, VV
    - How does penetration depth vary?
    - Which polarization best correlates with biomass?
2. **Tomography methods**: Capon vs. Beamforming
    - Compare vertical resolution
    - Assess robustness to noise
3. **Multi-look sizes**: 25m vs. 50m
    - Trade-off between resolution and stability
    - Impact on height estimation accuracy
4. **Dataset size**: Full dataset vs. subset

- Effect of number of baselines
- Minimum tracks required for reliable inversion

14 Conclusion

This tutorial demonstrates a complete workflow for SAR tomography of tropical forests, from data preprocessing through validation with LiDAR. The ability to extract 3D forest structure from SAR data provides valuable information for biomass estimation, forest monitoring, and carbon accounting, especially in tropical regions where optical remote sensing is limited by cloud cover.

The integration of multiple polarizations, processing methods, and independent validation data creates a robust framework for understanding forest vertical structure. These techniques are applicable to various forest types and SAR systems, making them valuable tools for large-scale forest monitoring.

> ⚠ Important Considerations
>
> When applying these methods to your own data:
>
> - **Baseline configuration**: Ensure adequate vertical wavenumber sampling
> - **Data quality**: Check for phase unwrapping errors and temporal decorrelation
> - **Terrain effects**: Always apply terrain correction in mountainous areas
> - **Validation data**: Use concurrent LiDAR or field measurements when possible
> - **Processing parameters**: Adjust multi-look size based on forest heterogeneity

---

This tutorial provides both theoretical understanding and practical implementation for SAR tomography analysis. The modular structure allows adaptation to different datasets and research questions.