Reading and Writing Multi-band TIFFs with Rasterio

```
 import rasterio
from rasterio.transform import from_origin
import numpy as np
```

Reading a multi-band TIFF

def read_multiband_tiff(filepath): with rasterio.open(filepath) as src: # Read all bands into a 3D array (bands, height, width) data = src.read()

```
     # Or read specific bands
    band1 = src.read(1)  # Read first band

    # Read multiple specific bands
    bands = src.read([1, 2, 3])  # Read RGB bands
return data
```

Writing a multi-band TIFF

def write_multiband_tiff(output_path, data, transform, crs): # Assuming 'data' is a 3D array (bands, height, width) count = data.shape[0] # Number of bands height = data.shape[1] width = data.shape[2]

```
 with rasterio.open(
    output_path,
    'w',
    driver='GTiff',
    height=height,
    width=width,
    count=count,  # Number of bands
    dtype=data.dtype,
    crs=crs,
    transform=transform
) as dst:
    # Write all bands
    dst.write(data)

    # Or write band by band
    for band_idx in range(count):
        dst.write(data[band_idx], band_idx + 1)
```

Extracting Information from TIFF

def get_tiff_info(filepath): with rasterio.open(filepath) as src: # Image size (width, height) width = src.width height = src.height

```
    # CRS (Coordinate Reference System)
    crs = src.crs

    # GSD (Ground Sample Distance)
    # Get pixel size from transform
    pixel_size_x = src.transform[0]  # width of a pixel
    pixel_size_y = -src.transform[4]  # height of a pixel

    # Get bounds
    bounds = src.bounds

    # Get transform
    transform = src.transform

    # Get number of bands
    band_count = src.count

    info = {
        'size': (width, height),
        'crs': crs,
        'gsd': (pixel_size_x, pixel_size_y),
        'bounds': bounds,
        'transform': transform,
        'band_count': band_count
    }

    return info
```

Reprojecting Image to Another CRS

from rasterio.warp import calculate_default_transform, reproject, Resampling

```
def reproject_raster(
    src_path,
    dst_path,
    dst_crs,
    target_resolution=None,
    target_bounds=None
):
    """
    Reproject a raster to a new CRS with optional target resolution and extent

    Parameters:
    - src_path: path to source raster
    - dst_path: path for output raster
    - dst_crs: target CRS (can be EPSG code or proj4 string)
    - target_resolution: tuple of (x_res, y_res) in target CRS units
    - target_bounds: tuple of (left, bottom, right, top) in target CRS
    """

    with rasterio.open(src_path) as src:
        # Calculate transform and dimensions
        if target_bounds is None:
            # Use the default bounds (transformed from source)
            transform, width, height = calculate_default_transform(
                src.crs,
                dst_crs,
                src.width,
                src.height,
                *src.bounds,
                resolution=target_resolution
            )
        else:
            # Use specified bounds
            left, bottom, right, top = target_bounds
            left, bottom, right, top = target_bounds
            if target_resolution:
                xres, yres = target_resolution
                width = int((right - left) / xres)
                height = int((top - bottom) / yres)
                transform = from_origin(left, top, xres, yres)
            else:
                # Keep approximately same resolution as source
                src_res = src.transform[0]
                width = int((right - left) / src_res)
                height = int((top - bottom) / src_res)
                transform = from_origin(left, top, (right-left)/width, (top-bottom)/height)

        # Create destination dataset
        kwargs = src.meta.copy()
        kwargs.update({
            'crs': dst_crs,
            'transform': transform,
            'width': width,
            'height': height
        })

        with rasterio.open(dst_path, 'w', **kwargs) as dst:
            # Reproject each band
            for i in range(1, src.count + 1):
                reproject(
                    source=rasterio.band(src, i),
                    destination=rasterio.band(dst, i),
                    src_transform=src.transform,
                    src_crs=src.crs,
                    dst_transform=transform,
                    dst_crs=dst_crs,
                    resampling=Resampling.bilinear
                )
```

Usage Examples

Example usage of the functions

Read image info

```
 filepath = 'input.tif'
info = get_tiff_info(filepath)
print(f"Image size: {info['size']}")
print(f"CRS: {info['crs']}")
print(f"GSD: {info['gsd']}")
print(f"Number of bands: {info['band_count']}")
```

Reproject image

```
 reproject_raster(
    'input.tif',
    'output_reprojected.tif',
    dst_crs='EPSG:3857',  # Web Mercator
    target_resolution=(10, 10),  # 10 meter resolution
    target_bounds=(xmin, ymin, xmax, ymax)  # Optional target extent
)
```

Additional Tips

1. **Memory Management**:

   - For large files, consider reading/writing in blocks using `src.block_windows()`
   - Use `rasterio.windows.Window` for reading specific regions

2. **Compression Options**:

   # Add compression when writing

```
kwargs.update({
    'compress': 'lzw',
    'tiled': True,
    'blockxsize': 256,
    'blockysize': 256,
})
```

3. **Nodata Handling**:

   # Set nodata value when writing

```
kwargs.update({
    'nodata': -9999
})
```

4. **Common CRS Formats**:
   - EPSG codes: `'EPSG:4326'` (WGS84)
   - proj4 strings: `'+proj=utm +zone=33 +datum=WGS84'`
   - WKT strings: Use `rasterio.crs.CRS.from_wkt()`

This guide covers the basics of working with multi-band TIFFs using rasterio. The functions provided are robust and handle common use cases, but you might need to adjust parameters like *resampling method*, compression options, or block sizes depending on your specific needs.