Image Metadata Overlay

A Python tool that reads JPG images, extracts EXIF metadata (date, time, GPS location), and creates copies with configurable text overlays displaying this information.

Features

- 📷 Extracts EXIF metadata from JPG images
- 🕒 Displays date and time from image metadata
- 📍 Shows GPS location in human-readable format (e.g., 40°42'46"N, 74°0'21"W)
- 🗺️ Converts GPS coordinates to UTM or other projected coordinate systems
- 🎨 Customizable text appearance (color, size, position)
- ✨ Text outline for better visibility using native Pillow stroke API
- 🔄 Batch processing with multiprocessing (up to 6 workers by default)
- 🛡️ Preserves original EXIF metadata in output files
- 💾 Smart file collision handling (rename, skip, or overwrite)
- 📊 Progress bars for batch operations
- 📝 Comprehensive logging with file output support
- 🎯 Command-line interface with extensive options
- ✅ Dry-run mode for preview without processing

Project Structure

```
multiImageTextOverlay/
├── main.py                # Entry point - run this to process images
├── image_processor.py     # Core image processing and overlay logic
├── exif_handler.py        # EXIF metadata extraction utilities
├── config.py              # User-configurable settings with validation
├── requirements.txt       # Python dependencies
├── input/                 # Place your JPG images here (configurable)
├── output/                # Processed images will be saved here (configurable)
└── fonts/                 # TrueType font files
    └── arial.ttf          # Default font (you need to add this)
```

Installation

1. **Clone or download this project**

2. **Install Python dependencies:**

   ```
   pip install -r requirements.txt
   ```

3. **Add a TrueType font file:**

- Download a font file (e.g., Arial, Roboto, etc.) in `.ttf` format
- Place it in the `fonts/` directory
- Update `FONT_PATH` in `config.py` to match your font filename

Usage

Basic Usage

1. **Add JPG images to the `input/` folder**

2. **Run the script:**

   ```
   python main.py
   ```

3. **Find processed images in the `output/` folder**

Command-Line Options

```
# Process with default settings
python main.py

# Specify custom input/output directories
python main.py --input photos --output processed

# Customize text appearance
python main.py --position top-right --color 255 0 0 --font-size 72

# Control processing
python main.py --workers 4 --collision skip

# Enable verbose logging
python main.py --verbose

# Save logs to file
python main.py --log-file process.log

# Preview without processing
python main.py --dry-run

# Combine options
python main.py -i photos -o processed -p top-right -c 255 255 0 -s 60 -v
```

Available Options

```
-h, --help              Show help message and exit
-i, --input DIR         Input directory containing images (default: input)
-o, --output DIR        Output directory for processed images (default: output)
```

```
-p, --position POS       Text position: top-left, top-right, bottom-left, bottom-
right
-c, --color R G B        Text color as RGB values 0-255
-s, --font-size SIZE     Font size in points
-q, --quality QUALITY    Output JPEG quality 1-100
-w, --workers N          Maximum number of parallel workers (max 6)
--collision MODE         File collision handling: overwrite, skip, rename
--dry-run                Preview files without processing
-v, --verbose            Enable debug logging
--quiet                  Suppress console output except errors
--log-file FILE          Save logs to specified file
```

Configuration Options

Edit `config.py` to customize default settings:

Directory Settings

- `INPUT_DIR`: Default input directory (default: "input")
- `OUTPUT_DIR`: Default output directory (default: "output")

Text Appearance

- `TEXT_COLOR`: RGB tuple for text color (default: (255, 255, 255) - white)
- `OUTLINE_COLOR`: RGB tuple for outline color (default: (0, 0, 0) - black)
- `OUTLINE_WIDTH`: Outline thickness in pixels (default: 2)

Font Settings

- `FONT_SIZE`: Font size in points (default: 96)
- `FONT_PATH`: Path to TrueType font file (default: "fonts/arial.ttf")

Text Positioning

- `TEXT_POSITION`: Corner placement - `'top-left'`, `'top-right'`, `'bottom-left'`, `'bottom-right'`
- `PADDING`: Distance from image edge in pixels (default: 20)

Output Settings

- `OUTPUT_QUALITY`: JPEG quality 1-100 (default: 95)

Coordinate System Settings

- `SHOW_UTM_COORDINATES`: Enable/disable UTM coordinate display (default: True)
- `TARGET_EPSG`: Target EPSG code for coordinate transformation (default: 25832 - UTM Zone 32N)
- `UTM_ZONE`: UTM zone number for display (default: 32)
- `UTM_HEMISPHERE`: UTM hemisphere, 'N' or 'S' (default: 'N')

Processing Settings

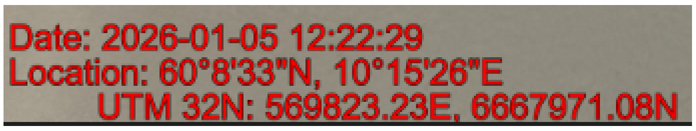
- `MAX_WORKERS`: Maximum number of parallel workers (default: 6)

- `FILE_COLLISION_MODE`: How to handle existing files - 'overwrite', 'skip', 'rename' (default: 'rename')

Example Output

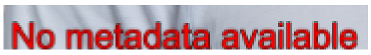
The overlay will display metadata like:

```
Date: 2024-08-15 14:30:22
Location: 40°42'46"N, 74°0'21"W
UTM 32N: 123456.78E, 987654.32N
```



Date: 2026-01-05 12:22:29
Location: 60°8'33"N, 10°15'26"E
UTM 32N: 569823.23E, 6667971.08N

If an image has no metadata, it will display: "No metadata available"



No metadata available

Coordinate System Conversion

The tool supports automatic conversion of GPS coordinates (WGS84) to UTM or other projected coordinate systems:

- **WGS84 to UTM conversion**: GPS coordinates are automatically transformed to UTM coordinates
- **Customizable target CRS**: Configure any EPSG code in `config.py` (e.g., 25832 for UTM Zone 32N, 25833 for UTM Zone 33N)
- **Display both formats**: Shows both degree-minute-second format and UTM coordinates on the image
- **Efficient caching**: Coordinate transformers are cached per process to optimize batch operations
- **Error resilient**: Falls back gracefully if coordinate transformation fails

The coordinate conversion uses the **pyproj** library, which provides accurate transformations between different coordinate reference systems based on PROJ definitions.

Dependencies

- **Pillow (PIL) >= 10.0.0**: Image processing and text rendering
- **piexif >= 1.1.3**: EXIF metadata extraction
- **tqdm >= 4.65.0**: Progress bars for batch processing
- **pyproj >= 3.0.0**: Coordinate system transformations (WGS84 to UTM/other CRS)

Advanced Features

Multiprocessing

The tool automatically uses up to 6 CPU cores for parallel processing of images, significantly speeding up batch operations. You can adjust this with the `--workers` option.

EXIF Preservation

Original EXIF metadata is preserved in processed images, including camera settings, GPS data, and timestamps.

File Collision Handling

- **rename** (default): Adds a counter suffix to avoid overwriting (e.g., image_1.jpg, image_2.jpg)
- **skip**: Skips processing if output file already exists
- **overwrite**: Replaces existing files

Logging

- Console logging with INFO level by default
- `--verbose` enables DEBUG level logging with timestamps
- `--quiet` suppresses all output except errors
- `--log-file` saves complete logs to a file for review

Error Handling

Robust error handling with specific exception catching for:

- Invalid image files
- Corrupted EXIF data
- Missing fonts
- File I/O errors
- Invalid GPS coordinates

Notes

- Only JPG/JPEG images are currently supported
- Images without EXIF data will still be processed but show "No metadata available"
- GPS coordinates are displayed in degrees, minutes, seconds format
- Configuration is validated at startup to catch errors early
- Font fallback mechanism tries multiple system fonts if custom font fails
- Original images in the `input/` folder are not modified

License

This project is open source and available for personal and commercial use.