🌊 Underwater Video & Image Color Correction

A Python-based tool for automatic color correction of underwater videos and images filmed with GoPro Hero 3 (and compatible cameras) using a red physical filter. This project provides both a command-line interface and a user-friendly GUI.

📋 Table of Contents

✨ Features

- **Automatic White Balance Correction**: Compensates for blue-green color cast caused by water
- **Red Channel Enhancement**: Restores red tones absorbed by water
- **Adaptive Contrast Enhancement**: Uses CLAHE algorithm for improved visibility
- **Gamma Correction**: Brightens mid-tones for better depth perception
- **Optional Denoising**: Reduces sensor noise in low-light conditions
- **Audio Preservation**: Maintains original audio track (requires FFmpeg)
- **Real-time Progress Tracking**: Visual feedback during processing
- **User-friendly GUI**: Intuitive Tkinter interface with sliders and checkboxes
- **Batch Processing Ready**: Can be integrated into automated workflows

🔧 Prerequisites

Required Software

1. **Python 3.12+** (or Python 3.8+)

- Download from python.org

2. **FFmpeg** (for audio preservation)
    - **Windows**: Download from ffmpeg.org
        - Extract files and add to PATH
    - **macOS**: Install via Homebrew

```bash
brew install ffmpeg
```

- **Linux (Ubuntu/Debian)**:

```bash
sudo apt-get update
sudo apt-get install ffmpeg
```

Verify FFmpeg Installation

```bash
ffmpeg -version
```

If this command displays version information, FFmpeg is properly installed.

Installation

Step 1: Clone or Download the Project

```bash
# Option 1: Clone with git
git clone https://github.com/yourusername/underwater-color-correction.git
cd underwater-color-correction

# Option 2: Download and extract the ZIP file
```

Step 2: Install Python Dependencies

```bash
```

```
# Create a virtual environment (recommended)
python -m venv venv

# Activate virtual environment
# On Windows:
venv\Scripts\activate
# On macOS/Linux:
source venv/bin/activate

# Install required packages
pip install opencv-python numpy Pillow
```

Alternative: Install from requirements.txt

Create a `requirements.txt` file:

```
txt

opencv-python>=4.8.0
numpy>=1.24.0
Pillow>=10.0.0
```

Then install:

```
bash

pip install -r requirements.txt
```

🚀 Usage

GUI Application

The easiest way to use the tool is through the graphical interface:

```
bash

python underwater_gui.py
```

GUI Workflow:

1. **Select Input Video**: Click "Browse..." next to "Input Video" and select your MP4 file
2. **Choose Output Location**: Click "Browse..." next to "Output Video" (or accept the auto-suggested filename)
3. **Adjust Parameters**:

- **Color Correction Intensity**: Slide from 0.0 (no correction) to 2.0 (aggressive)
- **Contrast Enhancement**: Slide from 1.0 (minimal) to 4.0 (maximum)
- **Enable Denoising**: Check box if you want noise reduction (slower processing)

4. **Start Processing**: Click "🎬 Start Correction"
5. **Wait for Completion**: Progress bar shows real-time status

Command Line

For automation or batch processing, use the command-line interface:

Basic Usage

```bash
python underwater_color_correction.py input.mp4 output.mp4
```

With Custom Parameters

```bash
# Subtle correction
python underwater_color_correction.py input.mp4 output.mp4 --intensity 0.5 --contrast 1.5

# Aggressive correction with denoising
python underwater_color_correction.py input.mp4 output.mp4 --intensity 1.8 --contrast 3.5 --denoise

# No color correction, only contrast enhancement
python underwater_color_correction.py input.mp4 output.mp4 --intensity 0.0 --contrast 3.0
```

Command-Line Arguments

```
positional arguments:
  input                 Input MP4 video file
  output                Output MP4 video file

optional arguments:
  -h, --help            Show help message and exit
  --intensity FLOAT     Correction intensity (0.0-2.0, default: 1.0)
                        0.0 = no correction, 2.0 = aggressive
  --contrast FLOAT      Contrast enhancement (1.0-4.0, default: 2.0)
  --denoise             Enable denoising (slower but cleaner)
  --quiet               Suppress progress output
```

🎛️ Parameters Explained

Color Correction Intensity (0.0 - 2.0)

Controls the strength of color correction applied to the video:

- **0.0**: No color correction (original colors preserved)
- **0.5**: Subtle correction for shallow water (0-10m)
- **1.0**: Standard correction for medium depth (10-20m) - **DEFAULT**
- **1.5**: Strong correction for deeper water (20-30m)
- **2.0**: Maximum correction for very deep or murky water (30m+)

What it affects:

- White balance adjustment
- Red channel enhancement
- Blue/green reduction
- Gamma correction

Contrast Enhancement (0.0 - 4.0)

Controls the adaptive contrast enhancement using CLAHE algorithm:

- **0.0**: No contrast enhancement (original contrast preserved)
- **1.0**: Minimal contrast enhancement
- **2.0**: Standard enhancement - **DEFAULT**
- **3.0**: Strong enhancement for very low contrast scenes
- **4.0**: Maximum enhancement (may introduce artifacts)

Use cases:

- Low values (1.0-2.0): Clear water with good visibility
- Medium values (2.0-3.0): Typical underwater conditions
- High values (3.0-4.0): Murky water or low light

Denoising (Checkbox)

Applies advanced noise reduction algorithm:

- **Enabled**: Removes sensor noise, produces cleaner image
    - ⚠ Significantly slower processing (2-3x longer)

    - Recommended for videos shot in low light or high ISO
- **Disabled**: Faster processing, preserves fine details
    - Recommended for well-lit videos
    - Better for videos with lots of movement

🔬 How It Works

Color Correction Pipeline

The tool applies a series of image processing operations to each video frame:

```
Input Frame
  ↓
1. White Balance Correction
  ├─ Convert to LAB color space
  ├─ Calculate average A/B channels
  └─ Neutralize color cast
  ↓
2. Red Channel Enhancement
  ├─ Split RGB channels
  ├─ Enhance red (compensate water absorption)
  └─ Reduce blue/green dominance
  ↓
3. Contrast Enhancement (CLAHE)
  ├─ Convert to LAB color space
  ├─ Apply adaptive histogram equalization
  └─ Improve local contrast
  ↓
4. Gamma Correction
  ├─ Build lookup table
  └─ Brighten mid-tones
  ↓
5. Denoising (optional)
  ├─ Apply fast non-local means
  └─ Reduce sensor noise
  ↓
Output Frame
```

Audio Processing

When FFmpeg is available:

1. **Extract**: Audio is extracted from the original video
2. **Process**: Video frames are corrected without audio

3. **Merge**: Corrected video is merged with original audio
4. **Encode**: Final video is encoded with H.264 and AAC audio

Why These Corrections?

Underwater Color Problems:

1. **Red light absorption**: Water absorbs red wavelengths first (even with red filter)
2. **Blue-green dominance**: Short wavelengths (blue/green) penetrate deeper
3. **Low contrast**: Particles in water scatter light, reducing contrast
4. **Color cast**: All underwater footage has unnatural color balance
5. **Light attenuation**: Exponential light loss with depth

Our Solutions:

- White balance removes blue-green cast
- Red channel boost compensates absorption
- CLAHE improves local details visibility
- Gamma correction brightens darker areas
- Denoising cleans up low-light noise

🛠️ Troubleshooting

Common Issues

1. "FFmpeg not found" Warning

Problem: Audio is not preserved in output video

Solution:

- Install FFmpeg (see Prerequisites)
- Verify installation: `ffmpeg -version`
- Ensure FFmpeg is in system PATH

Workaround:

- Process video without audio
- Manually merge audio later using video editing software

2. "ModuleNotFoundError: No module named 'PIL'"

Problem: Pillow (PIL) not installed (required for GUI)

Solution:

```
bash

pip install Pillow
```

Note: Pillow is only required for the GUI application (`underwater_gui.py`). The command-line script (`underwater_color_correction.py`) works without it.

3. "ModuleNotFoundError: No module named 'cv2'"

Problem: OpenCV not installed

Solution:

```
bash

pip install opencv-python
```

4. Processing is Very Slow

Possible Causes:

- Denoising is enabled (expected behavior)
- High resolution video (4K takes longer)
- Low-end hardware

Solutions:

- Disable denoising for faster processing
- Process smaller test clips first
- Consider downscaling very high-resolution videos

4. Output Colors Look Unnatural

Problem: Correction parameters too aggressive

Solution:

- Reduce intensity (try 0.5-0.8)
- Lower contrast (try 1.5-2.0)
- Test different values on short clips

5. Video File Won't Open

Problem: Unsupported codec or corrupted file

Solution:

- Ensure file is MP4 format
- Try converting with FFmpeg:

```bash
ffmpeg -i input.mov -c:v libx264 -c:a aac input.mp4
```

Performance Tips

1. **Test on Short Clips First**: Extract 5-10 seconds to test parameters

```bash
ffmpeg -i input.mp4 -ss 00:00:10 -t 00:00:05 test_clip.mp4
```

2. **Process Overnight**: Large files take time, start before bed
3. **Use SSD**: Process files on solid-state drives for faster I/O
4. **Close Other Programs**: Free up RAM and CPU resources

📸 Examples

Example 1: Standard Correction

```bash
python underwater_color_correction.py dive_video.mp4 dive_corrected.mp4
```

- Intensity: 1.0 (default)
- Contrast: 2.0 (default)
- Best for: 10-20m depth, clear water

Example 2: Shallow Water

```bash
python underwater_color_correction.py shallow.mp4 shallow_out.mp4 --intensity 0.5 --contrast 1.5
```

- Intensity: 0.5 (subtle)
- Contrast: 1.5 (minimal)
- Best for: 0-10m depth, good lighting

Example 3: Deep/Murky Water

```bash
python underwater_color_correction.py deep.mp4 deep_out.mp4 --intensity 1.8 --contrast 3.5 --denoise
```

- Intensity: 1.8 (strong)
- Contrast: 3.5 (high)
- Denoising: enabled
- Best for: 30m+ depth, poor visibility

Example 4: Night Dive

```bash
python underwater_color_correction.py night.mp4 night_out.mp4 --intensity 1.2 --contrast 3.0 --denoise
```

- Intensity: 1.2 (moderate)
- Contrast: 3.0 (strong)
- Denoising: essential for low light
- Best for: artificial light sources

📝 Project Structure

```
underwater-color-correction/
├── underwater_color_correction.py    # Main processing script
├── underwater_gui.py                 # GUI application
├── README.md                         # This file
├── requirements.txt                  # Python dependencies
└── examples/                         # Example videos (optional)
    ├── input/
    └── output/
```

🤝 Contributing

Contributions are welcome! Areas for improvement:

- Support for other video formats (AVI, MOV, etc.)
- Batch processing multiple files
- Preset profiles for different cameras
- Machine learning-based color correction
- Real-time preview in GUI

📄 License

This project is released under the MIT License. Feel free to use, modify, and distribute.

🙏 Acknowledgments

- OpenCV library for image processing capabilities
- FFmpeg project for multimedia handling
- Underwater photography community for color correction insights

📧 Support

For issues, questions, or suggestions:

- Open an issue on GitHub
- Check existing documentation
- Test with sample clips before processing full videos

Happy Diving! 🤿