Usage Examples

This document provides detailed examples of using the Human Blur CLI Tool in various scenarios.

Table of Contents

Basic Examples

Example 1: Process a Single Photo

```
python blur_humans.py vacation.jpg
```

Output:

```
============================================================
Human Blur Tool - Background Analysis Helper
============================================================

Loading YOLO model: yolov8n.pt...
✓ Model loaded successfully

Processing single image: vacation.jpg

  Detected 2 human(s) in vacation.jpg
  ✓ Saved to vacation-background.jpg

✓ Processing completed successfully!
```

Result: Creates `vacation-background.jpg` with humans blurred out.

Example 2: Process with Custom Blur

```
# Light blur (value: 21)
python blur_humans.py photo.jpg --blur 21

# Medium blur (value: 51)
python blur_humans.py photo.jpg --blur 51

# Maximum blur (value: 99, default)
python blur_humans.py photo.jpg --blur 99
```

Example 3: Adjust Detection Sensitivity

```
# More sensitive (detects more people, may have false positives)
python blur_humans.py crowd.jpg --confidence 0.3

# Default sensitivity
python blur_humans.py crowd.jpg --confidence 0.5

# Less sensitive (only very clear detections)
python blur_humans.py crowd.jpg --confidence 0.8
```

Advanced Usage

Example 4: Use a More Accurate Model

```
# Default: Fast but less accurate
python blur_humans.py photo.jpg --model yolov8n.pt

# Balanced: Good accuracy and speed
python blur_humans.py photo.jpg --model yolov8s.pt

# High accuracy: Slower but more reliable
python blur_humans.py photo.jpg --model yolov8m.pt

# Maximum accuracy: Best for challenging images
python blur_humans.py photo.jpg --model yolov8l.pt
```

When to use different models:

- `yolov8n.pt` (nano): Batch processing, real-time needs
- `yolov8s.pt` (small): General purpose
- `yolov8m.pt` (medium): Professional photography
- `yolov8l.pt` (large): Difficult lighting or angles
- `yolov8x.pt` (extra-large): Maximum quality needed

Example 5: Combine Multiple Options

```
# Custom blur + higher confidence + better model
python blur_humans.py family_photo.jpg \
  --blur 75 \
  --confidence 0.6 \
  --model yolov8m.pt
```

Batch Processing

Example 6: Process Entire Directory

```
python blur_humans.py /path/to/photos/
```

Output:

```
============================================================
Human Blur Tool - Background Analysis Helper
============================================================

Loading YOLO model: yolov8n.pt...
✓ Model loaded successfully

Processing directory: /path/to/photos

Found 10 image(s) to process

Processing [1/10]: IMG_001.jpg
  Detected 3 human(s) in IMG_001.jpg
  ✓ Saved to IMG_001-background.jpg

Processing [2/10]: IMG_002.jpg
  Detected 1 human(s) in IMG_002.jpg
  ✓ Saved to IMG_002-background.jpg

Processing [3/10]: IMG_003.jpg
  No humans detected in IMG_003.jpg

...

============================================================
Results: 8/10 images processed successfully
============================================================
```

Example 7: Batch Process with Custom Settings

```
# Process entire directory with medium blur
python blur_humans.py ./photos/ --blur 51

# Process with stricter detection
python blur_humans.py ./photos/ --confidence 0.7

# Batch process with best quality
python blur_humans.py ./photos/ --model yolov8m.pt
```

Performance Tuning

Example 8: Speed vs Accuracy Trade-offs

```
# FASTEST: Nano model, default confidence
python blur_humans.py large_dataset/ --model yolov8n.pt
# ~100-200 images per minute (CPU)

# BALANCED: Small model, slightly higher confidence
python blur_humans.py photos/ --model yolov8s.pt --confidence 0.6
# ~50-100 images per minute (CPU)

# ACCURATE: Medium model, higher confidence
python blur_humans.py photos/ --model yolov8m.pt --confidence 0.7
# ~20-40 images per minute (CPU)

# MAXIMUM QUALITY: Large model, strict confidence
python blur_humans.py photos/ --model yolov8l.pt --confidence 0.8
# ~10-20 images per minute (CPU)
```

Note: With GPU acceleration, speeds increase 5-10x

Common Scenarios

Example 9: Real Estate Photography

Goal: Remove people from property photos while preserving architecture

```
python blur_humans.py real_estate_photos/ \
  --blur 99 \
  --confidence 0.5 \
  --model yolov8s.pt
```

Why these settings:
- Maximum blur (99) to completely remove human presence
- Default confidence (0.5) catches most people
- Small model (yolov8s.pt) for good balance

Example 10: Privacy Protection

Goal: Anonymize people in public street photography

```
python blur_humans.py street_photos/ \
  --blur 75 \
  --confidence 0.4 \
  --model yolov8m.pt
```

Why these settings:
- Medium-high blur (75) for anonymity
- Lower confidence (0.4) to catch distant/partial people
- Medium model (yolov8m.pt) for better distant detection

Example 11: Background Analysis Research

Goal: Study environmental features without human interference

```
python blur_humans.py research_images/ \
  --blur 99 \
  --confidence 0.3 \
  --model yolov8m.pt
```

Why these settings:
- Maximum blur (99) to eliminate all human features
- Low confidence (0.3) to catch even unclear human forms
- Medium model (yolov8m.pt) for reliable detection

Example 12: Crowd Scenes

Goal: Process images with multiple people

```
python blur_humans.py crowd.jpg \
  --blur 51 \
  --confidence 0.6 \
  --model yolov8m.pt
```

Why these settings:
- Medium blur (51) handles overlapping regions better
- Higher confidence (0.6) reduces false positives
- Medium model (yolov8m.pt) better at multiple detections

Shell Scripting Integration

Example 13: Automate Processing

```
#!/bin/bash
# process_all_folders.sh

# Process multiple directories
for dir in photos_2023 photos_2024 photos_2025; do
    echo "Processing $dir..."
    python blur_humans.py "$dir/" --blur 99 --model yolov8s.pt
done

echo "All folders processed!"
```

Example 14: Process Only Specific Files

```
#!/bin/bash
# process_large_images.sh

# Process only large images (over 1MB)
find ./photos -type f -size +1M \( -name "*.jpg" -o -name "*.png" \) | while read
img; do
    echo "Processing: $img"
    python blur_humans.py "$img" --blur 75
done
```

Troubleshooting Examples

Example 15: Missing Some People

Problem: Tool misses some people in the image

Solution: Lower confidence threshold

```
python blur_humans.py photo.jpg --confidence 0.3
```

Example 16: Too Many False Positives

Problem: Tool detects objects as people

Solution: Increase confidence threshold

```
python blur_humans.py photo.jpg --confidence 0.7
```

Example 17: Blur Not Strong Enough

Problem: Can still see faces/features

Solution: Increase blur intensity

```
python blur_humans.py photo.jpg --blur 99
```

Example 18: Processing Too Slow

Problem: Taking too long to process

Solution 1: Use faster model

```
python blur_humans.py large_folder/ --model yolov8n.pt
```

Solution 2: Install GPU support (if NVIDIA GPU available)

```
pip install torch torchvision --index-url https://download.pytorch.org/whl/cu118
python blur_humans.py large_folder/  # Will automatically use GPU
```

Python Integration

Example 19: Use as a Module

```python
#!/usr/bin/env python3
from pathlib import Path
from blur_humans import HumanBlurProcessor

# Initialize processor
processor = HumanBlurProcessor(
    model_name='yolov8s.pt',
    blur_intensity=75
)

# Process single image
image_path = Path('photo.jpg')
processor.process_image(image_path, confidence=0.5)

# Process directory
dir_path = Path('photos/')
successful, total = processor.process_directory(dir_path, confidence=0.5)
print(f"Processed {successful}/{total} images")
```

Example 20: Custom Processing Pipeline

```python
#!/usr/bin/env python3
import cv2
from blur_humans import HumanBlurProcessor

# Initialize
processor = HumanBlurProcessor(blur_intensity=99)

# Load image
image = cv2.imread('photo.jpg')

# Detect humans
boxes = processor.detect_humans(image, confidence=0.5)
print(f"Found {len(boxes)} people at: {boxes}")

# Apply blur
result = processor.blur_regions(image, boxes)

# Custom save location
cv2.imwrite('output/custom_name.jpg', result)
```

Tips & Best Practices

1. **Start with defaults** - Use default settings first, then adjust
2. **Test on one image** - Before batch processing, test settings on one image
3. **Balance blur** - Too much blur (99) may look unnatural; try 51-75 for natural look
4. **GPU for batch** - If processing 100+ images, GPU acceleration is highly recommended
5. **Keep originals** - Tool never modifies originals; they’re always safe
6. **Experiment with confidence** - Different scenes need different confidence levels
7. **Model selection** - Use yolov8n.pt for most cases; upgrade only if needed

---

Need more help? Check the main README.md for detailed documentation.