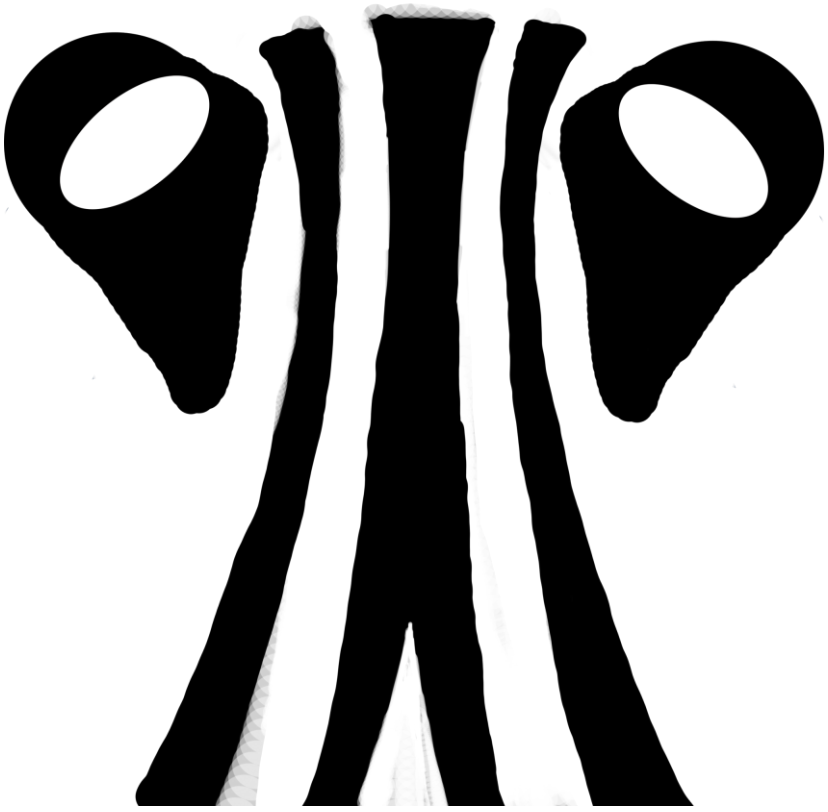


Wild Catcher

（野生動物 AI 検出＆動画管理アプリ）

ドキュメント

Diego Alonso

目次

- 動画の処理
- 検出とファイル名変更のロジック

9. **エラー対処とトラブルシューティング**
10. **付録**
    - 追加の注意事項
    - 連絡先情報

1. 概要

この Wild Catcher（野生動物検出＆処理）のマニュアルへようこそ。Wild Catcher は、画像や動画を処理し、人間、車両、動物などの対象物を検出し、メディアファイルの管理とレビューを行うためのさまざまな機能を提供します。

Wild Catcher は、主に以下の 2 つの機能で構成されています：

- **動画検出アプリ**：AI 検出モデルを使用して画像と動画を処理する GUI ベースのアプリケーション。
- **動画処理ツールアプリ**：動画ファイルを効率的に閲覧および管理するための統合動画処理ツール。

2. インストール・初期設定

システム要件

- **オペレーティングシステム**：Windows 7 以上（64 ビット推奨）
- **Python バージョン**：Python 3.6 以上
- **必要な Python パッケージ**：
  - PyQt5
  - OpenCV（cv2）
  - VLC（Python バインディング）
  - Torch
  - その他コードに応じた依存関係

インストール手順

1. **Python のインストール**：
   - 公式ウェブサイトから Python をダウンロードしてインストールします：https://www.python.org/downloads/。
2. **必要なパッケージのインストール**：
   - コマンドプロンプトを開き、以下のコマンドを実行します：

```
pip install PyQt5 opencv-python torch torchvision
pip install python-vlc
```

3. **アプリケーションのダウンロード**：
   - VideoDetectionApp と VideoPlayer のコード、および必要なアイコンや画像を含む assets フォルダをダウンロードします。
4. **VLC のセットアップ**：
   - システムに VLC メディア動画処理ツールがインストールされていることを確認します。
   - コードで指定されているように（vlc フォルダ）、VLC ライブラリをアプリケーションディレクトリにコピーします。
5. **アプリケーションの実行**：

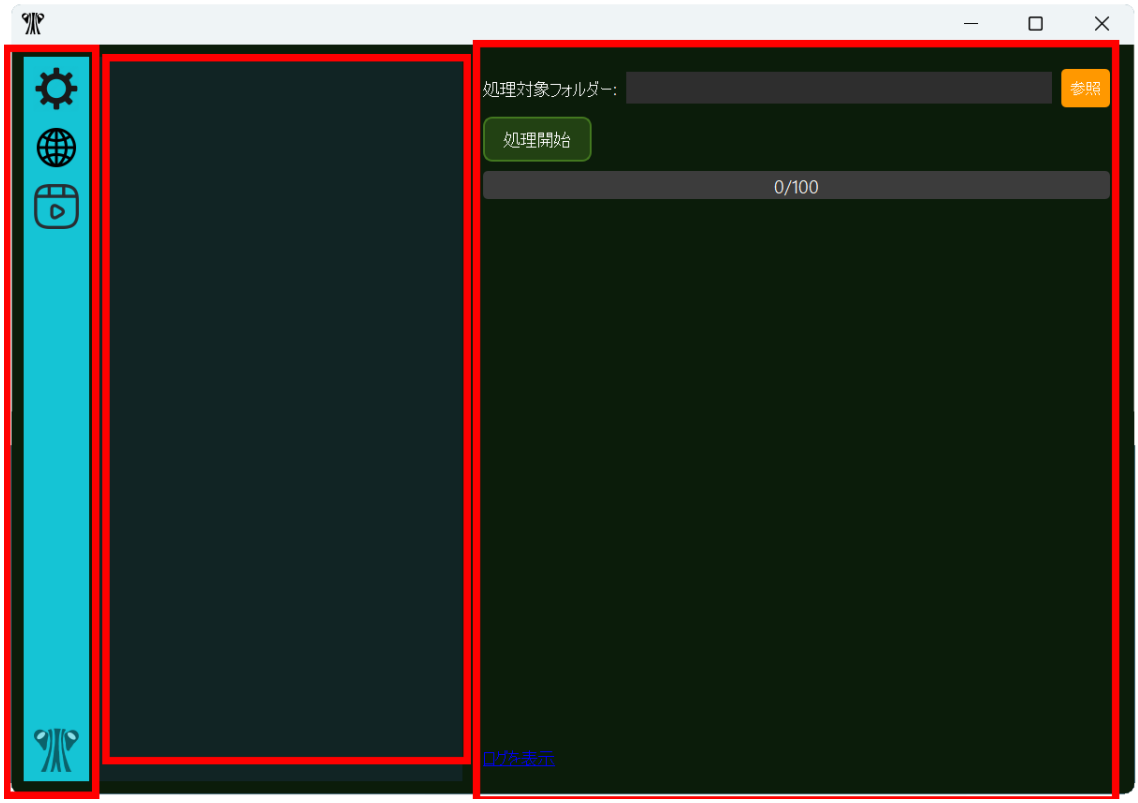
```
Python  wild_catcher_app.py
```


3. インターフェース

概要

動画検出アプリを起動すると、ユーザーフレンドリーなインターフェースが表示され、メディアファイルの処理が簡単に行えます。インターフェースは主に以下の 3 つのセクションで構成されています：

- **サイドバー**：設定、言語オプション、動画処理ツールへのクイックアクセスを提供します。
- **設定パネル**：さまざまな処理パラメータを設定できます。
- **メインエリア**：入力フォルダの選択、処理の開始、ログの表示など、主要な操作を行います。



サイドバー

サイドバーはアプリケーションウィンドウの左側に位置し、以下の要素が含まれています：

設定ボタン

- **アイコン**:歯車アイコン(settings_icon.png)
- **機能**:設定パネルの表示/非表示を切り替えます。
- **使用方法**:クリックして処理パラメータを調整する設定を表示または非表示にします。

言語ボタン

- **アイコン**:地球儀アイコン(language_icon.png)
- **機能**:言語オプションパネルを開きます。
- **使用方法**:クリックしてアプリケーションの言語を変更します。

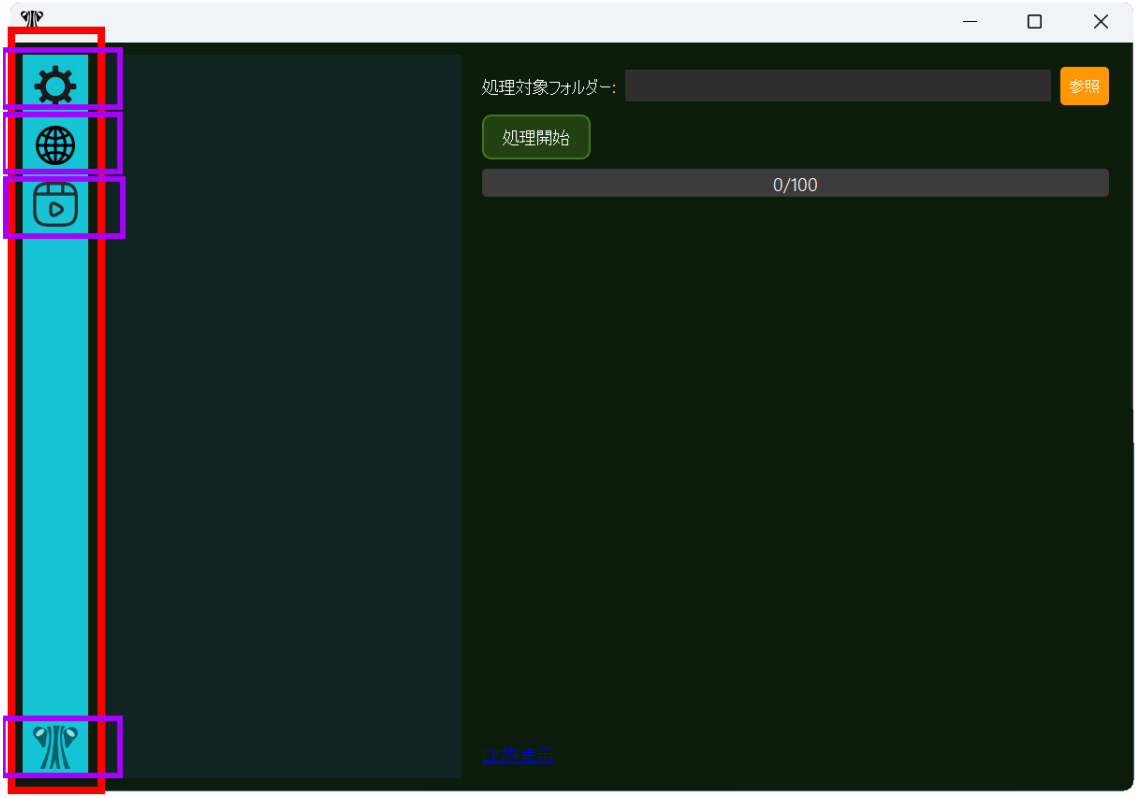
動画処理ツールボタン

- **アイコン**:再生アイコン(player_icon.png)
- **機能**:動画処理ツールアプリを起動します。
- **使用方法**:クリックして統合動画処理ツールを開きます。

アプリアイコン

- **アイコン**:アプリケーションのロゴ(app_icon.png)
- **位置**:サイドバーの一番下に配置されています。
- **外観**:半透明(50%の不透明度)でクリック不可。

- **機能**：ビジュアルなブランド識別子として機能します。



設定パネル

- **表示**：サイドバーの**設定ボタン**を使用して切り替え可能です。
- **内容**：処理動作を設定するためのさまざまなオプションが含まれています。
- **レイアウト**：上部に揃えられ、コントロールが縦方向に配置されています。

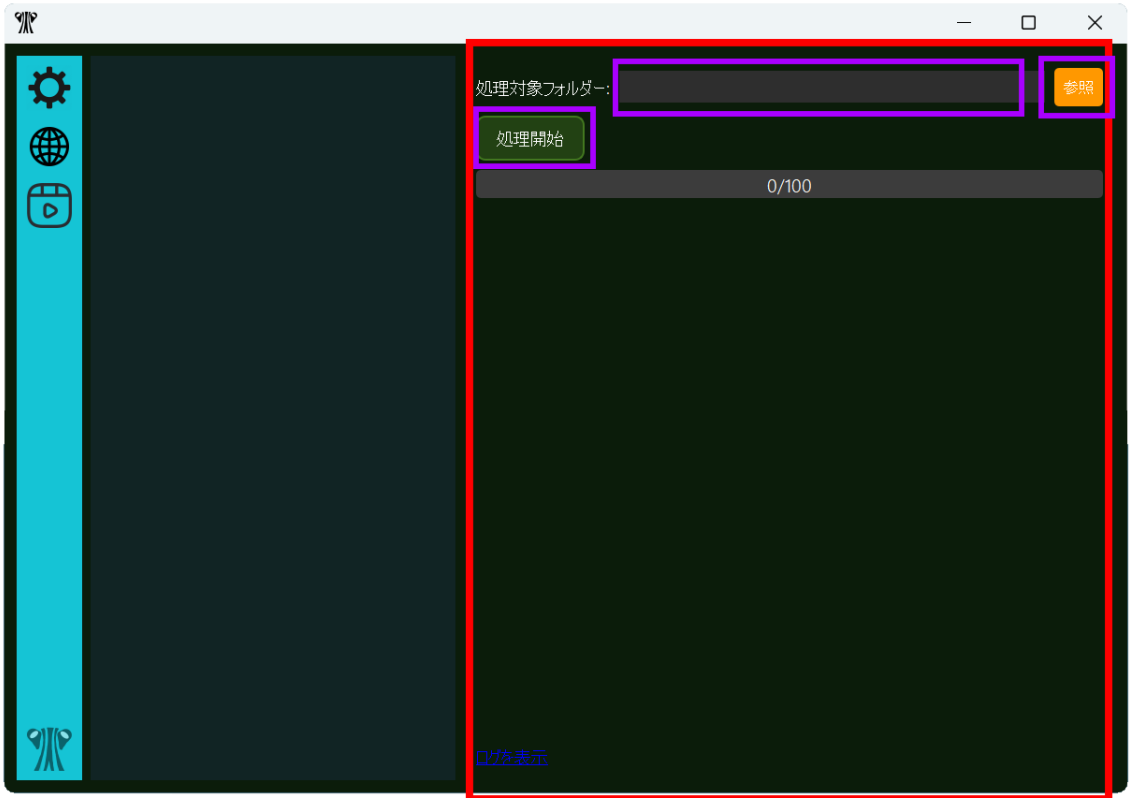
メインエリア

メインエリアは、アプリケーションの主要な機能を操作する場所です。

入力ディレクトリの選択

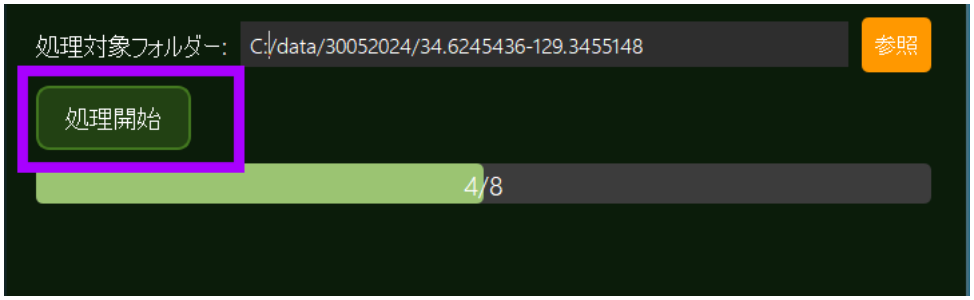
- **構成**：
  - **ラベル**：入力フォルダを指定する場所を示します。
  - **テキストフィールド**：選択したフォルダのパスを表示します。
  - **参照ボタン**：入力フォルダを選択するダイアログを開きます。
- **機能**：処理する画像と動画を含むフォルダを指定します。

- **使用方法**:
    1. **参照**ボタンをクリックします。
    2. ダイアログで希望のフォルダを選択します。
    3. フォルダのパスがテキストフィールドに表示されます。



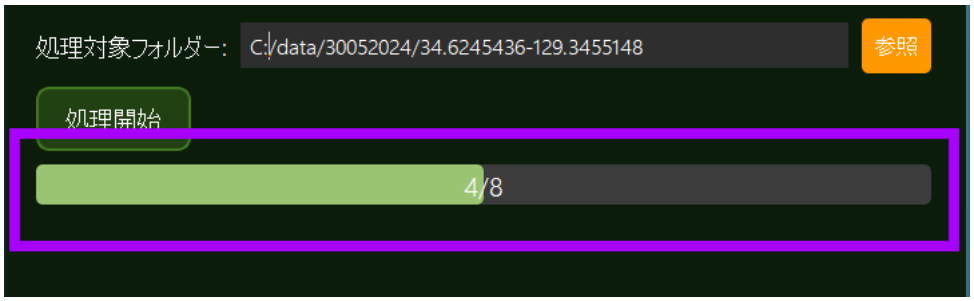
処理開始ボタン

- **ラベル**: "処理開始"
- **機能**: 指定した入力フォルダ内のメディアファイルの処理を開始します。
- **外観**: パディングと目立つ背景色でスタイリングされています。
- **使用方法**: 設定を調整した後、クリックして処理を開始します。



プログレスバー

- **機能**：処理の進行状況を表示します。
- **詳細**：
  - 処理されたファイル数と合計ファイル数を表示します。
  - 処理中に動的に更新されます。
- **外観**：進行状況を示すカラーの部分でデザインされています。

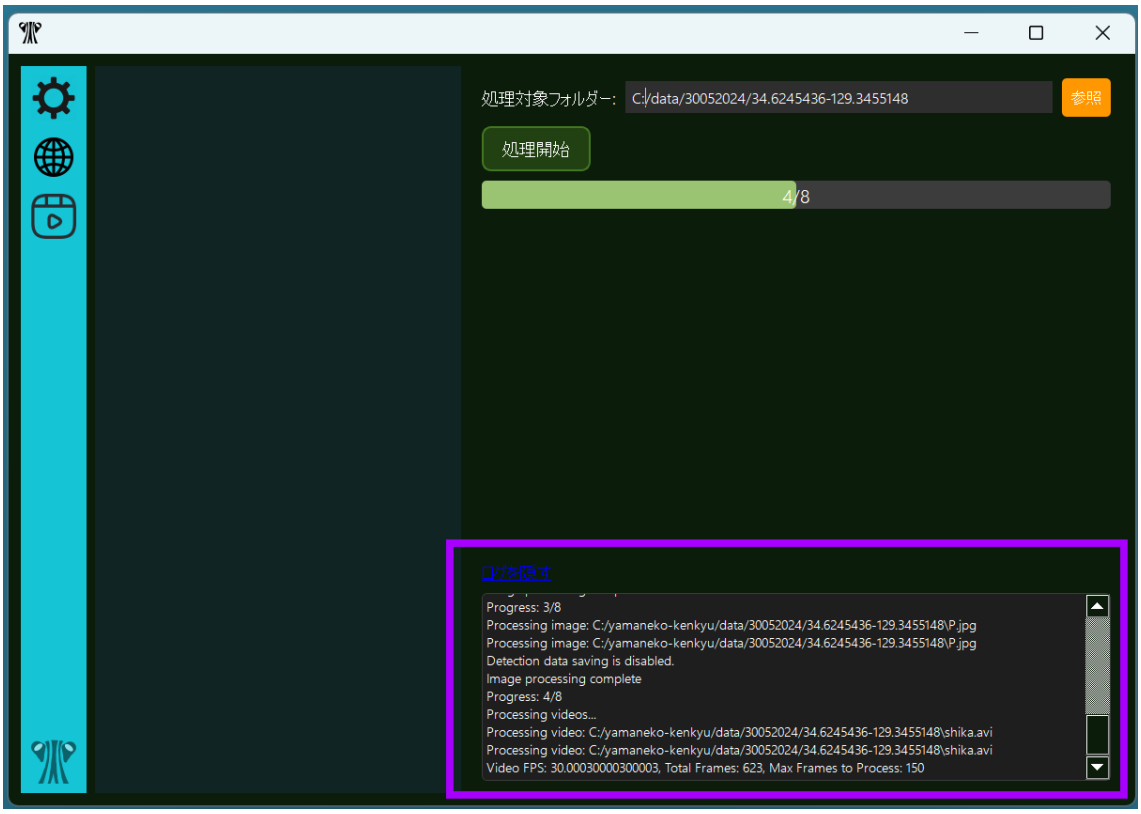


ログ表示ラベル

- **ラベル**：″ログを表示″ または ″ログを隠す″（表示状態に応じて）
- **機能**：ログテキストエリアの表示/非表示を切り替えます。
- **使用方法**：クリックして処理操作の詳細なログを表示または非表示にします。

ログテキストエリア

- **機能**：処理中に生成されたログメッセージを表示します。
- **特徴**：
  - 読み取り専用のテキストエリア。
  - スペースを節約するために初期状態では非表示。
- **使用方法**：**ログを表示**ラベルを使用してこのエリアを表示または非表示にします。



4. 設定パネルの詳細

設定パネルが表示されている場合、アプリケーションがメディアファイルを処理する方法に影響するさまざまなパラメータを設定できます。

フレーム間隔

- **コントロール**：スピンボックス（数値入力）
- **ラベル**：″フレーム間隔″
- **機能**：動画のフレームを処理する際にスキップするフレーム数を指定します。
- **デフォルト値**：16
- **使用方法**：n 番目ごとのフレームを処理するための値を設定します（例：16 を設定すると、16 フレームごとに AI で動物検出の処理を行います）。

確信度のしきい値

- **コントロール**：ダブルスピンボックス（小数点入力）
- **ラベル**：″確信度のしきい値″
- **機能**：検出が有効と見なされる最小の確信度レベルを設定します。
- **範囲**：0.0 から 1.0
- **デフォルト値**：0.4
- **使用方法**：特定の確信度レベル以下の検出をフィルタリングするために調整します。

動画の処理時間（秒）

- **コントロール**：スピンボックス（数値入力）
- **ラベル**：″動画の処理時間（秒）″
- **機能**：各動画の最初の指定された秒数までの処理を制限します。
- **デフォルト値**：5 秒
- **使用方法**：長い動画の一部のみを処理する際に便利です。

「detection_data」フォルダを作成

- **コントロール**：チェックボックス
- **ラベル**：″'detection_data' フォルダを作成″

- **機能**：検出画像やクロップ画像を別フォルダに保存するかどうかを決定します。
- **使用方法**：チェックを入れて検出データの保存を有効にします。

検出がない動画を削除

- **コントロール**：チェックボックス
- **ラベル**：″検出がない動画を削除″
- **機能**：チェックを入れると、有効な検出がない動画が処理後に削除されます。
- **使用方法**：ストレージスペースを管理するために注意して使用します。

すべてのフレームを保存

- **コントロール**：チェックボックス
- **ラベル**：″すべてのフレームを保存″
- **機能**：確信度が最も高かったフレームだけでなく、検出されたすべてのフレームを保存します。
- **使用方法**：さらなる分析のために検出されたすべてのフレームを保持するためにチェックを入れます。

タグでファイル名を変更

- **コントロール**：チェックボックス
- **ラベル**：″タグでファイル名を変更″
- **機能**：検出カテゴリに基づいて指定されたタグを追加してファイル名変更します。
- **使用方法**：ファイルの自動ファイル名変更を有効にするためにチェックを入れます。

人・車のタグ

- **コントロール**：テキスト入力（ラインエディット）
- **ラベル**：″人・車のタグ″
- **デフォルト値**：″hito_″
- **機能**：人間や車両が検出されたときにファイル名に追加するプレフィックスを指定します。
- **使用方法**：希望に応じてタグをカスタマイズします。

動物のタグ

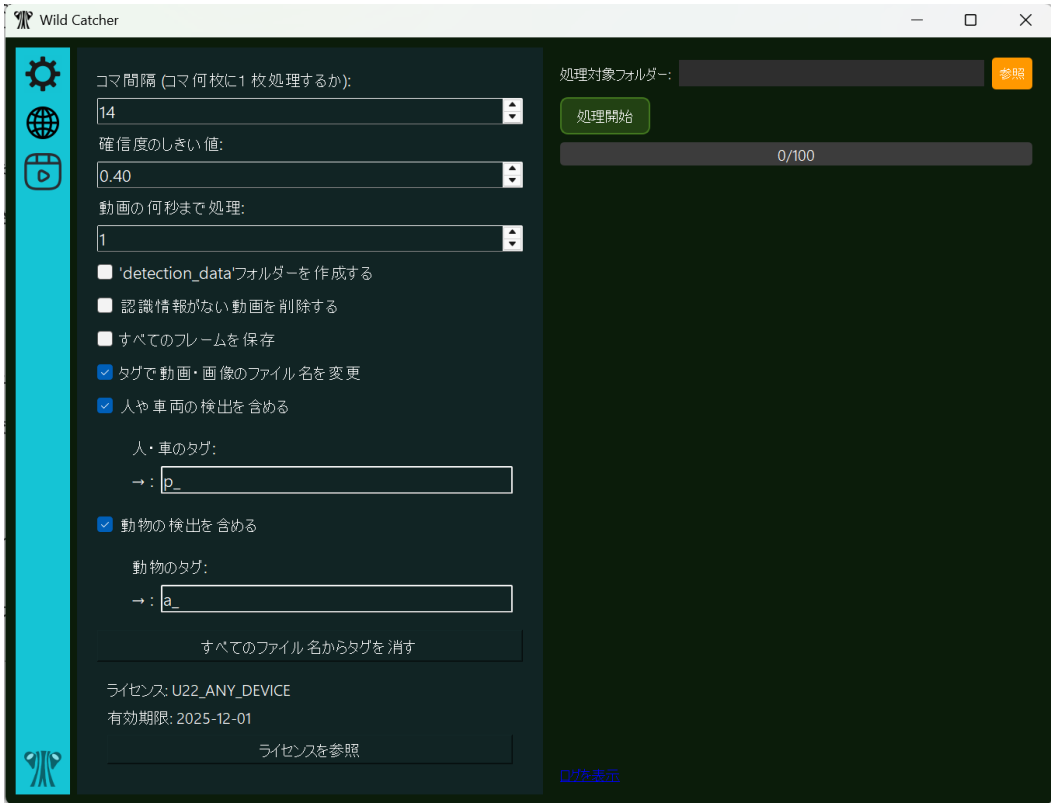
- **コントロール**：テキスト入力（ラインエディット）
- **ラベル**：〝動物のタグ〟
- **デフォルト値**：〝animal_〟
- **機能**：動物が検出されたときにファイル名に追加するプレフィックスを指定します。
- **使用方法**：希望に応じてタグをカスタマイズします。

すべてのファイル名からタグを削除

- **コントロール**：ボタン
- **ラベル**：〝すべてのファイル名からタグを削除〟
- **機能**：選択した入力フォルダ内のファイル名から指定されたタグを削除します。
- **使用方法**：処理後にファイル名をクリーンアップするためにクリックします。

ライセンス情報

- **ライセンス名**
- **ライセンスの有効期限**



5. 言語オプション

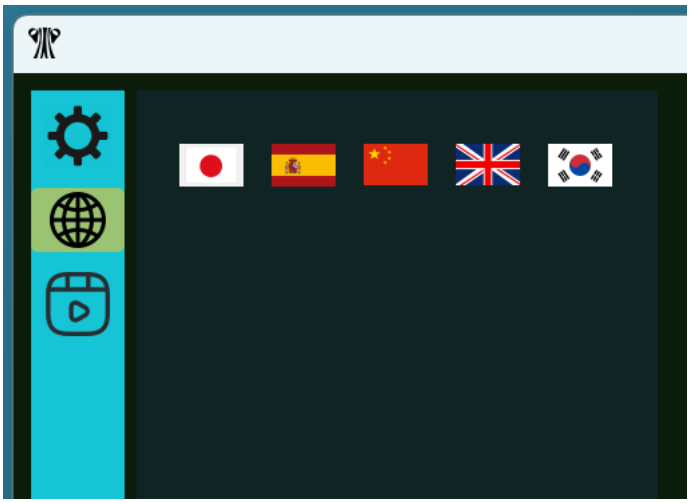
Wild Catcher は、多様なユーザーベースに対応するために複数の言語をサポートしています。

利用可能な言語

- 日本語
- スペイン語
- 中国語
- 英語
- 韓国語

言語の変更

1. サイドバーの**言語ボタン**をクリックします。
2. 言語オプションパネルが表示され、各言語を表す国旗が表示されます。
3. 希望の言語に対応する国旗をクリックします。
4. アプリケーションのすべてのラベルとテキストが選択した言語に更新されます。



6. 処理機能

処理の開始

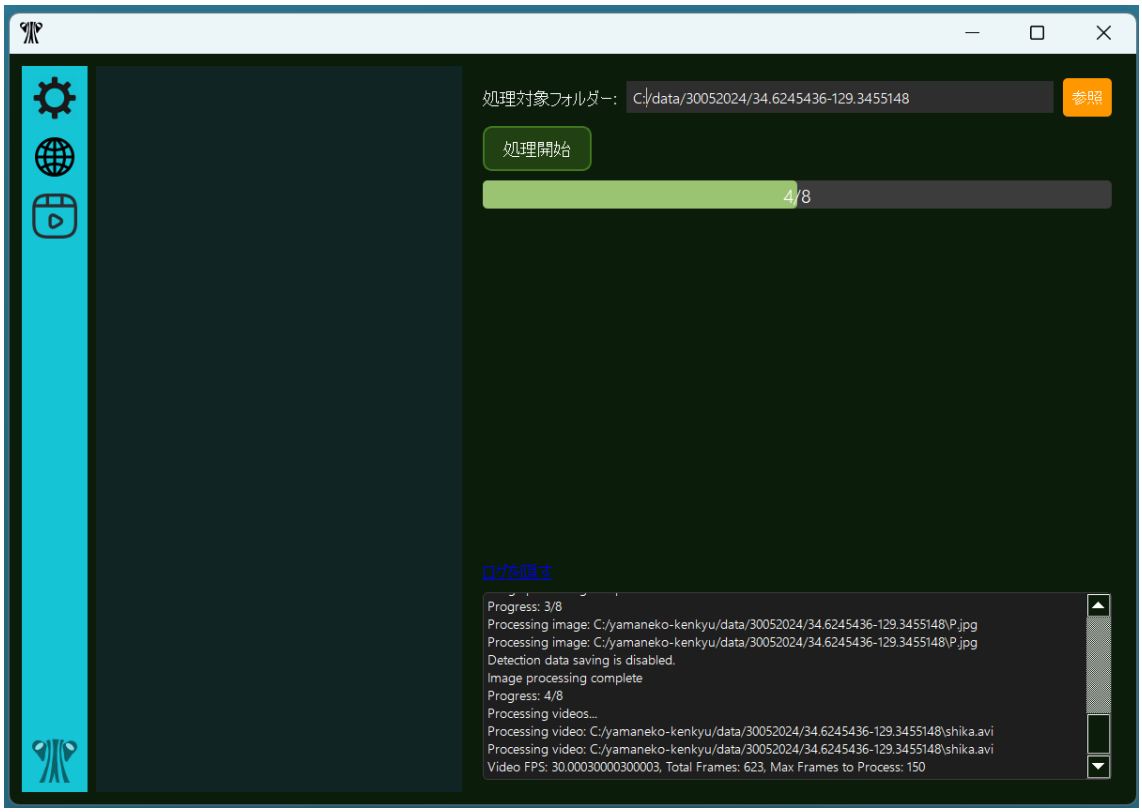
1. メディアファイルを含む入力フォルダを選択したことを確認します。
2. 必要に応じて設定パネルで設定を調整します。
3. メインエリアの**処理開始**ボタンをクリックします。
4. アプリケーションが処理を開始し、プログレスバーが適宜更新されます。
5. ログメッセージは表示されていればログテキストエリアに表示されます。

処理中に何が起こるか

- アプリケーションは、指定されたプレフィックスを既に持つファイルを除外して、入力フォルダ内の画像と動画をスキャンします。
- 画像と動画は AI 検出モデルを使用して処理されます。
- 確信度のしきい値やその他の設定に基づいて検出が行われます。
- 設定に応じて、アプリケーションは以下を行う場合があります:
    - タグでファイル名を変更します。
    - detection_data フォルダに検出データを保存します。
    - 有効な検出がないファイルを削除します。

ログメッセージ

- 処理状況に関するリアルタイムのフィードバックを提供します。
- 以下の情報を含みます:
    - 処理中のファイル。
    - 行われた検出。
    - 実行された操作(例:ファイル名変更、削除)。
    - エラー等
- トラブルシューティングや操作の確認に役立ちます。


処理対象フォルダー:
C:/data/30052024/34.6245436-129.3455148
参照
処理開始
4/8
ログを隠す
Progress: 3/8
Processing image: C:/yamaneko-kenkyu/data/30052024/34.6245436-129.3455148\P.jpg
Processing image: C:/yamaneko-kenkyu/data/30052024/34.6245436-129.3455148\P.jpg
Detection data saving is disabled.
Image processing complete
Progress: 4/8
Processing videos...
Processing video: C:/yamaneko-kenkyu/data/30052024/34.6245436-129.3455148\shika.avi
Processing video: C:/yamaneko-kenkyu/data/30052024/34.6245436-129.3455148\shika.avi
Video FPS: 30.00030000300003, Total Frames: 623, Max Frames to Process: 150

7. 動画処理ツールアプリケーション

概要

動画処理ツールアプリは、動画ファイルを閲覧および管理するための統合ツールです。再生、動画間のナビゲーション、速度制御、動画ファイルのファイル名変更をサポートしています。

動画処理ツールの起動

- **動画検出アプリ**から、サイドバーの**動画処理ツールボタン**をクリックします。

インターフェース要素

動画フレーム

- 再生中に動画が表示されるメインエリア。

コントロール

- **動画を開く**:
  - **ボタンラベル**: "動画を開く"
  - **機能**:動画ファイルを選択するファイルダイアログを開きます。
- **前の動画**:
  - **アイコン**: "⏪"
  - **機能**:現在のフォルダ内の前の動画に移動します。
- **再生/一時停止ボタン**:
  - **アイコン**: "⏯"
  - **機能**:動画の再生と一時停止を切り替えます。
- **リプレイボタン**:
  - **アイコン**: "🔁"
  - **機能**:現在の動画を最初から再生します。
- **次の動画**:
  - **アイコン**: "⏩"
  - **機能**:現在のフォルダ内の次の動画に移動します。
- **速度制御**:
  - **ラベル**: "再生速度:"

- **コンボボックス**：1.0x から 32x までの再生速度を選択できます。

動画のファイル名変更機能

- **ファイル名変更テキストボックス**：
  - **目的**：現在の動画の新しい名前を入力するための入力フィールド。
- **ファイル名変更ボタン**：
  - **ラベル**：″ファイル名変更″
  - **機能**：現在の動画ファイルを指定した名前にファイル名変更します。
- **使用方法**：
  1. テキストフィールドに新しい名前を入力します（ファイル拡張子は不要）。
  2. **ファイル名変更**ボタンをクリックします。
  3. アプリケーションはファイル名変更し、プレイリストを更新します。

8. 技術的詳細

アプリケーションの基礎となる機能に興味のあるユーザー向けに、このセクションではメディアファイルの処理方法について詳しく説明します。

画像の処理

- **関数**:process_image_file
- **ワークフロー**:
    1. OpenCV を使用して画像ファイルを読み込みます。
    2. AI 検出モデルを使用して画像を処理します。
    3. 確信度のしきい値に基づいて検出をフィルタリングします。
    4. 検出カテゴリに基づいて適切なプレフィックスを決定します。
    5. ファイル名変更が有効な場合は画像ファイル名変更します。
    6. 設定されていれば検出データを保存します。

```
        quiet=True )
    detections = results[0].get('detections', [])
    valid_detections = [d for d in detections if d['conf'] > confidence_threshold]
    if not valid_detections:
        log(f"No valid detections in {image_path}")
        if delete_no_detections:
            try:
                os.remove(image_path)
                log(f"Deleted image: {image_path}")
            except Exception as e:
                log(f"Failed to delete {image_path}: {str(e)}")
        return
    prefix = ""
    for detection in valid_detections:
        if detection['category'] == '1':
            prefix = animal_prefix
        else:
            prefix = hito_prefix
    if rename_images:
```

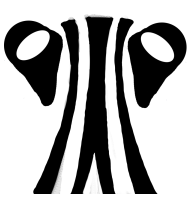

```
            image_dir = os.path.dirname(image_path)
            new_image_name = prefix + image_file_name
            new_image_path = os.path.join(image_dir, new_image_name)
            if os.path.exists(new_image_path):
                log(f"Cannot rename {image_path}: File {new_image_name} already exists")
            else:
                os.rename(image_path, new_image_path)
                log(f"Renamed image: {image_path} -> {new_image_path}")
                image_file_name = new_image_name
        if output_base is not None:
            output_folder_name = f"{os.path.splitext(image_file_name)[0]}"
            output_dir = os.path.join(output_base, output_folder_name)
            os.makedirs(output_dir, exist_ok=True)
            output_image_path = os.path.join(output_dir, 'detections.jpg')
            draw_detections_on_image(image.copy(), valid_detections, confidence_threshold,
output_image_path)
            log(f"Saved detection image to {output_image_path}")
            for i, detection in enumerate(valid_detections):
                cropped_image = crop_image_with_bbox_image(image, detection['bbox'])
                cropped_image_path = os.path.join(output_dir, f'cropped_image_{i}.jpg')
                cv2.imwrite(cropped_image_path, cropped_image)
                log(f"Saved cropped image to {cropped_image_path}")
            log("Image processing complete")
        else:
            log("Detection data saving is disabled.")
        log("Image processing complete")
```

動画の処理

- **関数**:process_video_file
- **ワークフロー**:
    1. フレーム間隔の設定で指定された間隔で動画からフレームを抽出します。
    2. 各フレームを AI 検出モデルで処理します。

3. 確信度のしきい値に基づいて検出をフィルタリングします。
4. 検出カテゴリに基づいて適切なプレフィックスを決定します。
5. ファイル名変更が有効な場合は動画ファイル名変更します。
6. 設定されていれば検出データを保存します。
7. 有効な検出がない場合、削除が有効なら動画を削除します。

```
def process_video_file(
    video_file, detector, confidence_threshold, output_base, log,
    every_n_frames=16, max_duration_seconds=10, save_all_detections=False,
    rename_videos=False, delete_no_detections=False, hito_prefix="hito_",
animal_prefix="nekokamo_"
):
    video_path = video_file
    video_file_name = os.path.basename(video_file)
    log(f"Processing video: {video_path}")
    cap = cv2.VideoCapture(video_path)
    if not cap.isOpened():
        log(f"Could not open video {video_path}")
        return
    fps = cap.get(cv2.CAP_PROP_FPS)
    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    max_frames = int(max_duration_seconds * fps)
    log(f"Video FPS: {fps}, Total Frames: {total_frames}, Max Frames to Process:
{max_frames}")
    frame_count = 0
    temp_dir = tempfile.mkdtemp()
    frame_files = []
    frame_indices = []
    best_detection = None
    best_frame = None
    max_confidence = -1
    frames_with_detections = []
    detections_found = False
    try:
        while cap.isOpened():
            ret, frame = cap.read()
```

```
            if not ret:
                break
            frame_count += 1
            if frame_count > max_frames:
                log(f"Reached max duration ({max_duration_seconds} seconds) for
{video_path}")
                break
            if frame_count % every_n_frames == 0:
                frame_filename = os.path.join(temp_dir, f'frame_{frame_count}.jpg')
                cv2.imwrite(frame_filename, frame)
                frame_files.append(frame_filename)
                frame_indices.append(frame_count)
        cap.release()
        if not frame_files:
            log(f"No frames extracted from {video_path}")
            return
        results = process_images(
            im_files=frame_files,
            detector=detector,
            confidence_threshold=0.0,
            use_image_queue=False,
            quiet=True
        )
        for i, result in enumerate(results):
            detections = result.get('detections', [])
            valid_detections = [d for d in detections if d['conf'] >
confidence_threshold]
            if valid_detections:
                detections_found = True
                frame_path = frame_files[i]
                frame = cv2.imread(frame_path)
                prefix = ""
                for detection in valid_detections:
                    if detection['category'] == '1':
                        prefix = animal_prefix
                    else:
```


```
                prefix = hito_prefix
            if save_all_detections and output_base is not None:
                frames_with_detections.append((frame_path, valid_detections))
            else:
                for detection in valid_detections:
                    if detection['conf'] > max_confidence:
                        max_confidence = detection['conf']
                        best_detection = detection
                        best_frame = frame.copy()
    if not detections_found:
        log(f"No valid detections in {video_path}")
        if delete_no_detections:
            try:
                os.remove(video_path)
                log(f"Deleted video: {video_path}")
            except Exception as e:
                log(f"Failed to delete {video_path}: {str(e)}")
        return
    if rename_videos:
        video_dir = os.path.dirname(video_path)
        new_video_name = prefix + video_file_name
        new_video_path = os.path.join(video_dir, new_video_name)
        if os.path.exists(new_video_path):
            log(f"Cannot rename {video_path}: File {new_video_name} already exists")
        else:
            os.rename(video_path, new_video_path)
            log(f"Renamed video: {video_path} -> {new_video_path}")
            video_file_name = new_video_name
            video_path = new_video_path
    if output_base is not None:
        output_folder_name = f"{os.path.splitext(video_file_name)[0]}"
        output_dir = os.path.join(output_base, output_folder_name)
        os.makedirs(output_dir, exist_ok=True)
        if save_all_detections and frames_with_detections:
            for idx, (frame_path, detections) in enumerate(frames_with_detections):
                frame = cv2.imread(frame_path)
```


```
                frame_with_detections = frame.copy()
                output_image_path = os.path.join(output_dir,
f"frame_{idx}_with_detections.jpg")
                draw_detections_on_image(frame_with_detections, detections,
confidence_threshold, output_image_path)
                log(f"Saved frame with detections to {output_image_path}")
                for j, detection in enumerate(detections):
                    cropped_image = crop_image_with_bbox_image(frame,
detection['bbox'])
                    cropped_image_path = os.path.join(output_dir,
f"frame_{idx}_cropped_{j}.jpg")
                    cv2.imwrite(cropped_image_path, cropped_image)
                    log(f"Saved cropped image to {cropped_image_path}")
            log(f"Saved all detections for video {video_file_name} to {output_dir}")
        elif best_detection is not None and best_frame is not None:
            output_image_path = os.path.join(output_dir,
'best_frame_with_detections.jpg')
            draw_detections_on_image(best_frame.copy(), [best_detection],
confidence_threshold, output_image_path)
            cropped_image = crop_image_with_bbox_image(best_frame,
best_detection['bbox'])
            cropped_image_path = os.path.join(output_dir, 'cropped_image.jpg')
            cv2.imwrite(cropped_image_path, cropped_image)
            log(f"Saved best frame with detection to {output_image_path}")
            log(f"Saved cropped image to {cropped_image_path}")
        else:
            log("No frames to save.")
    else:
        log("Detection data saving is disabled.")
    log("Video processing complete")
finally:
    shutil.rmtree(temp_dir)
```


検出とファイル名変更のロジック

- **検出カテゴリ**:
  - カテゴリ '1': 動物。
  - その他のカテゴリ: 人間または車両。
- **プレフィックス**:
  - **人・車のタグ**: 人間や車両が検出された場合に適用されます。
  - **動物のタグ**: 動物が検出された場合に適用されます。
- **ファイルのファイル名変更**:
  - 適切なプレフィックスを元のファイル名に追加します。
  - 重複するファイル名が作成されないようにします。

9. エラー対処とトラブルシューティング

よくある問題

アプリケーションがクラッシュするか起動しない

- **解決策**：すべての依存関係が正しくインストールされ、VLC ライブラリが正しくセットアップされていることを確認してください。

検出が行われない

- **解決策**：確信度のしきい値を調整するか（0.15-0.6 が好ましい）、メディアファイルに検出可能なオブジェクトが含まれていることを確認してください。

ファイル名変更できない

- **解決策**：ファイルの権限を確認し、同じ名前の他のファイルがディレクトリに存在しないことを確認してください。

動画処理ツールが起動しない

- **解決策**：video_player.py スクリプトへのパスを確認し、アクセス可能であることを確認してください。

ログ

- アプリケーションにはトラブルシューティングを支援するためのログが提供されています。
- **ログテキストエリア**でエラーメッセージや処理情報の詳細を確認してください。

10. 付録

追加の注意事項

- **カスタマイズ**：アプリケーションは柔軟に設計されており、ユーザーが設定やタグを調整して特定のニーズに合わせることができます。
- **パフォーマンスの考慮事項**：大量のファイルや高解像度の動画を処理する場合、かなりの計算リソースが必要になることがあります。GPU 上での実行を推奨しています。

連絡先情報

サポートや問題の報告については、以下にご連絡ください：

- **サポートメール**：DiegoAlonsoJapan@gmail.com