GIMP-Turbo開発計画

2025年3月24日

17:37

プロジェクト概要

# 目的:

本プロジェクトの目的は、画像品質の向上に特化したソフトウェアの開発である。

# 経緯:

画像生成AIを利用した画像・イラスト作成を行っていく上で、元画像の品質向上が不足すると画像が崩れたり、想定通りの画像にならないことが多い。

ただし、画像の品質を向上させるためのソフトウエアは操作コストが高く、難易度の低いWeb上のサービスは費用がかかるなど不便なことが多いため、画像品質向上に特化した軽量・高速・ユーザビリティなソフトウェア開発の決断に至った。

プロジェクト方針

# 基本方針

* + 安定した環境で結果に差異が発生しづらい開発環境を優先
  + 開発中に問題が発生した場合、以下の順にトップダウンの対応を行う
    1. 公式マニュアルやソース内のドキュメントを確認し、手順に誤りがないか確認
    2. 作業が手順通りに進められているか、ログ等で確認（原因調査ではなくあくまで手順の確認）
    3. 事象を回避可能な代替案が存在しないか確認
    4. 問題なく動作している箇所と、事象が発生している箇所を特定し、調査範囲を狭めていく

# ライブラリ選定方針

* + 実装が簡単: 初期段階での開発をスムーズに進めるため、シンプルな実装を優先。
  + 処理が軽量: ユーザーのPCリソースを効率的に使用し、快適な操作性を提供。
  + 操作が簡単: ユーザーインターフェースを直感的にし、誰でも簡単に使えるようにする。
  + 高品質な画像を実現: 最終的な目標として、ユーザーが満足する高品質な画像を提供。

## 

# 技術方針

* + ソフトウエア利用環境: Windowsネイティブ環境（stable diffusionの利用者を想定）
  + 開発環境: Windows11に構成したWSL内のUbuntu（再現性の高い開発環境の構築が目的）
  + リンク方式: スタティックリンク（オールインワンで利用しやすく、機能間のオーバーヘッド極小化が目的）
  + 画像処理開発言語: C++（処理速度を重視するため）
    1. 画像処理用ライブラリ: OpenCV（豊富なドキュメントと機能性）
  + GUI開発言語: Python（再現性を重視）
    1. GUIフレームワーク: PyQt（再現性、機能性を重視）

基本設計

# 基本機能

* + コンソールアプリ
    1. 画像解析
       - ノイズの除去レベル測定
       - どの程度の画像品質向上が望ましいかの解析
    2. 画像品質向上
       - ノイズ除去（解析結果に基づく自動選択）
       - アップスケーリング（バイキュービック補間）
       - 画像品質向上
       - アップスケーリング前のサイズに戻す
  + GUIアプリ
    1. 画像表示
       - ユーザー操作により取り込んだ画像を表示
       - コンソールアプリが処理した画像のプレビュー表示
       - ユーザ操作によるパラメータ変更に応じたプレビュー画像のリアルタイム更新
    2. パラメータ取り込み
       - コンソールアプリの解析結果パラメータを取り込み

# 品質向上スキーム

* + ノイズ解析  
    画像内のノイズの種類とレベルを解析します。
  + ノイズ除去  
    解析結果に基づいて適切なフィルタを選択し、ノイズを除去します。
  + アップスケーリング  
    画像の解像度を上げるためにバイキュービック補間を適用します。
  + 画像解析  
    アップスケーリング後の画像を解析し、品質向上のためのパラメータを取得します。
  + 画像品質向上  
    取得したパラメータに基づいて、画像の品質を向上させます。
  + 解像度復元  
    アップスケーリング前の解像度に復元（縮小？）

## 

詳細設計

# コンソールアプリ

* + ビルドシステム
    1. CMake（OpenCVのビルドに広く使用されており、豊富なドキュメントとサポート）
  + ノイズ除去
    1. OpenCVを利用したノイズ解析の結果、適切なフィルタでノイズを除去
       - ガウシアンフィルタ（一般的なノイズ除去）
       - メディアンフィルタ（塩胡椒ノイズに対して効果的）
  + アップスケーリング
    1. バイキュービック補間（実装が簡単で、処理速度も速く、品質も良好）
  + 画像品質向上
    1. コントラスト調整
       - ヒストグラム平坦化
    2. シャープネス強調
       - アンシャープマスク
    3. 色補正
       - ホワイトバランス調整
    4. ガンマ補正
       - 固定のガンマ値を使用

# GUIアプリ

* + 取り込み画像エリア

位置: 左側

サイズ: 300×300ピクセル

機能: ユーザーが選択した画像を表示

* + プレビュー画像エリア

位置: 右側

サイズ: 300×300ピクセル

機能: 処理後の画像を表示

* + 進捗インジケータ

位置: ノイズ調整用スライダーの上

機能: プレビュー画像表示までの進捗を示す

* + ノイズ調整用スライダー

位置: 上部

機能: ノイズ除去のレベルを調整

* + アップスケーリング用スライダー

位置: 中央

機能: 画像のアップスケーリングレベルを調整

* + 画質向上用スライダー

位置: 下部

機能: 画像の全体的な画質向上レベルを調整

* + 画像取り込みボタン

位置: 画面下部

機能: ユーザーが画像を取り込むためのボタン

* + 画像保存ボタン

位置: 画面下部

機能: ユーザーが処理した画像を保存するためのボタン

* + 閉じるボタン

位置: 画面下部

機能: アプリケーションを閉じるためのボタン

別紙「GUIイメージ図」も参照

環境構築

# 開発環境

* + OS: ubuntu（WSL環境）

## 

# ディレクトリ構成

* + Windows側:

フォルダ構成: C:\Users\RJ059040\Documents\projects\GIMP-Turbo

共有フォルダ: C:\Users\RJ059040\Documents\projects\GIMP-Turbo\Shared

* + WSLのUbuntu側:

ディレクトリ構成: /home/pakchi/projects/GIMP-Turbo

共有ディレクトリ: /mnt/c/Users/RJ059040/Documents/projects/GIMP-Turbo/Shared

シンボリックリンク: /home/pakchi/projects/GIMP-Turbo/Shared -> /mnt/c/Users/RJ059040/Documents/projects/GIMP-Turbo/Shared

# 導入コンポーネント

* + Windows:
    1. Visual Studio Code
    2. Git for Windows
  + WSLのUbuntu:
    1. Git
    2. OpenCV
    3. Cmake
    4. Python
    5. PyQt
    6. 依存関係ライブラリ
       - libgtk2.0-dev
       - pkg-config
       - libavcodec-dev
       - libavformat-dev
       - libswscale-dev

# プロジェクトマイルストーン

Step0: 環境構築

* + 目的: Step1からStep6までに利用する環境やコンポーネントを洗い出し、すべて導入。
  + 実装内容:
    1. 開発環境の設定:

Step1: 画像データ入出力アプリの開発

* + 目的: 画像ファイルの読み込みと保存を行う基本的な機能を実装。
  + 実装内容:
    1. コンソールアプリとして開発
    2. 入出力のファイルパスをハードコーディングして、画像を読み込み・保存する。

Step2: Windowsの「送る」コマンド対応アプリの開発

* + 目的: ユーザーが選択した画像ファイルを簡単に処理できるようにする。
  + 実装内容:
    1. コンソールアプリとして開発
    2. Windowsの「送る」コマンドで取得したパスに従って画像を取り込み、別名で保存。
    3. 必要に応じてPowershellをパイプさせる

Step3: 基本的な画像品質を向上アプリの開発

* + 目的: 基本的な画像品質向上機能を実装。
  + 実装内容:
    1. コンソールアプリとして開発
    2. 画像データの受け取りは「送る」コマンドを利用
    3. 既定のパラメータ通りの画像品質向上を実施。

Step4: 画像解析と画像品質向上を連動させるアプリの開発

* + 目的: 画像解析結果に基づいて、画像品質を最適化。
  + 実装内容:
    1. コンソールアプリとして開発
    2. 最適なパラメータを取得するための画像解析機能
    3. 取得したパラメータ通りに画像品質を向上

Step5: コンソールアプリの入出力となるGUIアプリの開発

* + 目的: ユーザーが直感的にパラメータを調整できるGUIを提供。
  + 実装内容:
    1. モックとして開発
    2. GUI側の入出力機能のみで実現可能な機能は実装
    3. この時点でコンソールアプリとの統合はしない

Step6: コンソールアプリとGUIアプリの統合

* + 目的: スタティックリンクの画像処理ソフトウェア開発
  + 実装内容:
    1. コンソールとGUIの統合
    2. 「画像の選択」ボタンをクリックして表示されるダイアログで取り込む画像を選択
    3. 「画像の保存」ボタンをクリックすることで、プレビュー表示されている画像を保存
    4. 「閉じる」ボタンをクリックすることで、プレビューやソフトウェアのメモリ・キャッシュ・プロセスを削除し閉じる
    5. 画像解析結果のパラメータをGUIに初期値として反映する
    6. ユーザーによるGUI操作でコンソール側にパラメータを反映する
    7. 設定されたパラメータによる品質向上後の画像をプレビュー画像として表示する
    8. すべてのコンポーネントを統合し、単体動作可能なソフトウェアとする

Step7: 完成したアプリにブラッシュアップの余地がある場合、改修を検討する

* + 目的: 完成したアプリの品質をさらに向上させる。
  + 実装内容:
    1. ユーザーからのフィードバックを収集。
    2. 必要に応じて改修を行い、アプリの品質を向上させる。