Advanced Algorithm for Enhancement of Fashion Imagery

Jean-Luc Peloquin
Computer Science Department
University of Nevada, Las Vegas
peloqj1@unlv.nevada.edu

April 28, 2024
(updated March 6, 2025)

Abstract

This report presents an advanced algorithm designed for the enhancement of fashion imagery, focusing on background removal, and dimension transformation to improve detectability through reverse image search. By integrating state-of-the-art image processing techniques, the algorithm enhances image clarity, enables precise background extraction, and supports automatic cropping and .png exportation with transparent backgrounds. The enhancements address challenges such as artifacting, color distortion, and blurring that degrade reverse image search results.

1 Introduction

Often, reverse image searching an image is either extremely accurate or the search engines interpret the input too-broadly, leading to insufficient results. The image quality given to the reverse image search engine is paramount to receiving a desired result. This project aims to develop a comprehensive methodology that focuses on improving common shortcomings of unprocessed fashion images through a multi-stage enhancement pipeline that includes adaptive analysis, upscaling, color correction, detail enhancement, noise reduction, and background removal.

1.1 Related Work

Existing solutions like Adobe Photoshop offer tools for image enhancement but require significant user expertise and are not freely accessible. Current methods generally lack the capability to process images in batches and do not cater specifically to the nuanced requirements of fashion imagery enhancement. This project leverages new methodologies to fill these gaps, utilizing a combination of advanced techniques:
- Lanczos algorithm for high-quality 4x upscaling
- LAB color space processing for better color handling
- High-pass filtering and unsharp masking for detail enhancement
- CLAHE (Contrast Limited Adaptive Histogram Equalization) for adaptive contrast enhancement
- Non-local means denoising for noise reduction
- U2Net deep learning model (via rembg) for background removal
- Adaptive processing based on image content analysis
- Parallel processing for efficient batch operations

1.2 Requirements

The project is built using the Python coding language. To run the software, you must have Python 3.12 installed and the following additional libraries:

- opencv-python: For advanced image processing operations
- numpy: For numerical operations and array handling
- Pillow: For basic image manipulation and format conversion
- rembg: For background removal using the U2Net deep learning model
- plyer: For desktop notifications in the UI
- (optional) pyseam: For content-aware resizing functionality

All documentation for these libraries can be seen in the references section of this document. The program accepts .jpg, .jpeg, and .png files as valid inputs to process. Included with the project is an executable version of the program that has not been tested on any OS other than win32. A sample of 100 images and 11 images are in the "input" and "in" folders respectively within the .zip folder, which is a sample of images used for training. The time to process each image is similar, so the in folder only has 11 for a quick demo of the functionality. The folders "output" and "out" hold the post-processed images for the "input" and "in" folders respectively.

Alternatively, there is a standalone version of the project in dist/ui/ui.exe that is a mirror of the program. This takes a moment to open, see README for details.

2 The Proposed Algorithm

Our approach implements a comprehensive image enhancement pipeline with adaptive processing capabilities. The system analyzes each image to determine optimal processing parameters based on lighting conditions, contrast levels, noise presence, and detail complexity. This adaptive approach ensures that each image receives appropriate treatment rather than applying one-size-fits-all enhancements.

The enhancement pipeline consists of the following stages:

1. Image Analysis: The system analyzes each image to determine optimal parameters using `analyze_image_content()` and `get_adaptive_config()` functions.

2. Pre-processing:
   - White balance correction using gray world assumption
   - Noise reduction using non-local means denoising

3. Resolution Enhancement:
   - Lanczos upscaling (4x by default)
   - Preserves details better than bicubic or bilinear methods

4. Color and Contrast Enhancement:
   - CLAHE for adaptive contrast enhancement
   - Shadow and highlight recovery
   - Color enhancement with separate saturation and vibrance controls

5. Detail Enhancement:
   - High-pass filtering in LAB color space
   - Texture enhancement
   - Unsharp masking with configurable parameters

6. Background Processing:
   - Background removal using rembg (U2Net model)
   - Optional background replacement with configurable colors
   - Edge feathering for natural transitions

7. Post-processing:
   - Auto-cropping based on content
   - Quality assessment

The implementation supports both individual and batch processing with parallel execution capabilities, making it efficient for processing large collections of fashion images. A user-friendly graphical interface built with Tkinter provides easy access to the enhancement functionality, allowing users to select input and output folders and initiate the processing with visual feedback.

3 Experiments

The algorithm was tested on a diverse dataset scraped from Grailed.com, which includes fashion items varying significantly in size and resolution. We compared our method's performance in enhancing image quality and facilitating accurate reverse image searches against traditional methods like those provided by Adobe Photoshop. Our results demonstrate marked improvements in image detail and integrity with minimal adverse

effects at a higher efficiency and mass automation that is not possible through manual enhancements through photoshop.

4 Conclusion

The developed algorithm offers a comprehensive solution for enhancing fashion imagery, significantly improving the quality and searchability of product photos. Pictured below is an image pre-processing and post-processing (through the created algorithm):

The post processed image is a much higher resolution and has the background removed. This is an example of a successful transformation. This algorithm does have limitations, though, mainly due to the background being very similar to the foreground.

Here is an example of an unsuccessful processing due to the shadow of the object obscuring defined edges in contrast to the background:

Another limitation of the algorithm is if the original image quality is too low. If the resolution is too low, even upscaling does not provide enough context for a successful transformation.

When tested in a reverse-searching scenario, the upscaled images performed as well or better than the original on every occasion. On images where the processing is excellent (such as the bomber jacket shown), it improves the reverse image search to include more results. On a more unsuccessful processing (such as the pair of sneakers shown), the search result is the same as the preprocessed image. I hypothesize that the methodology I created only helps the auto-detection algorithm of the reverse image search engine even if the processed result is not a drastic improvement on the original image, which at worst is essentially a specific preprocessing step on the input images.

The algorithm provides processed images at a greater efficiency (in both speed and quantity) compared to Photoshop. While a photoshop operation takes on average 45 seconds, an operation using my methodology takes on average 6 seconds for a high-resolution image, which can also be done in mass by using batch processing.

Future work requires further exploration of the integration of AI-driven techniques to enhance the algorithm's capability and efficiency, particularly in real-time processing environments. Potential improvements include:

- AI-based object recognition for smarter cropping
- Style transfer options for consistent product presentation
- Batch configuration profiles for different product categories
- Cloud processing integration for higher throughput
- Mobile application version

A greater dataset is required for additional testing, and greater computer power for training an advanced model for background removal, upscaling, and clarity enhancements. In addition, creating a process to recognize if an image needs upscaling would be helpful in reducing runtime by only using Lanczos on low-resolution images instead of universally, the most computationally intensive process in my algorithm.

Acknowledgment

I want to thank my professor, Dr. Bryar Shareef, whose invaluable input and guidance was instrumental in refining and bolstering my ideas for this project. His expertise and thoughtful insights significantly helped me define the direction and success of my work.

References

1] Adobe, "Smooth enhance fabric with generative fill," Adobe Photoshop Tutorials, 2024. [Online]. Available: https://creativecloud.adobe.com/learn/photoshop/web/smooth-enhance-fabric-generative-fill. [Accessed: Apr. 28, 2024].

[2] "Lanczos resampling," Wikipedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Lanczos_resampling. [Accessed: Apr. 28, 2024].

[3] "High-pass filter," Wikipedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/High-pass_filter. [Accessed: Apr. 28, 2024].

[4] Cambridge in Colour, "Unsharp mask," 2024. [Online]. Available: https://www.cambridgeincolour.com/tutorials/unsharp-mask.html. [Accessed: Apr. 28, 2024].

[5] "rembg," PyPI, 2024. [Online]. Available: https://pypi.org/project/rembg/. [Accessed: Apr. 28, 2024].

[6] "Image background removal," YouTube, uploaded by D. Gatis, 2024. [Online]. Available: https://www.youtube.com/watch?v=2X9rxzZbYqg. [Accessed: Apr. 28, 2024].

[7] Grailed, "Grailed," 2024. [Online]. Available: https://www.grailed.com/. [Accessed: Apr. 28, 2024].

[8] Data to Fish, "How to convert JPEG to PNG in Python," 2024. [Online]. Available: https://datatofish.com/jpeg-to-png-python/. [Accessed: Apr. 28, 2024].

[9] "Plyer documentation," Read the Docs, 2024. [Online]. Available: https://plyer.readthedocs.io/en/latest/. [Accessed: Apr. 28, 2024].

[10] "Pillow documentation," Read the Docs, 2024. [Online]. Available: https://pillow.readthedocs.io/en/stable. [Accessed: Apr. 28, 2024].

[11] D. Gatis, "rembg," GitHub, 2024. [Online]. Available: https://github.com/danielgatis/rembg. [Accessed: Apr. 28, 2024].

[12] "NumPy documentation," NumPy, 2024. [Online]. Available: https://numpy.org/doc/. [Accessed: Apr. 28, 2024].

[13] "OpenCV documentation," OpenCV, 2024. [Online]. Available: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html. [Accessed: Apr. 28, 2024].

[14] "CLAHE (Contrast Limited Adaptive Histogram Equalization)," OpenCV, 2024. [Online]. Available: https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html. [Accessed: Apr. 28, 2024].

[15] "Non-local Means Denoising," OpenCV, 2024. [Online]. Available: https://docs.opencv.org/4.x/d1/d79/group__photo__denoise.html#ga4c6b0031f56ea3f98f768881279ffe93. [Accessed: Apr. 28, 2024].

[16] "LAB Color Space," OpenCV, 2024. [Online]. Available: https://docs.opencv.org/4.x/de/d25/imgproc_color_conversions.html. [Accessed: Apr. 28, 2024].

[17] "Python multiprocessing," Python Documentation, 2024. [Online]. Available: https://docs.python.org/3/library/multiprocessing.html. [Accessed: Apr. 28, 2024].

[18] "Tkinter — Python interface to Tcl/Tk," Python Documentation, 2024. [Online]. Available: https://docs.python.org/3/library/tkinter.html. [Accessed: Apr. 28, 2024].

[19] "U2-Net: Going Deeper with Nested U-Structure for Salient Object Detection," Pattern Recognition, 2020. [Online]. Available: https://arxiv.org/abs/2005.09007. [Accessed: Apr. 28, 2024].