PROJECT ABSTRACT

Duplicate Image Detection using Perceptual Hashing

A perceptual-hashing tool that finds near-duplicate images in large folders by clustering 64-bit pHash fingerprints within a Hamming-distance threshold.

01. Title

Duplicate Image Detection using Perceptual Hashing

02. Introduction / Background

Photo collections accumulate near-duplicates: the same picture saved at multiple resolutions, recompressed by messaging apps, or cropped before re-upload. Cryptographic hashes such as MD5 only detect byte-identical files, because changing a single pixel produces a completely different digest. Perceptual hashing addresses the real need by mapping visually similar images to similar hash values, so that a small distance between two fingerprints indicates the images look alike. This project builds a duplicate-detection tool around pHash, the DCT-based perceptual hash, with an evaluation notebook that measures detection quality on a labeled duplicate set.

03. Problem Statement

Large image folders contain near-duplicate files that waste storage and clutter libraries, yet byte-level checksums cannot find them: recompression, resizing, or minor edits defeat cryptographic hashing entirely. Users need a content-based method that groups visually similar images together, presents the groups for human review, and reports measurable precision and recall so the tool's behavior is transparent rather than a black box.

04. Objectives

The objectives of this project are to implement DCT-based perceptual hashing (pHash) for image fingerprinting; to build a folder scanner that hashes large collections into a reusable fingerprint index; to cluster near-duplicates by Hamming distance with a configurable threshold; to provide safe review-and-quarantine actions instead of silent deletion; and to evaluate the detector with precision, recall, and F1 on a labeled duplicate set during the build.

- Implement 64-bit pHash fingerprinting (dHash and aHash included as comparison modes).

- Build a recursive folder scanner with a fingerprint cache for incremental re-scans.
- Cluster near-duplicates by Hamming distance with a configurable threshold.
- Generate side-by-side preview reports and safe quarantine/CSV-manifest actions.
- Evaluate with precision, recall, and F1 on a labeled duplicate set via the included notebook.

05. Proposed System / Working

The scanner walks the folder tree and loads each image, converting it to a canonical size and grayscale. Each image is transformed with a discrete cosine transform; the low-frequency coefficients are binarized against their median to form a 64-bit pHash fingerprint, following the construction described by Zauner (2010). Fingerprints are stored in an indexed cache so only new or changed files are re-hashed on repeat runs. Pairwise Hamming distances link images within the configured threshold into duplicate groups, and a report generator renders each group with thumbnails side by side together with the member distances. The user reviews the groups and chooses per-group actions (quarantine, CSV manifest, or ignore); originals are never touched automatically. The evaluation notebook runs the detector over a labeled duplicate set, computes precision, recall, and F1, and plots the threshold tradeoff curve.

- Folder scan and canonical normalization of each image.
- DCT-based 64-bit pHash fingerprint computation.
- Indexed fingerprint cache for incremental re-scanning.
- Hamming-distance clustering into duplicate groups.
- Side-by-side preview report with per-group review actions.
- Notebook evaluation: precision, recall, F1, and threshold tradeoff curve.

06. Technologies / Components Used

The implementation uses Python 3.10 with Pillow for image loading and normalization, the ImageHash library for pHash/dHash/aHash fingerprinting, and NumPy/SciPy for the DCT and Hamming-distance computation. pandas manages the fingerprint index and duplicate manifests, while a Jupyter notebook performs the evaluation and Matplotlib/Seaborn plot the threshold tradeoff curves. An argparse CLI drives the tool and an HTML report presents duplicate groups for review.

- Python 3.10, Pillow (image loading, normalization).
- ImageHash (pHash, dHash, aHash perceptual fingerprints).
- NumPy, SciPy (DCT, Hamming-distance computation).
- pandas (fingerprint index, duplicate manifests).
- Jupyter notebook (evaluation: precision, recall, F1).
- Matplotlib, Seaborn (threshold tradeoff curves).
- argparse CLI plus HTML report for group review.

07. Key Features

Key features include 64-bit perceptual fingerprints with three hash modes for comparison, a recursive scanner with a reusable fingerprint cache, a configurable Hamming-distance threshold exposing the precision/recall tradeoff, duplicate-group clustering with representative-image selection, side-by-side preview reports for human review, and safe actions (quarantine folder or CSV manifest) with no silent deletion.

- 64-bit pHash fingerprints; dHash and aHash comparison modes.
- Recursive folder scanner with reusable fingerprint cache.
- Configurable Hamming-distance threshold.
- Duplicate-group clustering with representative selection.
- Side-by-side preview report for human review.
- Safe quarantine and CSV-manifest actions; never silent deletion.
- Evaluation notebook computing precision, recall, and F1.

08. Applications / Use Cases

The tool applies to personal photo-library cleanup, de-duplicating image datasets before machine-learning training, media-asset management in small studios, and forensic-style triage of large image dumps where byte checksums are insufficient. It also serves as a teaching vehicle for perceptual hashing theory and information-retrieval evaluation.

- Personal photo-library cleanup and storage reclamation.
- De-duplication of image datasets before ML training.
- Media-asset management for small studios.
- Triage of large image dumps where checksums fail.
- Teaching perceptual hashing and retrieval evaluation.

09. Results & Scope

The expected outcome is a working duplicate detector whose quality is measured, not asserted: the included notebook computes precision, recall, and F1 on a labeled duplicate set during the buyer's build and plots how the threshold trades precision against recall. Scope covers common formats (JPEG, PNG, WebP), thousands of images on a laptop via the fingerprint cache, and a clear statement of failure modes (heavy crops or rotations can evade detection; visually similar-but-different photos can cluster together). Future scope includes rotation-robust hashing variants and a graphical review interface.

- Measured outcome: precision, recall, and F1 computed by the notebook during the buyer's build.
- Threshold tradeoff curve documenting the precision/recall dial.

- Scope: JPEG/PNG/WebP folders, thousands of images on a laptop.
- Documented failure modes: heavy edits evade detection; lookalikes can cluster.
- Future scope: rotation-robust hashes, graphical review UI.

10. Conclusion

Perceptual hashing turns duplicate detection from a byte-comparison problem into a visual-similarity problem, and this project delivers that as a complete, evaluable tool: scanner, fingerprint index, clustering, human-review reporting, and a notebook that measures precision and recall on labeled data. Built to order, it ships with source code, the evaluation notebook, report, presentation, and viva preparation material.

11. References

The project cites the pHash construction, the ImageHash implementation, and standard information-retrieval evaluation.

- C. Zauner, 'Implementation and Benchmarking of Perceptual Image Hash Functions,' M.S. thesis, Upper Austria University of Applied Sciences, 2010.
- J. Buchner, 'ImageHash' perceptual hashing library, https://github.com/JohannesBuchner/imagehash.
- C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008 (precision, recall, F1).