NOVEL APPROACHES TO CONTENT MODERATION OF END-TO-END ENCRYPTED IMAGES USING PERCEPTUAL HASHES

Mantha Akshara[1], Peng Ruijia[1], Tan Si Ying[1], Ruth Ng Ii-Yung[2], Chan Ming Kai Alvin[2]
[1] Raffles Institution, 1 Raffles Institution Lane, Singapore 575954
[2] DSO National Laboratories, 12 Science Park Drive, Singapore 118225

Abstract

In this paper, we address the issue of content moderation of harmful images in applications with end-to-end encryption (E2EE). In E2EE, the image is encrypted before transmission on the server such that the server cannot read the original image, preserving user privacy. Perceptual image hashing is a type of hash which returns analogous results when similar images are hashed. We explore three approaches to privacy-preserving content moderation in E2EE image communication using perceptual hashes – using hashes individually, taking the majority result of all the hashes, and using a decision tree. The decision tree approach demonstrates promising results with a 95% accuracy rate, the highest of all the content moderation methods we explored, showing the robustness of our approach. We tested the decision tree over a variety of hashed images of people to demonstrate the feasibility of our approach to be used in various real-world applications, and created an accompanying library.

1. Introduction

With the advent of the Internet, the problem of content moderation has become increasingly important. Content moderation is necessary to prevent the spread of harmful, inappropriate, or illegal material, ranging from sensitive company records to copyrighted images to offensive or inappropriate content such as R21 material, that is, mature material strictly for viewing by adults 21 and above. Applications of content moderation in real-world scenarios include companies preventing employees from sharing sensitive company information via email, and ensuring that copyrighted images are not distributed unlawfully. However, the massive amount of media online makes manual verification of individual pieces of media by human monitors infeasible, necessitating automation in content moderation.

We explore content moderation in the setting of end-to-end encrypted (E2EE) communication of images. This involves users sending content over a server. In E2EE communication, only users involved in the communication have access to the content sent, and the content should be kept private from the server, as illustrated in Figure 1. An example of E2EE image communication is sending images over WhatsApp. This complicates the content moderation process as the server cannot directly access images shared by users and scan them for problematic content. Relying solely on user reports to flag problematic images is also insufficient as it leaves room for abuse between two consensual parties, and does not prevent harm to the user upon initial exposure to the image, especially for disturbing images such as Child Sexual Abuse Material (CSAM). Given these challenges, it is imperative to find a privacy-preserving method to automate content moderation. This may involve a client-side check on the original image or a server-side check on

a privacy-preserving representation of the image. Should the image be reported to be visually similar to undesirable content, further content moderation actions such as sending the original image to human moderators can be carried out on the grounds that the image has a high probability of going against the Terms of Service of the communication platform.

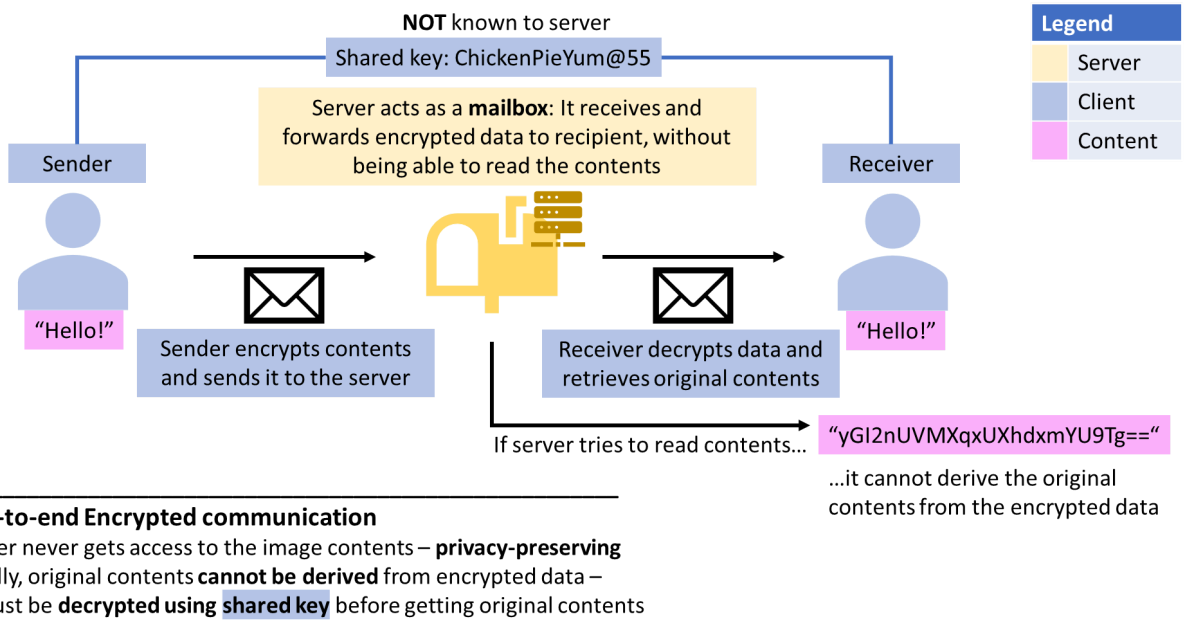


Figure 1: Overview of E2EE communication process

The approach we use in this paper is perceptual image hashing, a lightweight and fast method commonly used in detecting similar images in image forensics. Perceptual hash algorithms take an image as input and output a fixed-length hash as output. In E2EE image communication, the perceptual hash algorithms are run client-side on the image to produce perceptual hashes, which are sent to the server along with the encrypted image contents. The server then decides whether to forward the encrypted image to the receiver based on the hashes received. The image cannot be reconstructed from its hash, preserving the privacy of the image, even when the server is given access to its perceptual hash to be compared against a database of perceptual hashes of known bad images for content moderation purposes. Unlike cryptographic hashes which produce a vastly different hash should one bit of the input be modified, perceptual hashes are able to detect visually similar images by considering the global features of the image in the construction of the hash. For example, a desaturated variant of an R21 image would be considered similar to its original, allowing for detection and content moderation of such modified R21 images. This often involves preprocessing the image via downscaling and changing its colour space to reduce detail. Hence, a key metric in measuring the performance of a perceptual hash algorithm would be its robustness in its ability to classify two visually similar images as similar, and vice versa.

1.1 Contributions

In this paper, we:

- analyse the pros and cons of each perceptual hash algorithm

- explore perceptual hash algorithm application approaches, using combinations of four different perceptual hash algorithms
- run experiments quantifying the effectiveness of our approaches
- produce an accompanying library utilising these approaches

We have formulated three different approaches that utilise the properties of these perceptual hashes in a real-life content moderation system of E2EE image communications.

1. Individual Hash – The verdict for whether an image is visually similar to an R21 image in the dataset was determined solely from a single hash algorithm.
2. Majority Decision – The results from all four hash algorithms were considered in determining whether the image is visually similar to an R21 image in the dataset, and the final verdict was the verdict of the majority of the hash algorithms. In the case of a tie, the verdict of the best-performing hash from Approach 1 was taken.
3. Decision Tree – The results for a combination of all four hash algorithms were passed through a decision tree, which would produce a final verdict.

With approach 1 and 2 as benchmarks, our decision tree approach has produced promising results for improving the robustness of comparing visually similar images, as elaborated upon under Section 4.3.

2. Background

Our work builds upon Sharma [1] which analyses the performance of perceptual hashes, which is why we chose to focus on difference hash (dHash), perceptual hash (pHash), wavelet hash (wHash) and non-negative matrix factorisation hash (NMFHash). However, a hash using Singular Value Decomposition (SVD) was discarded due to poor performance in our initial tests and [3].

Difference hash (dHash) uses the difference between the intensity values of adjacent pixels to compute the final hash. Perceptual hash (pHash) generates a hash by applying a discrete cosine transform (DCT) to an image to separate it into frequency and scalar domains, reducing the image detail by extracting the low-frequency components, and then converting the processed image to a hash. Wavelet hash (wHash) works like pHash but uses discrete wavelet transform (DWT) instead. Lastly, NMFHash generates the hash by first dividing the image into circular rings, which are then straightened and used to create a secondary image. The hash is then found by performing non-negative matrix factorisation (NMF) on the secondary image. A more comprehensive overview of the four hashes is available in Appendix 1.

Previous literature reviewing the performance of perceptual hashes [1][2][3] has determined that, individually, their performance in real-world situations may not be optimal. Furthermore, different content-preserving modifications such as colour correction and rotation may be applied to an image to generate a visually similar variant, both by malicious users attempting to bypass automated content moderation checks and non-malicious users for aesthetic or practical purposes. Different hash algorithms are robust against different content-preserving modifications, and there is no hash that performs the best in detecting all modifications. Hence, a

combination of perceptual hashes may result in greater robustness in detecting visually similar images for content moderation. In this paper, we explore and make recommendations on the approaches utilising perceptual hashes as mentioned in Section 1.1, whether individually or combined, to improve the robustness of these hashes in detecting visually similar images.

3. Methodology

We empirically tested the robustness of our three novel approaches on two self-curated image datasets, adapting methodology from previous literature.

3.1 Datasets

We tested our approaches on two datasets, Magazines and People, to quantify their effectiveness. In our context, we define undesirable content to be the transmission of R21 images in E2EE image communications. Hence, each dataset consists of "R21" images, mature material strictly for viewing by adults 21 and above, which are the images to be compared against; "PG" images, images that are suitable for viewing by all ages, which should not be considered visually similar to the R21 images; "modified" images which should. The modified images were generated by applying different content-preserving modifications to the R21 images. Such modifications include colour changes such as saturation and brightness, and geometrical changes such as crop and rotation. Refer to Appendix 2 for the breakdown of the specific modifications.

The Magazines dataset consists of 300 R21 images (adult magazine covers), 5605 modified images and 5250 PG images (fashion magazine covers). The People dataset consists of 249 R21 images (more general than magazine covers), 4731 modified images, and 5269 PG images. Refer to Appendix 3 for more details.

3.2 Hash algorithms

We used four perceptual hashing methods for our experiment. Three of them (pHash, dHash and wHash) used the implementations in the ImageHash Python library [4], while we implemented NMFHash [5] in Python. Additional details can be found under Section 2.

Experiments quantifying the robustness of individual hashes against different content-preserving modifications have been conducted in previous literature [1][2][3]. We have employed similar metrics and modifications in our tests, which returned corroborating results.

3.2.1 Image Pre-processing

Perceptual hashes often conduct pre-processing to reduce image detail so that similar images are more likely to be considered visually similar. This often involves normalisation to a smaller image size, and a simplified colour space. For dHash, pHash and wHash, the image was normalised to 17 x 16 (dHash) and 64 x 64 (pHash, wHash) and changed to grayscale. For NMFHash, the image was normalised to 512 x 512 and changed to YCbCr colour space.

3.2.2 Hash Similarity

We utilise standardised metrics for comparing the similarity of two hashes to determine whether images are visually similar. A threshold value is used to determine the bounds of similarity after which an image is classified as different. Images with low similarity are considered as visually dissimilar images, while images with high similarity are considered as visually similar images.

For pHash, dHash and wHash, Hamming distance, which measures the number of differing bits between two hashes, was used to compute the similarity between two images. A Hamming distance of 0 indicates that the images are identical, while a distance of 1 suggests the images are completely different. Hence, a smaller Hamming distance implies a higher similarity.

For NMFhash, the Pearson correlation coefficient was used to compute the similarity between two images. The range of the Pearson correlation coefficient is between -1 and 1, where a value of 1 indicates that the images are identical, while a value of -1 implies the images are completely different. Hence, a greater Pearson correlation coefficient value implies a higher similarity.

3.3 Data Processing

Our dataset consists of three types of images as mentioned in 3.1: R21 images, modified images, and PG images. Refer to section 3.1 for the definitions of these images.

First, we hashed all images in our dataset using the four perceptual hash algorithms described in Section 2, producing four hashes per image. Next, we calculated similarity scores as follows:

1. For each modified and PG image, its hash was compared to the corresponding hash of all R21 images, where both hashes were produced using the same hashing algorithm.
2. The highest similarity value (lowest Hamming distance for dHash, pHash, and wHash, and the highest Pearson correlation coefficient for NMFHash) across all R21 image hash comparisons was recorded as the similarity score for that image. The hashes of the modified images should have a high similarity score when compared to the hashes of the R21 images as they are visually similar, while the hashes of the PG images should not.
3. This process was repeated for all four hash algorithms.

3.4 Evaluation Metrics

To compare the different approaches, we made use of some metrics to evaluate their performance. For reference, positives refer to the modified images, while negatives refer to the PG images.

- Accuracy measures the overall effectiveness of the approach by considering both positive and negative predictions.
- Precision measures how many predicted positives are actually positive.
- Recall measures how many of the actual positives or negatives were correctly predicted as positive by the approach.
- F1 Score provides a balanced view of precision and recall.

For the formulae we used to evaluate these metrics, refer to Appendix 4.

4. Results and Discussion

4.1 Results

Below, we present the data of the various approaches collected from our experiments.

4.1.1 Performance of various approaches in detecting visually similar images

Hash Algorithm	Threshold Condition	Accuracy	Precision	Recall	F1-Score
dHash	≤ 0.334	89.18	100.00	78.22	87.78
pHash	≤ 0.340	88.81	100.00	77.48	87.31
wHash	≤ 0.191	88.76	99.18	78.04	87.34
NMFHash	> 0.952	75.45	97.73	51.81	67.72

Table 1. Thresholds for magazines dataset and comparison of hash algorithms using performance evaluation metrics. All values are in percentages (%).

Accuracy	Precision	Recall	F1-Score
89.18	100.00	78.22	87.78

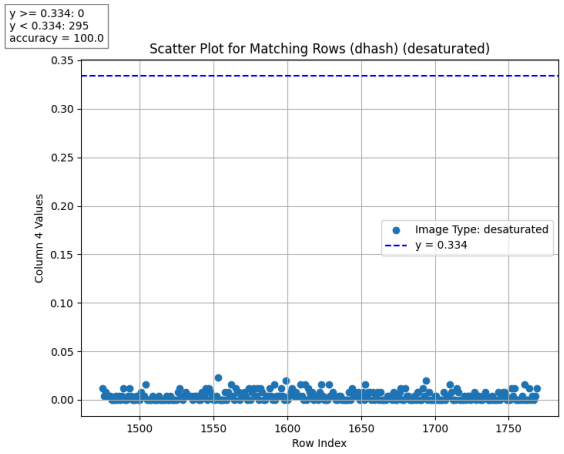
Table 2. Analysis of Majority Decision Approach performance using evaluation metrics. All values are in percentages (%).

Accuracy	Precision	Recall	F1-Score
95.12	99.80	90.36	94.84

Table 3. Analysis of Decision Tree performance using evaluation metrics. All values are in percentages (%).

Accuracy (%)	Precision	Recall	F1-Score
91.94	92.07	90.78	91.42

Table 4. Decision Tree Performance on People Dataset



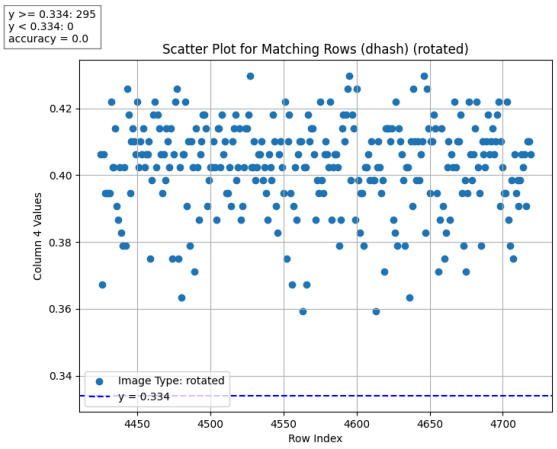


Figure 2. dHash performance for desaturation and rotation content modifications

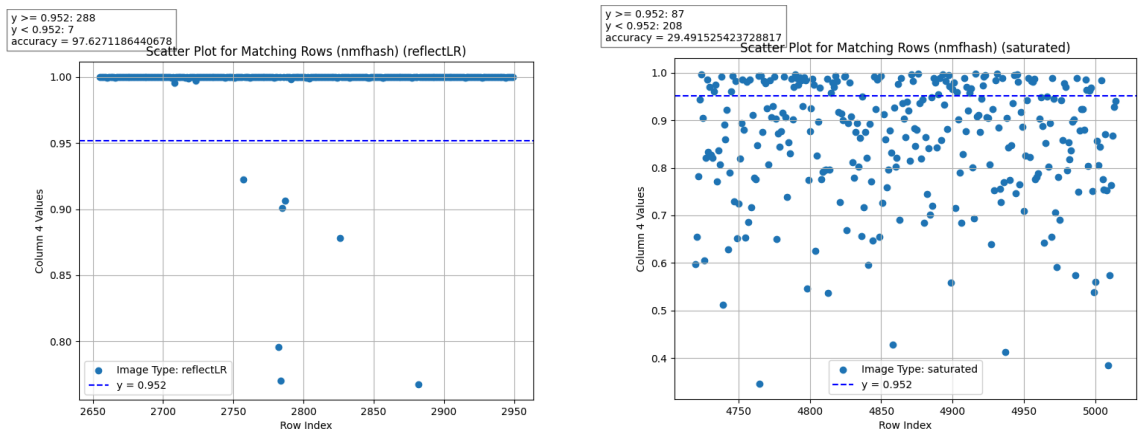


Figure 3. NMF hash performance for reflection and saturation content modifications

4.1.2 Time and Memory

dHash, pHash and wHash all take relatively similar time taken to hash images, whereas NMFHash is significantly slower, likely because its algorithm normalises the image to 512 x 512 before processing while the others normalise to 16 x 16. Interestingly, when compared to SHA256, a cryptographic hash, dHash performed slightly faster. In terms of memory, dHash consumes the most memory, followed by NMFHash, then pHash and SHA256 which have similar memory usage. wHash uses the least memory. The performance of individual hashes should be considered in the choice of hashes for any approach depending on the use case. For example, users may want to exclude NMFHash in an application where bulk image transfers are expected due to its slow speed. Refer to Appendix 5 for the data of the hashes' time and memory consumption.

4.2 The Flaws of Majority Decision

The ImageHash hashes' strong ability to detect colour-changing modifications (Figure 2) is reflected in the high accuracy, precision and recall across the three hashes, with dHash performing the best (Table 1). Conversely, NMFHash performs well in handling geometrical modifications such as reflections due to its construction of a secondary image from ring partitions of the primary image. However, it struggles with most other image alterations, such as saturation (Figure 3). Refer to Appendix 6 for a full set of graphs of every hashes' performance against various modifications.

The above observations reflect the ability of different hash algorithms to complement one another in detecting a wide variety of content-preserving modifications. They suggest the great potential of different hash algorithms in working together to improve the accuracy of detecting visually similar images. A naive approach to doing so is the Majority Decision approach.

In the Majority Decision approach, the input image was evaluated based on every hash type, with a separate verdict for each. The final verdict was decided based on the majority verdict of the four hashes. In the case of a tie, the dHash verdict was used to tiebreak, as it produced the most

reliable results for an individual hash (Table 1). Table 2 shows the results and performance of this approach.

The performance of this approach was very similar to that of dHash, the best performing individual hashing algorithm, when tested on the same Magazines dataset. This is likely because three out of the four hash algorithms used (dHash, pHash, wHash) perform similarly and are effective in identifying the same types of image modifications, such as noise and colour changes (Figure 2), and this is enough to constitute a majority in this approach. While NMFHash performs well against geometric modifications like reflection (Figure 3), its different result alone is insufficient to alter the final verdict. Additionally, since dHash was used as the tiebreak algorithm, its outcome likely weighs more in the final verdict overall. Hence, this approach was able to perform just as well as the best performing individual hash algorithm, but was unable to further improve on accuracy and F1 score beyond that.

4.3 Advantage of Decision Tree

We elaborate on the construction of the decision tree in Approach 3 as mentioned in Section 1.1. Given the ability of machine learning as a powerful tool to process large amounts of data, we passed the data of the similarity scores of all modified and PG images generated by all four hash algorithms into a Decision Tree Classifier. The decision tree takes the four hash similarity scores as input, processes them with unique thresholds at each node, and outputs a verdict regarding whether the image is visually similar to an R21 image in the dataset. The Magazines dataset was split randomly into training and testing sets, such that 80% of the data was used for training the machine learning model, while 20% was used for testing the model's predictions. The same 20% of data was used in testing all three approaches from Section 1.1, producing the results in Section 4.1. The depth of the tree was limited to 3 to avoid overfitting of data, as increasing the depth further had no significant positive impact on the results.

Figure 3 shows the nodes of our decision tree as well as the threshold condition for splitting at each node, while Table 3 shows the performance of the decision tree classifier.

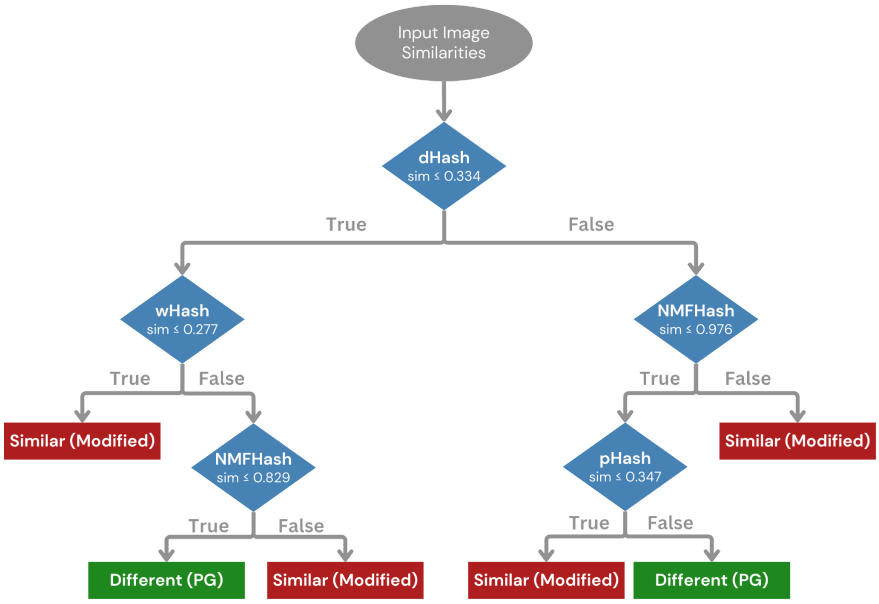


Figure 3. Decision Tree Visualisation

While the Majority Decision approach fails to play to the strengths of each hash algorithm sufficiently, the decision tree is able to improve the overall accuracy of detecting visually similar images, with thresholds determined by analysing complex trends in the similarity score data outputted by each hash algorithm using machine learning. It performs the best overall out of the three perceptual hash application approaches, especially regarding accuracy and F1 Score (Table 3). It is also able to achieve significantly lower false negative rates than both the majority decision approach and any individual hash. This improvement in false negative rates is especially important in the context of E2EE content moderation, as any undetected harmful content allows for further dissemination in the absence of human moderators due to the privacy-preserving nature of E2EE communications.

4.4 Generalisability of Decision Tree Approach

We also tested our threshold conditions in all three approaches on the People dataset, a different set of images from Magazines as mentioned in section 3.1. All the threshold conditions used were the same as those obtained from the Magazines dataset, and were not recalibrated to fit this new dataset. The decision tree is also static, and was not retrained on the People dataset. The results for the decision tree approach are presented in Table 4. For results of the other two approaches, refer to Appendix 7.

The decision tree still performs quite well, with generally high accuracy and F1 Score, even on a completely different dataset. The accuracy of 91% is still higher than that of the other two approaches, even for the Magazines dataset from which the relevant thresholds were determined. Thus, the decision tree approach is highly generalisable to R21 content, and it is not only specific to the one type of image (magazines) that it was trained on. We believe the approach can also be extended to detecting visually similar image hashes to datasets of perceptual hashes of other kinds of images, such as copyrighted images or deep fake images of political figures.

4.5 Library Implementation

We have implemented a Python library that implements the different approaches to utilising the four hashes. It can be accessed at https://pypi.org/project/unihasher/ and includes both library usage code and a Proof-of-Concept Socket application. Details on the hash algorithm implementations used can be found under Section 3.2.

5. Conclusion

From our experimental results, it can be concluded that the decision tree approach allows for identification of a wider range of image modifications as visually similar to the original images, compared to simply implementing individual hash algorithms or the naive approach of Majority Decision. This is observed from the decreased rates of false negatives, as different hash algorithms perform better on specific image modifications. Hence, the approach of using decision trees to combine and improve on the performance of different perceptual hashes algorithms is a promising one that improves overall robustness in detection of visually similar images. Developers can also utilise our Python unihasher library to integrate or extend our novel decision tree approach into their content moderation processes to detect visually similar variants of problematic images with the assurance of improved robustness from our empirical tests.

5.1 Future Work

With extensive literature on various perceptual hash algorithms available, each with their own strengths towards certain content-preserving modifications, an area of future work may involve extending our highly flexible approach of decision trees to include more combinations of different hash algorithms as available in the literature, with [6] providing a comprehensive comparison of many existing hash algorithms. Furthermore, we believe that the approaches proposed in our paper can be extended to improving the overall robustness of perceptual hash algorithms aiming to evaluate similarity of video content, such as Facebook's TMK algorithm [7].

6. Acknowledgements

We would like to thank our mentor Ruth Ng Ii-Yung for introducing us to cryptography concepts and the technical workings of end to end encryption, and supporting us throughout our research journey. We would also like to thank our mentor Chan Ming Kai Alvin for assisting us in our implementations, and providing us with valuable guidance.

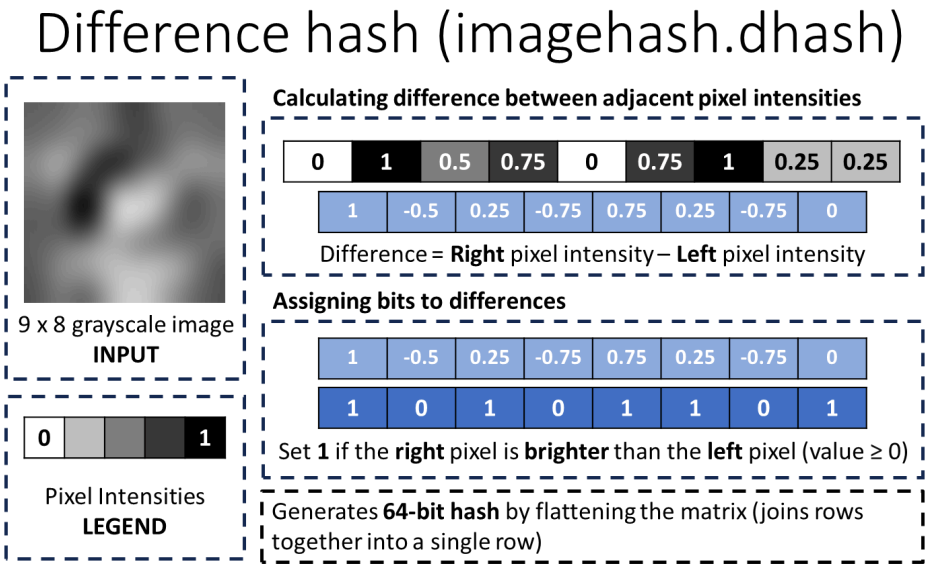
References

[1] S. Sharma, "Analysis of Perceptive Hashing Algorithm in Image Manipulation Detection," 2023 International Conference on Computational Intelligence, Networks and Security (ICCINS), Mylavaram, India, 2023, pp. 1-5, doi: 10.1109/ICCINS58907.2023.10450033.
[2] Sean McKeown, William J. Buchanan, Hamming distributions of popular perceptual hashing techniques, Forensic Science International: Digital Investigation, Volume 44, Supplement, 2023, 301509, ISSN 2666-2817, https://doi.org/10.1016/j.fsidi.2023.301509
[3] Hamadouche, Maamar & Zebbiche, K. & Guerroumi, Mohamed & Hanane, Tebbi & ZAFOUNE, Youcef. (2021). A comparative study of perceptual hashing algorithms: Application on fingerprint images.
[4] Buchner, J.. Image hash Python library. Available at: https://github.com/JohannesBuchner/imagehash?tab=BSD-2-Clause-1-ov-file#readme
[5] Z. Tang, X. Zhang and S. Zhang, "Robust Perceptual Image Hashing Based on Ring Partition and NMF," in IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 3, pp. 711-724, March 2014, doi: 10.1109/TKDE.2013.45.
[6] M. Roy, D. M. Thounaojam and S. Pal, "Various Approaches to perceptual image hashing systems-A Survey*," *2023 International Conference on Intelligent Systems, Advanced Computing and Communication (ISACC)*, Silchar, India, 2023, pp. 1-9, doi: 10.1109/ISACC56298.2023.10083762.
[7] Facebook TMK hash library. Available at: https://github.com/facebook/ThreatExchange/tree/main/tmk
[8] PG images for the Magazines dataset sourced from Vogue Archive https://archive.vogue.com/
[9] R21 images for the Magazines dataset sourced from Playboy Internet Archive https://archive.org/details/@playboy_archive
[10] PG images for the People dataset sourced from People Image Dataset on Kaggle https://www.kaggle.com/datasets/ahmadahmadzada/images2000
[11] R21 images for the People dataset sourced from nsfw_data_source_urls https://github.com/EBazarov/nsfw_data_source_urls

Appendix

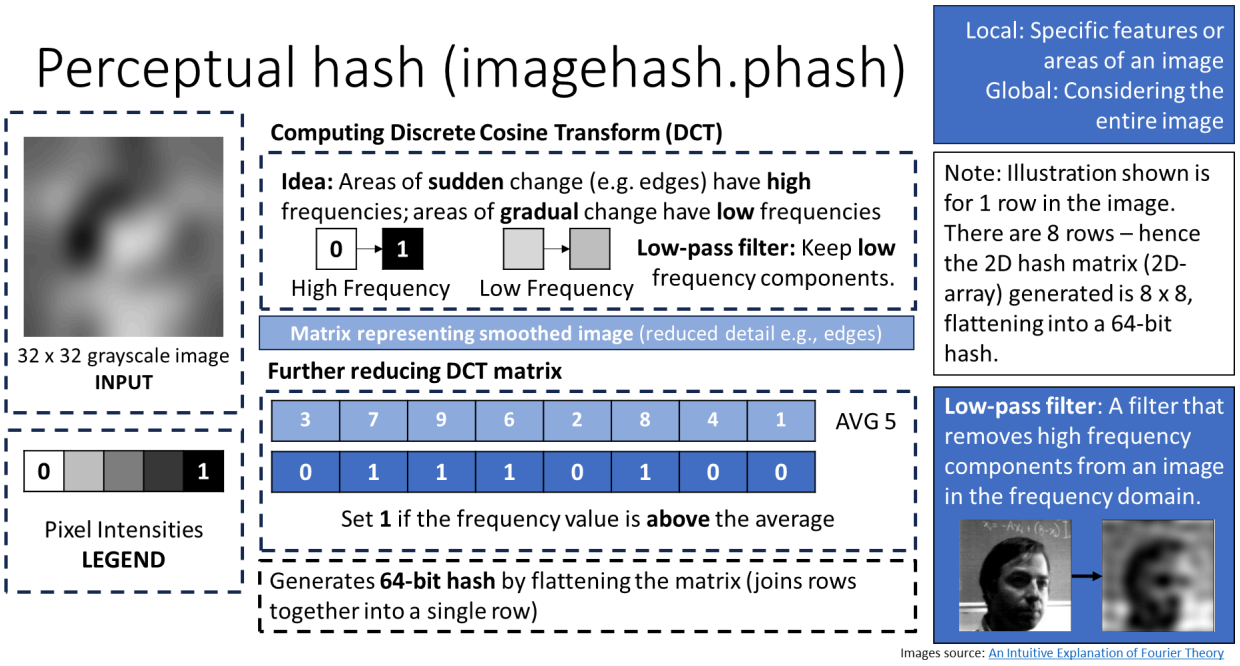
Appendix 1: Details on hash algorithms

Difference hash (dHash)



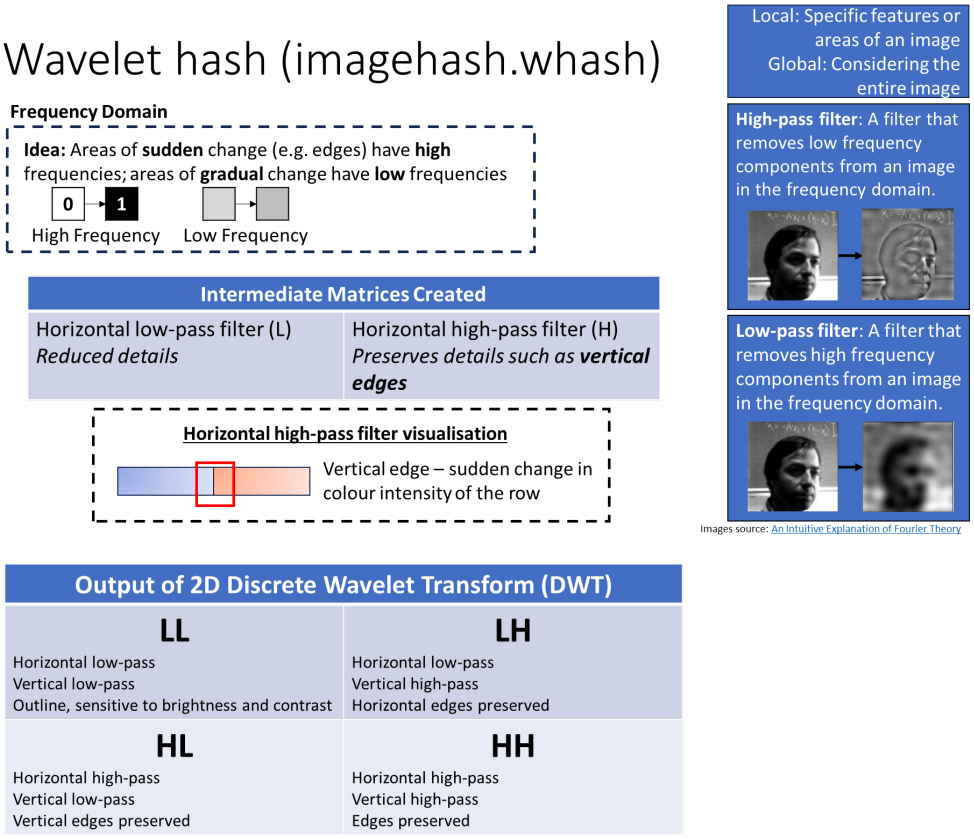
For a more comprehensive overview, refer to the HackerFactor article available at https://www.hackerfactor.com/blog/index.php?/archives/529-Kind-of-Like-That.html

Perceptual hash (pHash)



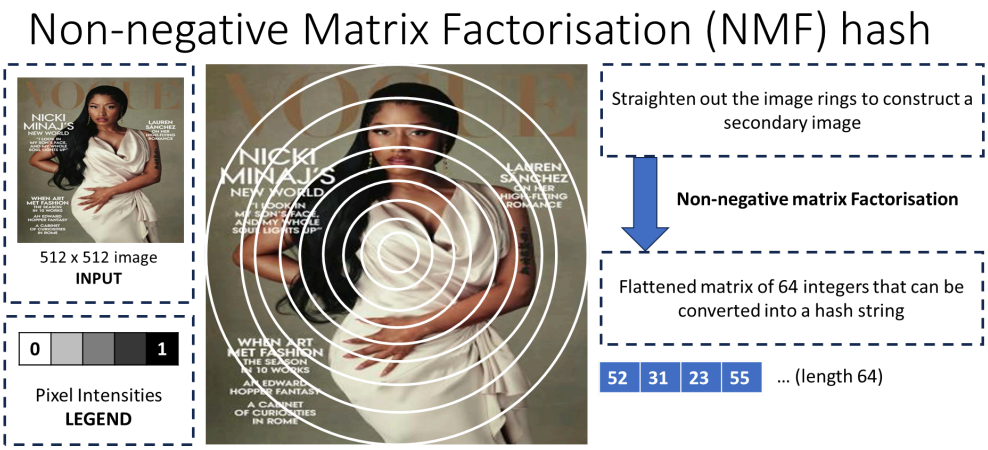
For a more comprehensive overview, refer to the HackerFactor article available at https://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html

Wavelet hash (wHash)



For a more comprehensive overview, refer to the Medium article available at https://fullstackml.com/wavelet-image-hash-in-python-3504fdd282b5

Non-negative Matrix Factorisation Hash (NMFHash)



For a more comprehensive overview, refer to the NMF paper available at [5]

Appendix 2: Image modifications applied to each image in the dataset

Modification	Type	Definition

Brightness	Dark	Decrease/increase brightness
	Bright	
Black and white		Change image to greyscale
Contrast	Low	Decrease/increase image contrast
	High	
Crop (5%)		Remove 5% of the image from the top, left, right, and bottom of the image, in effect reducing the overall pixel count to 81% of the original (0.9 *height* x 0.9 *width*).
Gaussian Blur		Blur image
Mirror	X-axis	Flip the image on its x/y-axis, preserving viewability, but generating significant pixel/binary level changes.
	Y-axis	
Noise	Colour	Add random bright and dark pixels
	Gaussian	Pixel values are changed by a normal distribution of variations
	Speckle	Create grainy variations in pixel intensity
Resize	32 x 32	Resize the image to the specified size
	64 x 64	
	128 x 128	
	256 x 256	
Rotate		Rotate the image by 45 degrees, leaving the background as black space
Saturation	Desaturated	Decrease/increase image saturation
	Saturated	

Appendix 3: Dataset construction

PG images for the Magazines dataset were downloaded from [8] manually, while R21 images were downloaded from [9] manually. The full dataset that we used can be found at https://drive.google.com/file/d/1sMX46jmYc285Z07sSkJa36R7VuuhwxoH/view?usp=drive_link.

PG images for the People dataset were downloaded from [10] and cleaned manually. It can be found at https://drive.google.com/file/d/18anMXPvSePLmQaW2aZPT8EqLDwNaUqZl/view?usp=drive_link. The R21 images were sourced from [11] via GET requests and cleaned. The GET request script used to construct the People dataset can be found at https://drive.google.com/file/d/1aSOlcbx-Vyx2rx9v7FupLkp5FU8CyUfu/view?usp=drive_link.

Appendix 4: Content moderation evaluation metric formulae

Formula
$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$
$Precision = \frac{TP}{TP + FP}$
$Recall = \frac{TP}{TP + FN}$
$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$

Appendix 5: Time and memory comparisons for hashes

Algorithm	Number of Images	Time Taken (seconds)	Memory Used (bytes)
dHash	30	0.2536	7,221,248
	50	0.4411	5,775,360
	50	0.494	5,619,712
	50	0.4006	5,574,656
	60	0.4999	7,467,008
NMFHash	30	39.0331	6,918,144
	50	71.8844	7,761,920
	50	71.6007	6,946,816
	50	72.681	7,831,552
	60	91.5425	7,270,400
pHash	30	0.318	5,148,672
	50	0.4534	4,444,160
	50	0.4434	4,104,192
	50	0.4671	3,842,048
	60	0.4714	4,272,128
sha256	30	0.2621	5,296,128
	50	0.437	4,063,232

	50	0.6055	3,780,608
	50	0.469	3,194,880
	60	0.5061	5,201,920
	30	0.5607	1,794,048
	50	0.9065	1,695,744
	50	0.9001	2,473,984
	50	0.9755	2,674,688
wHash	60	1.1385	2,699,264

Appendix 6: Graphs on performance of perceptual hash algorithms in detecting various content-preserving modified image variants (Magazines dataset)

Please access the following link to view the graphs: https://drive.google.com/drive/folders/10F0roXVi_TI6MdbGczX_MEtcuKpcGwBY?usp=drive_link

Appendix 7: Results from testing on People Dataset

Individual Hashes

Hash Algorithm	Threshold Condition	Accuracy	Precision	Recall	F1-Score
dHash	≤ 0.334	86.70	90.71	80.09	85.07
pHash	≤ 0.340	87.83	94.35	79.01	86.00
wHash	≤ 0.191	84.70	84.94	82.24	83.57
NMFHash	> 0.952	74.53	90.96	51.26	65.57

Table 5. Thresholds for People dataset and comparison of hash algorithms using performance evaluation metrics. All values are in percentages (%).

Majority Decision

Accuracy	Precision	Recall	F1-Score
87.86	93.43	79.96	86.17

Table 6. Analysis of Majority Decision Approach performance using evaluation metrics. All values are in percentages (%).