Cihan University-Erbil Scientific Journal (CUESJ)

Research Article

A New Scheme for Removing Duplicate Files from Smart Mobile Devices: Images as a Case Study

Ammar Asaad, Ali Adil Yassin Alamri*

Department of Computer, Education College for Pure Sciences, University of Basrah, 61004 Basrah, Iraq

ABSTRACT

The continuous development of the information technology and mobile communication world and the potentials available in the smart devices make these devices widely used in daily life. The mobile applications with the internet are distinguished simple, essay to use in any time/anywhere, communication between relatives and friends in different places in the world. The social application networks make these devices received several of the duplicate files daily which lead to many drawbacks such inefficient use of storage, low performance of CPU, RAM, and increasing consumption battery. In this paper, we present a good scheme to remove from the duplicate files, and we focus on image files as a common case in social apps. Our work overcomes on the above-mentioned issues and focuses to use hash function and Huffman code to build unique code for each image. Our experiments improve the performance from 1046770, 1995808 ns to 950000, and 1981154 ns in Galaxy and HUAWEI, respectively. In the storage side, the proposed scheme saves storage space from 1.9 GB, 1.24 GB to 2 GB, and 1.54 GB, respectively.

Keywords: Duplicating images, hash code, mobile device, performance, storage management

INTRODUCTION

One of the key things that can help us to build a scheme to get rid of this issue is the features extracted from images. It is difficult to observe and understand these features by a human{Shyu, 1998 #2343}. These features divide into two types global and local.[1] Image matching is the method used either global features which describe the whole image and can describe a whole image by only one vector, or local features, which focus on important details in the image and it is more powerful. To apply both methods, we need to deal with the content of the images and extract information from them; this means that both methods consume time to treat with images and need to have hardware that has ability and efficiency to get these features. This leads us to the fact that the features extraction in the mobile device will consume a lot of time, random memory, and cup time thus increasing the consumption of the battery; as a result, effect on daily use for the user.[2] Hence, the process of finding the best and most efficient methods of retrieval and matching files is the most important research topics in the era of mobile applied technology.[3] Matching is one of the essential tasks that are used to remove duplicated files. Mostly, it is selected the appropriate features to reduce the computational time, and it denotes to find the same file/s (image) save(s) in a database that is/are same to an input file. Matching is one of the indispensable processes in several applications.[4] In the past decade, mobile technology and internet have become a hotspot in scientific research and daily life.[5] Social media consider a common element between internet and mobile devices; it plays the main role to keep in touch with our colleagues/relatives and discovering new persons.[6] In the past few years, social network was not used in mobile devices in widely manner, but with the rapid development in the internet and mobile technologies, the social media become one of the important factors in human life.[7] At present, there are several social networks that work in mobile devices and personal computers such as Facebook, WhatsApp, Twitter, Viber, imp, and others called mobile apps.[5] The users of these apps exchange many types of files such as photos, audios, and videos. Ultimately, these files are stored in duplicity way led to lose of storage space of the mobile device. In this paper, we focus on the removal of duplicate images issued from social media apps to mobile device based on proposed image matching scheme that suits with mobile environment and deals of the whole image. The main objectives of our purpose scheme are as follows:

Corresponding Author: Dr. Ali Adil Yassin Alamri,
Department of Computer Science, Education College for Pure Science, University of Basrah, Basra, 6100, Iraq.
E-mail: en.uobasrah.edu.iq/ali.yassin@uobasrah.edu.iq

Received: Apr 08, 2019
Accepted: Apr 14, 2019
Published: Aug 20, 2019

DOI: 10.24086/cuesj.v3n2y2019.pp5-13

1. Our work can eliminate this repetition of images saved in mobile's storage from different social media apps with high performance
2. Our scheme is offered cost of consumption energy of battery and less used for both RAM and CPU
3. We use a hash function algorithm (message-digest algorithm 5 [MD5]) and Huffman code algorithm to build unique code for each image which is used to delete all reputation images
4. Our proposed scheme is applied on real-word data using two types of mobile devices (HUAWEI CUN-U29, Galaxy A5 [2017] SM A520F) and we achieve good results to increase storage space and improve the processing time.

Organization of the paper, Section 2 shows the relevant literature on the topic of image matching. Section 3 refers to the primitive tools of the proposed scheme while Section 4 explains the proposed scheme. Section 4 focuses on experimental results and discussions. The paper is concluded in Section 5.

RELATED WORKS

This section briefly discusses duplication files in a mobile device, refers the several studies of deduplication schemes as follows. Attractively, there are many fields to use deduplication such as paging files, image redundancies, and memory sectors.[8,9] This field has been applied primarily on data backup services such as Microsoft Office and Google Drive.[10] In addition, some researches in the field of removing duplicate files focus on multiple deduplication schemes.[10] At present, with rapid development in cloud computing and mobile device, there is a platform to support low latency network access and extra storage called mobile cloud computing (MCC).[10] Besides, MCC enables to support mobile in extra storage based on pay-as-you-go within cloud computing principle, but this technique leads to pay cost by the customer,[9,11] In the same time, the duplicate files are stayed in the storage of the cloud. Storer *et al*.[12] suggested a good scheme eliminated redundancies files depending on its type. However, the method of deleting duplicate files is selected manually. It works well with known files and suffers to deal with new files that do not exist in the index file of their work.[10] Haustein *et al*.,[13] the patent is about selecting a deduplication method depending on the file type and deduplication rate, where the redundant deletion is done by the server. They used the ratio of duplication files and then the implemented scheme operates based on this ratio to deduplication files from the server. Their works had many drawbacks by working in server-side and they proposed in a theoretical manner without executed in real-world data. Widodo *et al*.[10] presented a scheme to reduce energy consumption and the amount of data by detection duplicate files. This scheme suffers from low of duplicate detection performance and deduplication throughput for a few files in the beginning. Although their works focused on using cloud storage to duplicate files, the duplications files are stilled in devices' storage. There are several researchers focused on detection of duplicate data using hash functions such as MD5 and SHA-256.[14,15] These schemes cannot face the collisions growing the size of the signature increases as well as the processing time is increased with the size of signatures.[16] In this paper, we present a scheme that has been applied as mobile apps to remove the duplicate files imported to mobile by social media applications such as Facebook, WhatsApp, Viper or mobiles camera. The image files are one of most commonly used in social networks world, therefore, we focus on the image files as a case study in our experimental results relied on two types of mobile devices "Samsung and Hawawi." Our work focus to use hash function and Hffuman code to build unique code for each image and overcomes the drawbacks of collisions grows as the size of the signature increases. However, the scheme generates a good search index file used to insert an image or ignore in the repeating case. Moreover, our scheme is offered cost of consumption energy of battery and less used for both RAM and CPU of the mobile device. In addition, our work can manage the storage of mobile and cloud storage (based on some of Google's application such as Google Drive and Google Photos) by removing duplicate images.

PRIMARY TOOLS

In this section, we demonstrate the main tools used in our work to achieve the vital goals of the research.

Huffman Code

Huffman coding considers one of the most important algorithms proposed and published by David A. Huffman.[17] It is based on the frequency of the amount of a data item. Huffman algorithm works using the tree data structure to generate an optimal binary tree called Huffman tree to acquire encoding form.[17,18] The technical working of this algorithm uses a lower number of bits to encode the input data that exist more frequently.[19] Practically, Huffman considers an easy-to-use algorithm because of its simpler mathematical computation to obtain the several parameters.[20] Huffman coding algorithm consists of two phases: Building binary tree and Generating code. The result of the first phase is constructed through the occurrence frequency of each item. To achieve the code of the Huffman algorithm, each branch has a set (0 or 1) based on its direction left (0) or right (1).[19,21] Figure 1 explains the mechanism work of Huffman code algorithm.

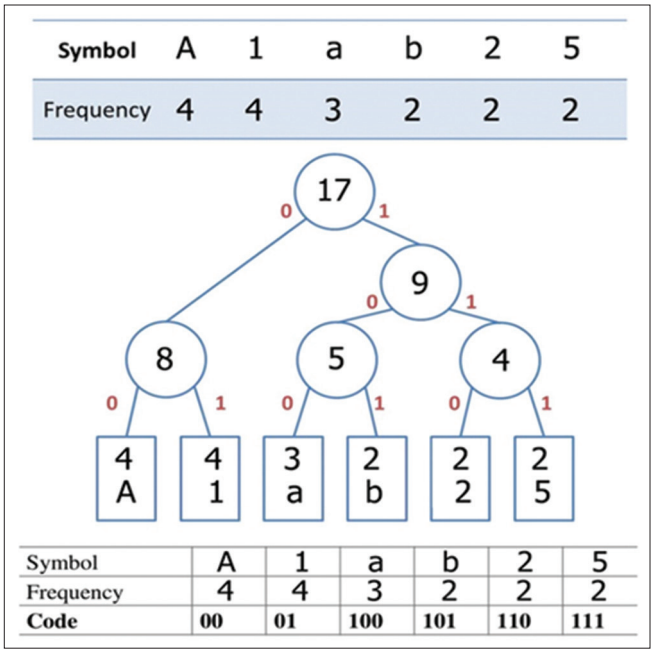


Figure 1: Explain the Huffman code algorithm

MD5 (Hash)

MD5 is one of the most algorithms worked to compress data files such as images, videos, and audios. The output of this algorithm is a fixed value (128-bit) when the algorithm is implemented once or many times on the same data.[22,23] This algorithm was developed by the Rivest[24] and considered simple to compute the MD5 value. The MD5 processes 512-bit blocks and breaks into 32-bit words. It includes 64 rounds. The MD5 steps as follows:[8] The first process is to add the bits to the block to equal the value of (448 mod 512), start with 1, followed by 0's, the second process is adding the length of the message to the value (448 mod 512), the third step set the MD5 register values (A, B, C, and D), each one 32 bits (hexadecimal number), and the fourth step is the heart of the algorithm; its four pressure functions each of these pressure functions have logical operations [Figure 2].

PROPOSED PROTOCOL

Our proposed work consists of two phases: Creating index file and removing daily images. The first phase focuses on detecting and creating an index file (IF see Table1 explain the Notification of symbols) without duplicate images, while the second one is implemented to remove images on a daily basis. Our work focuses on a vital part of smartphone when receives daily hundreds of photo files, video clips, audio, and others. There are many of these files repeated in mobile's storage. The proposed scheme plays the main role in deleting duplicated photos that caused to lose a large space of storage. Each a new photo checked inside IF (index file), it removes from storage when it saved in IF previously. Otherwise, a new photo inserts to the IF. As a result, the mobile device works more efficiently and performance as well as reduce energy consumption and disposal of redundancy of photos in the memory. The limited of hardware components of the mobile device makes the proposed scheme dealing with photos in light and high-performance manner. Therefore, we used MD5 hash function and Huffman code to build unique code for each photo. Figure 3 explains the main aims of our work.

Table 1: Notification of symbols

Symbol	Description
MD	Mobile device such as smartphone and tablet…
MS	Mobile's storage (external, internal)
IMG_i	Image that is saved inside MD
S	Set of images (IMG_i)
p_i	IMG_i's path
P	Set of IMG's path
Size()	Function to get size of each IMG
IF	Index file has path of IMG and its details
$Code_1$	Number created by convert hash to ASSCI code and compute code by multiply ASSCI by position
$Code_2$	Number obtained by Huffman coding and convert to decimal number
H	MD5 hash function
h_i	Hash code for IMG_i
UC_i	Final code resulted by XOR $Code_1$ and $Code_2$
DelDuplicate()	Function to delete all duplicate IMG and save one only
Search()	Function to detect duplicate IMG
DFDS()	Function to delete IMG from device storage
DeleteRowIF()	Function to delete a row from IF
Save()	Save details in index file (UC_i, p_i, Size (p_i),)
GetImg()	Get IMG_i based on its p_i
GetImages()	Get all p_i

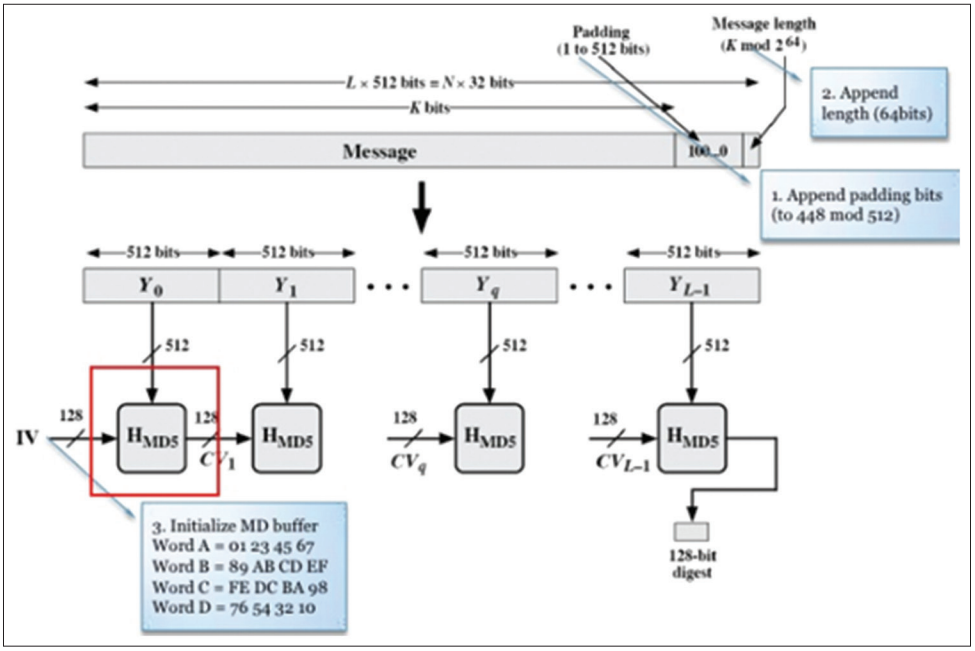


Figure 2: Algorithm mechanism and steps

Creating Index File Phase

In the era of information technology, all smartphones have several folders related to photos which received from different mobile applications such as WhatsApp and Facebook. Therefore, there are many duplicated images saved in mobile storage. These duplicated images have several drawbacks such slow mobile completion, the difficulty of communication, loss of flexibility to handle more than one application and slow the mobile response of the received orders from another device as well as power consumption due to slow performance. However, this phase is used once to create an index file and removing all duplicate images from mobile storage. The main steps of this phase are as follows:

a. Generate set of all mobile's images S = {img_1, img_2,… img_n} and their paths P = {p_1,p_2,…p_n} based on extension of images file such as JPEG, BMP.

b. Create unique code for each image ($img_i \in S$) based on hash (Eq.1 and Eq2.) and Huffman code functions as follows.

$$H_i = h(IMG_i) \quad (1)$$

$$Code_1 = \sum_{j=1}^{n} ASCII(H_i(j)) * position\left(H_i(j)\right) \quad (2)$$

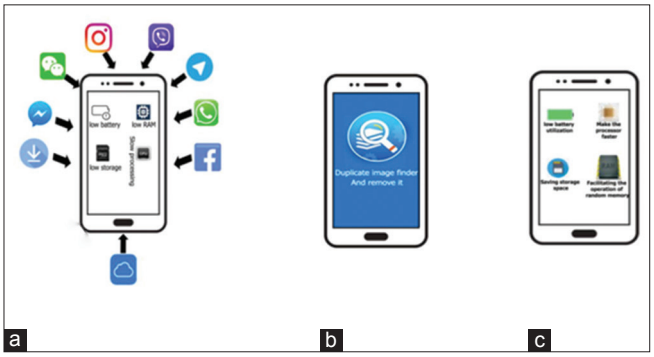


Figure 3: (a) Receiving daily images, (b) applying our work, and (c) the results of work

- Apply Huffman code function
- $Code_2$ = Huffman(hi),
- Unique Code = $Code_1$ XOR $Code_2$ (3)
- Figure 4 shows a code generation technique.

c. Create index file (IF) that each record consists of UC_i, P_i, Size(IMG_i). Algorithm 1 demonstrates the mechanism of generating IF to all mobile device images, [Figure 4].

Algorithm 1 of creating index file

Input: {Pi} 1 ≤i 1; where n is number of P

Output: Index File

Compute H_1 based on Equation (1)

Compute UC_1 based on Equation (2,3)

Save (UC_1, p_1, Size(p_1))

For i=2 to n do

Table 2: Some of the rules have been implemented on images entered into MD

Rules	Details
1	This rule describes receiving a picture through more than one program, and this image is in IF. All copies are deleted
2	This rule describes receiving a picture through more than one program, and this image is not found in IF. A single copy is saved and the rest of the copies are deleted
3	This rule describes receiving a picture through a single program and this image is not found in IF. This version is saved
4	This rule describes receiving a picture through a single program, and this image is found in IF. This version is deleted
5	This rule explains the reception of the same image more than once through a single program, and this image does not exist in IF. Hence, one copy is saved and the rest is deleted
6	This rule explains the reception of the same image more than once through a single program, and this image is in IF. All copies are deleted

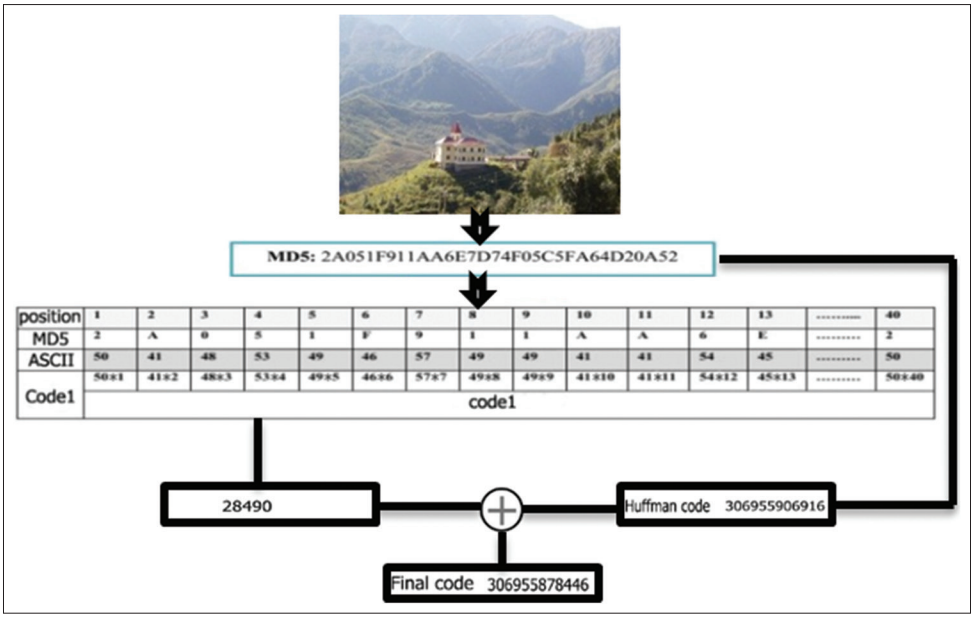


Figure 4: Demonstrates the process of generating unique code

Compute Hi as in Equation (1)

Compute UCi as in Equation (2, 3)

If (Search (UCi) =True)

DFDS (pi)

Else

Save (UCi, pi, Size(pi))

End for

End Algorithm 1 of Creating Index File.

Removing Daily Images Phase

All smartphones receive many messages on a daily these messages may be pictures or audio clips or video files or other and thus affect the storage space of the device as well as the performance of the device. Our scheme helps us get rid of the repetition for the images and thus let the machine work better. The removing of the repeated images depends on the code of image which has been computed in the first phase [Figure 5]. The main steps of this phase are as follows:

a. Generate set paths P' = {p'_1, p'_2,…p'_n } of mobile's image that incoming to device after creating an index file. These images have been imported to the device in daily.
b. Create set of unique code (UC' = {uc'_1, uc'_2,…uc'_n}) for each a new image based on the same procedure in step 2 of create index file phase.
c. Check $uc'_I \in$ IF(UC); if the result is false then extract a new record (p'_i, uc'_i) of img'i added to IF; otherwise, remove IMG_i from MDS. Tables 2 and 3 and Figure 3 denote to the

Table 3: Some scenarios for receiving images and how to process them

IMG	WhatsApp	Viper	Messenger	Download	Telegram	IF	Add to IF	Non add and removed from MDS	Rule apply
	√	√	×	√	√	√	×	√	Rule 1
	√	√	×	√	√	×	√	×	Rule 2
	√	×	×	×	×	×	√	×	Rule 3
	×	√	×	×	×	√	×	√	Rule 4
	√	×	×	×	×	×	√	×	Rule 5
	×	√	×	×	×	√	×	×	Rule 6

Table 4: Processing time of index file creation

Group	HUAWEI CUN-U29		Galaxy A5 (2017) SM-A520F	
	Size of group (KB)	Time of processing (nanosecond)	Size of group (KB)	Time of processing (nanosecond)
1-200	585080	32421400159	281353	10395266647
201-400	191979	11017601159	134797	5059068621
401-600	19271	1432455537	318393	11132803762
601-800	22118	1641043162	178462	6512452069
801-1000	18069	1359571455	132321	4641873378
1001-1200	16480	1335055147	12960	758972849
1201-1400	88118	5374266081	6760	401045846
1401-1600	202894	11694264538	-	-
1601-1800	34670	2326661993	-	-
1801-2000	5332	578131154	-	-
2001-2200	4897	540429308	-	-
2201-2400	69854	3991726847	-	-
2401-2600	81739	4908648533	-	-

Table 5: Add and remove received images

Day	HUAWEI CUN-U29			Galaxy A5 (2017) SM-A520F		
	Received	Add	Remove	Received	Add	Remove
1	37	7	30	63	13	50
2	27	7	20	18	8	10
3	18	8	10	12	8	4
4	19	15	4	20	12	8
5	12	8	4	50	30	20

mechanism of processing (adding/removing to IF) images which incoming to a mobile device (MD).

Algorithm 2 of adding new images

Input: {Pi} 1≤i≤n; where n is number of P,

Output: Add new images to Index File

For i=1 to n do

Compute H_i as in Equation (1)

Compute UC_i as in Equation (2, 3)

If (Search (UC_i) =True)

DFDS (p_i)

Else

Save (UC_i, p_i, Size(p_i))

End for

End Algorithm 2 of Adding new image

EXPERIMENTAL RESULTS

In this part, we evaluate the proposed scheme and use the images of the mobile device to guarantee the reproducibility of practical results. The experiments are implemented using Android Studio 3.3.1 running on Windows 10 64–bit operating system with an

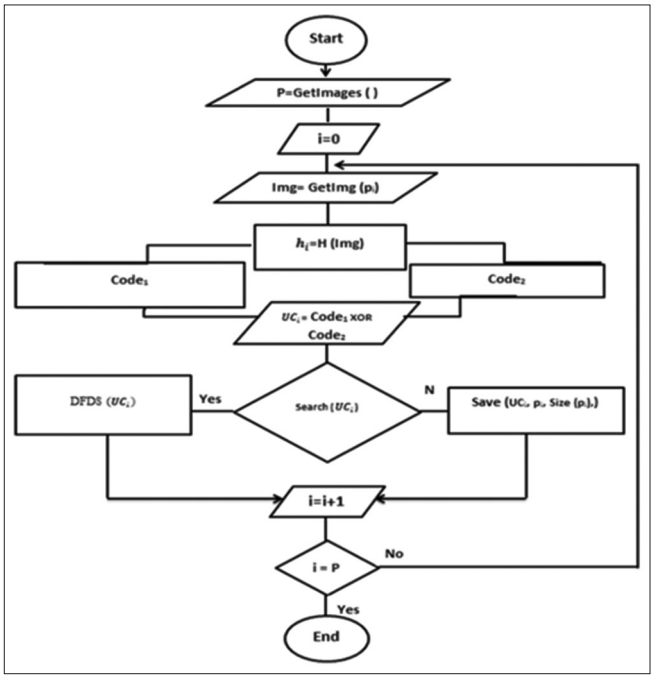


Figure 5: Explaining the main steps of the proposed scheme

Intel Core i5-2450M CPU at 2.50GHz 250GHz, 8 GB RAM, and 2.4 GHz CPU, and mobile HUAWEI CUN-U29 RAM 956.52 MB internal storage 4.15 GB with external storage 8 GB, battery 2200 mAh, Android version 5.1 (Lollipop_MR1), and another mobile Galaxy A5 (2017) SM-A520F, Android version 8.0 (Oreo), battery 3000 mAh, processor arm64-v8a 8 core, and total RAM 2815 MB. Our results have passed in many steps as follows.

Processing Time Calculation of Index File Creation

We used 2600 images to create index file (IF) in HUAWEI CUN-U29 mobile device. After that, we divided images into n groups (we obtained 13 groups based on the total number of

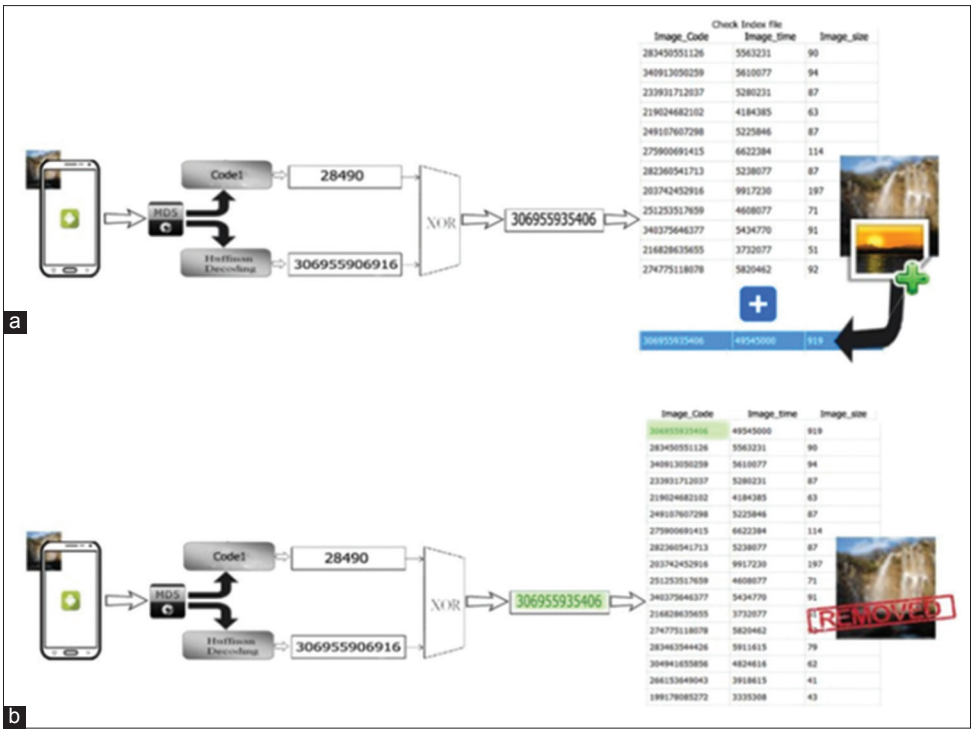
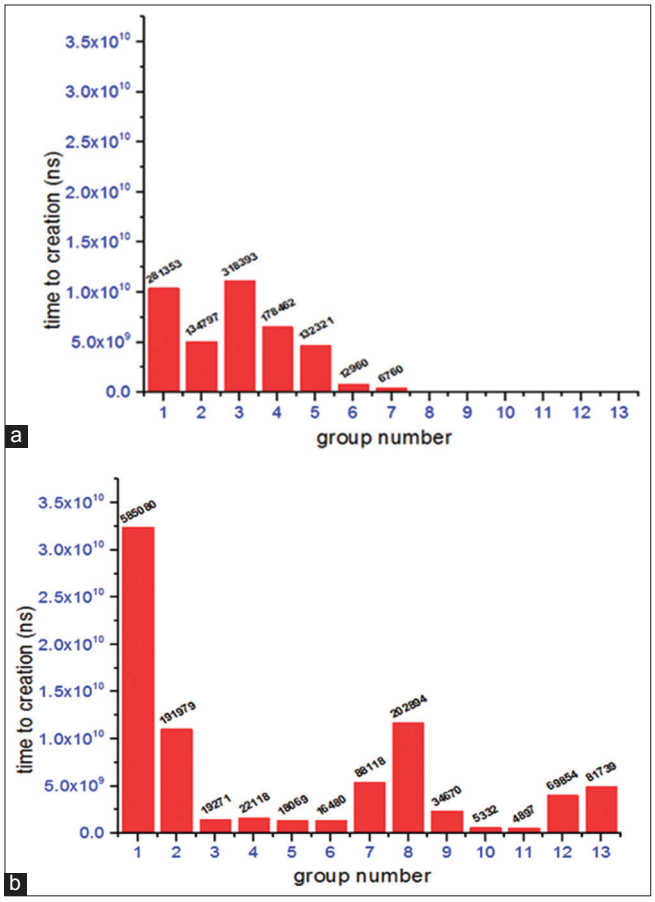


Figure 6: (a) It shows the process of receiving the image, generating the code, checking IF and making the decision, add of image in IF, (b) it shows the process of receiving the image from social media apps, generating the code, checking IF, and making the decision to delete image



AQ1 **Figure 7:** Processing time of index file creation (a) Galaxy A5 (2017) SM-A520F device for 1310 images (b) HUAWEI CUN-U29 device for 2600 images

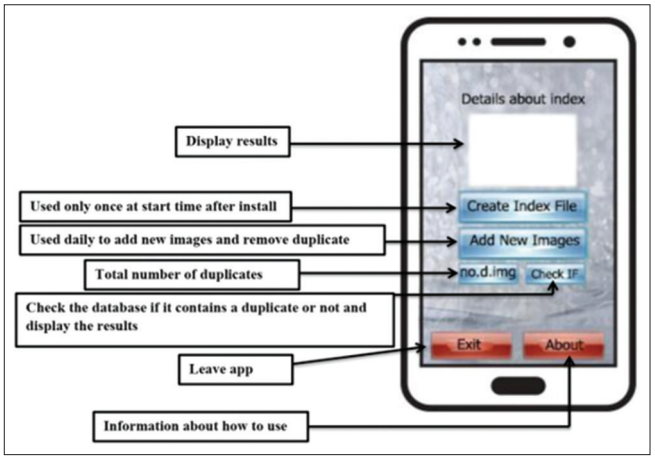


Figure 8: The mobile apps of our proposed scheme

images and each group consists of 200 images). In the other side, we implemented our work to another mobile device named Galaxy A5 (2017) SM-A520F contained 1310 images to construct index file (IF). Table 4 and Figure show the performance of the calculation of index file creation in both devices.

DAILY PHASE APPLY

We implemented our proposed scheme in real-word data saved on a mobile device that has properties as refereed in the previous section. Figure 6 shows the main operation in this phase. The proposed scheme has been implemented as mobile apps Figure 8 in several days to apply the main operations such as detecting and removing redundancy

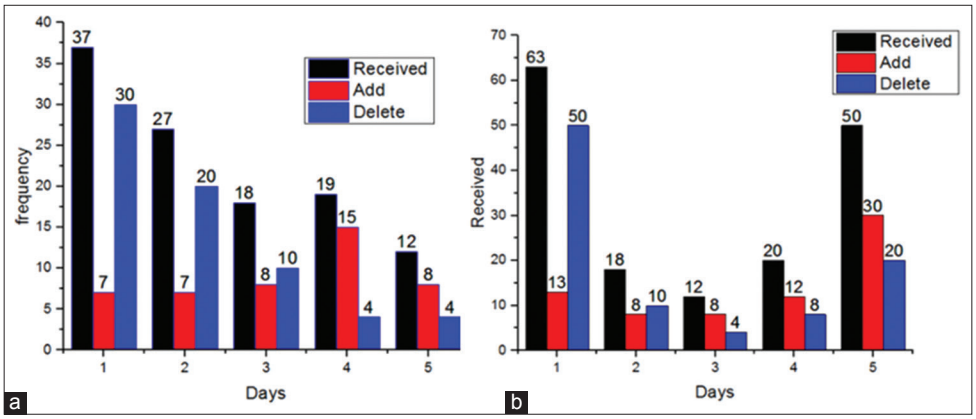


Figure 9: Add and remove images received (a) HUAWEI CUN-U29 device for 5 days received images (b) Galaxy A5 (2017) SM-A520F device for 5 days received images

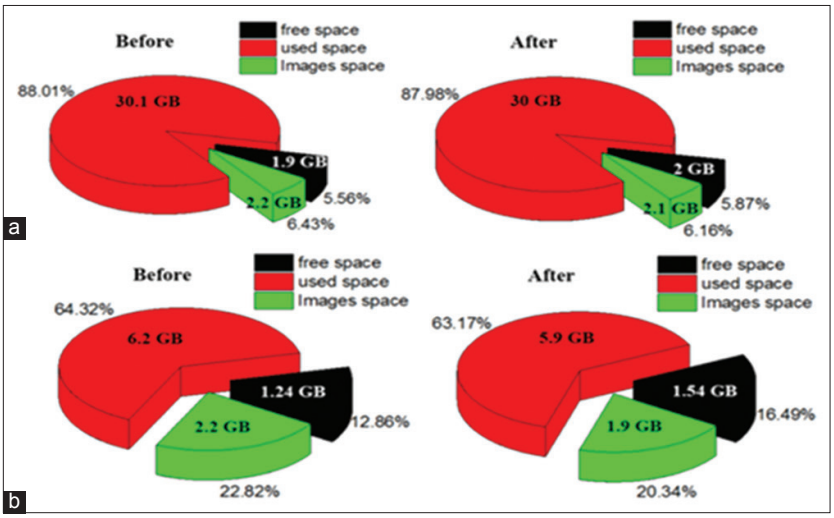
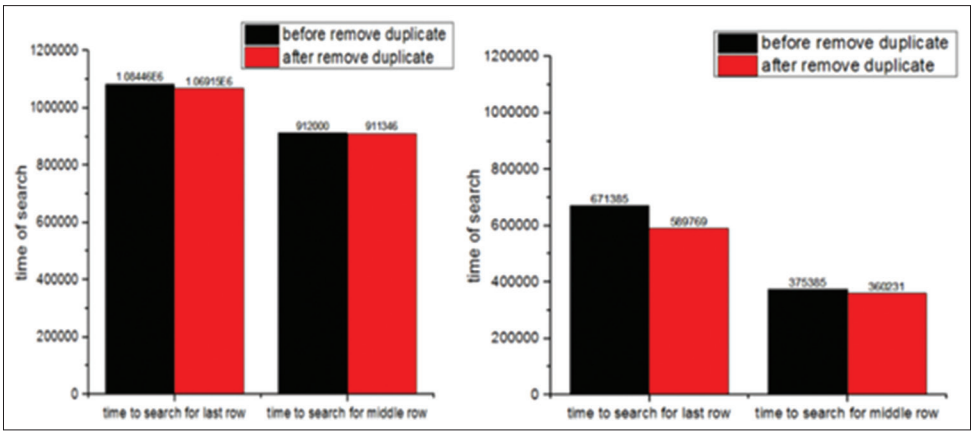


Figure 10: (a) Before and after apply for our work at Galaxy A5 (2017) SM-A520F and (b) before and after apply for our work at HUAWEI CUN-U29



AQ1 **Figure 11:** It shows the time spent searching for one image in the middle and one at the end. Before and after deleting duplicate files. In the two devices

images, adding new images. Table 5 and Figure 8 describe the central operations for 5 days. Furthermore, we test the mobile storage based on our proposed work and notice good results as Figure 9 displayed the storage of mobile devices (Galaxy A5 [2017] SM-A520F, HUAWEI CUN-U29) before and after implementing our proposed scheme. Moreover, we calculate the time of retrieved image on devices as in Figure 10.

CONCLUSIONS

One of the most important challenges for smartphone users is to keep the device working as high as possible. Low storage space is the most important issue. We solved this issue by the used hash function (MD5), Huffman code to generating unique code for each image and save this code in image in the index file and use it to rid of a duplicate from device's storage. Removing images have the same code and maintain one copy. The proposed scheme get a good result in remove this duplicate images in fast. We applied it at two devices (Galaxy A5 [2017] SM-A520F, HUAWEI CUN-U29). As a final result, the performance of device's CPU is better that makes the device work in the best way. As future work, in relation to the image matching schemes, the user may add something to the image, and as a result, new image obtained, but it still the same image with simple change. In future work, we will try to solve this issue.

REFERENCES

1. C. R. Shyu, C. Brodley, A. Kak, A. Kosaka, A. Aisen and L. Broderick. "Local Versus Global Features for Content-based Image Retrieval". In Proceedings. IEEE Workshop on Content-Based Access of Image and Video Libraries (Cat. No. 98EX173), 1998, pp. 30-34.
2. D. A. Lisin, M. A. Mattar, M. B. Blaschko, E. G. Learned-Miller and M. C. Benfield. "Combining Local and Global Image Features for Object Class Recognition". In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops, 2005, pp. 47-47.
3. M. Agarwal, A. Singhal and B. Lall. "3D local ternary co-occurrence patterns for natural, texture, face and bio medical image retrieval". *Neurocomputing*, vol. 313, pp. 333-345, 2018.
4. S. S. Hwang and S. H. Hong. "Pre-extraction of features and environment variable-based database filtering for fast image matching on mobile". *Mobile and Wireless Technologies*, vol. 2016, pp. 223-230, 2016.
5. J. Machajdik and A. Hanbury. "Affective Image Classification Using Features Inspired by Psychology and Art Theory". In Proceedings of the 18th ACM International Conference on Multimedia, 2010, pp. 83-92.
6. N. Lee, C. Kim, W. Choi, M. Pyeon and Y. Kim. "Development of indoor localization system using a mobile data acquisition platform and BoW image matching". *KSCE Journal of Civil Engineering,* vol. 21, pp. 418-430, 2017.
7. L. Wang and H. Wang. "Improving feature matching strategies for efficient image retrieval". *Signal Processing: Image Communication,* vol. 53, pp. 86-94, 2017.
8. K. Miller, F. Franz, M. Rittinghaus, M. Hillenbrand and F. Bellosa. "{XLH}: More Effective Memory Deduplication Scanners Through Cross-layer Hints". In Presented as Part of the 2013 {USENIX} Annual Technical Conference ({USENIX}{ATC} 13), 2013, pp. 279-290.
9. T. Liu, F. Chen, Y. Ma and Y. Xie. "An energy-efficient task scheduling for mobile devices based on cloud assistant". *Future Generation Computer Systems,* vol. 61, pp. 1-12, 2016.
10. R. N. Widodo, H. Lim and M. Atiquzzaman. "SDM: Smart deduplication for mobile cloud storage". *Future Generation Computer Systems,* vol. 70, pp. 64-73, 2017.
11. E. Ahmed, A. Gani, M. Sookhak, S. H. Ab Hamid and F. Xia. "Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges". *Journal of Network and Computer Applications,* vol. 52, pp. 52-68, 2015.
12. M. W. Storer, K. Greenan, D. D. Long and E. L. Miller. "Secure Data Deduplication". In Proceedings of the 4th ACM International Workshop on Storage Security and Survivability, 2008, pp. 1-10.
13. N. Haustein, C. A. Klein, U. Troppens and D. J. Winarski. "Method of and System for Adaptive Selection of a Deduplication Chunking Technique". Google Patents, 2009.
14. B. Zhu, K. Li and R. H. Patterson. "Avoiding the Disk Bottleneck in the Data Domain Deduplication File System". In Fast USENIX Conference, 2008, pp. 1-14.
15. D. Meister and A. Brinkmann. "Multi-level Comparison of Data Deduplication in a Backup Scenario". In Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference, 2009, p. 8.
16. Y. Won, R. Kim, J. Ban, J. Hur, S. Oh and J. Lee. "Prun: Eliminating Information Redundancy for Large Scale Data Backup System". In 2008 International Conference on Computational Sciences and Its Applications, 2008, pp. 139-144.
17. M. Sharma. "Compression using huffman coding". *IJCSNS International Journal of Computer Science and Network Security,* vol. 10, pp. 133-141, 2010.
18. R. Li, Y. Zhao, Q. Xu and X. Qi. "Research of Subnetting Based on Huffman Coding". In International Conference on Cloud Computing and Security, 2018, pp. 606-616.
19. O. Yue-Long, L.N. Zhang and N. Yu. "Researching on MD5's characteristics based on software reversing". *The Journal of China Universities of Posts and Telecommunications,* vol. 17, pp. 127-130, 2010.
20. I. C. Lin and L. C. Yang. "A Noise Generation Scheme Based on Huffman Coding for Preserving Privacy". In International Conference on Security with Intelligent Computing and Big-data Services, 2017, pp. 149-160.
21. R. Patel, V. Kumar, V. Tyagi and V. Asthana. "A Fast and Improved Image Compression Technique Using Huffman Coding". In 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), 2016, pp. 2283-2286.
22. S. H. Kim, J. Jeong and J. Lee. "Selective memory deduplication for cost efficiency in mobile smart devices". *IEEE Transactions on Consumer Electronics,* vol. 60, pp. 276-284, 2014.
23. Y. Tian, K. Zhang, P. Wang, Y. Zhang and J. Yang. Add "Salt" MD5 algorithm's FPGA implementation. *Procedia Computer Science*, vol. 131, pp. 255-260, 2018.
24. D. Pamula and A. Ziebinski. "Hardware Implementation of the MD5 Algorithm". 9th IFAC Workshop on Programmable Devices and Embedded Systems, Roznov pod Radhostem, Czech Republic, Feb. 2009.

Author Queries???
AQ1:Kindly cite figure 7 and 11 in the text part