PHOTOQR: A NOVEL ID CARD WITH AN ENCODED VIEW

Livio Tenze and Enrique Canessa

The Abdus Salam International Center for Theoretical Physics, Trieste, Italy

ABSTRACT

There is an increasing interest in developing techniques to identify and assess data to allow an easy and continuous access to resources, services or places that require thorough ID control. Usually, in order to give access to these resources, different kinds of documents are mandatory. In order to avoid forgeries without the need of extra credentials, a new system –named photoQR, is here proposed. This system is based on a ID card having two objects: one person's picture (pre-processed via blur and/or swirl techniques) and one QR code containing embedded data related to the picture. The idea is that the picture and the QR code can assess each other by a proper hash value in the QR. The QR without the picture cannot be assessed and vice versa. An open source prototype of the photoQR system has been implemented in Python and can be used both in offline and real-time environments, which effectively combines security concepts and image processing algorithms to obtain data assessment.

KEYWORDS

Quick Response (QR) code, data assessment, authentication, ID-card, Cybersecurity.

1. INTRODUCTION

There are many social, educational and workplace activities that require some sort of fast control identification (ID) for thousand attendees or spectators to massive events, hundreds of participants and scholars to conferences or dozens workers in factories. Usually, such IDs control consists of at least two steps. One step is for generating and issuing an actual card (*e.g.*, printed or plastic) containing some biometric details of a person archived, as for example, in a QR (Quick Response) 2D barcode. The second step is the verification by scanning of the identity contained in the QR to be presented upon request by the individual concerned. This last step can delay considerably the process of QR ID's controls when avoiding misuse and fraud. One example of such cumbersome complications was realized during the recent SARS-CoV-19 pandemic where the European Green Pass was adopted in shopping centers, bars, stadiums, universities and many other places.

There is an increasing interest in searching new systems to improve the data security while ensuring the privacy of ID cards. Moreover, QR identifiers are more and more used to access services or to certify something. Often it is not so simple to correlate the identity and the generated QR info. In this work we aim to develop a solution to this last problem by merging information inspired by the saying *"a picture is worth a thousand words".* In the present case, we believe "*an encoded picture can be worth a thousand doubts"*.

QR code can be used to store a limited size of data, hence it is very difficult to embed large sets of data including embedded small and compressed images. Moreover, if the density of bytes in the QR code is increased, the detection process with webcams or smartphones becomes very

difficult due to the limited resolution of the camera and the ability to both discriminate and read embedded data by using image processing algorithms. Therefore, another approach has to be considered in order to cope with these difficulties.

Our goal is to propose a new identification system (not only necessarily related to person identification) capable to assess QR data with a person's or object's photo to which they belong. The idea is to have a method providing a small compressed string of bytes (to be stored in the QR code) which can be used to assess –within a reasonable confidence interval, a picture associated to the QR code (pre-processed via blur and/or swirl techniques). To achieve this goal we use the hashing image processing technique mainly to detect duplicates of images [1].

The proposed algorithm, named photoQR, can be used as a simple and effective security assessment system. In fact, the identification relies on an ID badge where an encoded picture and a QR code are present. Both the encoded person's image and some of his/her biodata are embedded in the QR code, and these are strictly related. All data embedded in the QR are generated from a unique image such that the person's picture cannot be changed without changing the QR code and vice versa. The encryption algorithm to store the data in the QR can ensure that nobody can change the QR without knowing the encryption keys. As discussed in this work, the hope is that this photoQR approach will help to avoid forgeries.

2. BACKGROUND

Quick Response (QR) code is one of the most popular types of black/white 2D barcodes with a wide range of applications. QR code images are used for example to redirect access to websites, to send SMS messages, to ring up a phone number, to display brief text among other encoded information. Since few years, authenticated QR code with personal or student ID details are being used in universities, libraries and many other academic institutions specially in developing countries [2]. As illustrated in the upper part of the scheme below, the QR authentication of an ID card is usually a two step process: First, one extracts via smartphone scanning the embedded personal details generated during a registration QR process (which sometimes its redirects to a web-page for data authentication). This data is extracted from the horizontal and vertical patterns in the QR image. Second, one then verifies the extracted data against another ID document with a recent person's photograph. This second process can be time consuming in events or activities organized with a large participation of public, and does not exclude the possibility of misuse and fraud.

During the Covid-19 pandemic, many countries adopted individual's QR codes for the issuance of digitally signed certificate of vaccination or of Antigen and PCR tests (the so-called EU Green Pass), which was validated by simple smartphone scanning or an ad-hoc reader machine with a ringtone upon acceptance. The big limit of this approach was the proliferation of fake QR vaccination certificates on the web as issued in many countries worldwide, even with an high degree of professionalism, showcasing forged valid certificates [3]. Another problem found with this approach was due to confusion with the expiration dates in the QR codes issued for contact tracing registering. During the outbreak some people were getting frustrated when some QR codes worked and others did not because of the lack of a common database among providers [4]. Sharing fraudulent URLs via touch-free malicious QR codes (or Quishing) also proliferated with the Covid-19 pandemic. Simple QR codes were largely used for a fast and indirect access to many useful services because of the reorganization of our lifestyles [5].

QR codes are an active area of development where security, encryption, dimension and extra image effects are incorporated to further personalize the layouts and make them more reliable [6]. As emphasized in Ref. [4], to trust on the tools used such as QR coding is one thing we need the

most for an effective response during (future) pandemic. In the following we describe our photoQR algorithm. As depicted in the scheme, the main feature of photoQR is that this is a one step robust process. It can encode in one image a person's photo, personal ID data, and a unique hash string for automated control. The verification by standard scanning is fast and secure in the presence of a person –owner of the assigned photoQR identity card.

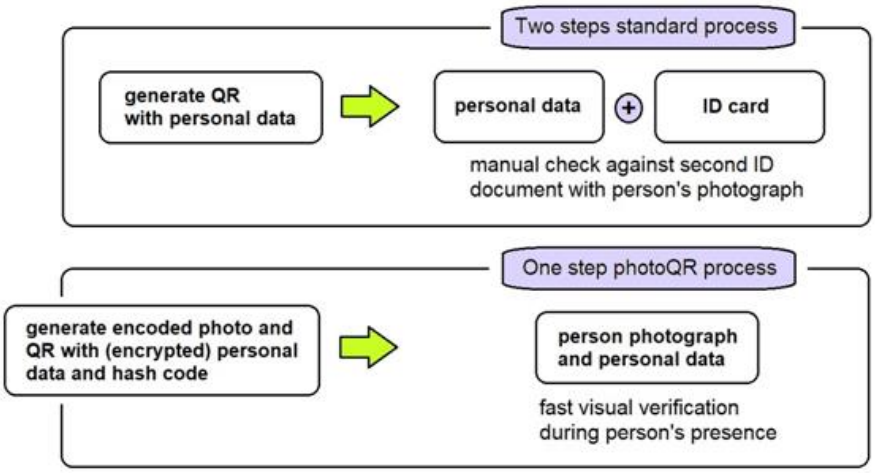

Figure 1. Classical QR code use and proposed photoQR system comparison in brief.

3. PHOTOQR DESCRIPTION

The system consists, as illustrated in Figure 1, of a simple badge composed by an image and a QR code. The main problem is to find a way to relate an encoded image and the QR. If the QR and the image are strictly related and this relation can be checked, one could immediately assess that both data (QR and undecoded image) are original and not forged.

In order to correlate an encoded image and a biometric QR code, we use an image processing technique known as *image hashing* [1]. Image hashing is usually used as a comparison method among images and to find duplicated pictures. Usually the hashing functions in computer science are used in fast searching of data inside a database: the addition and search of objects can be speed up by using hashing. When these functions are used in database indexing, they are characterized by the property of randomization of inputs. In other words two similar input words should provide completely different hash output. This property ensures a uniform fill of the database.

Conversely, in order to use the hashing function in image comparison, the operator requires the opposite property: similar images provide very similar hash outputs. The hashing function comparison is a very fast method to compare images and understand when they are similar or equal. The idea is to use the hashing comparison to tie the image and the QR in the badge. We embed the hash of the image in the badge inside the QR code.

In order to assess the badge data, the system get the image and evaluates the hash. Then it reads info inside the QR, where the original image hash is embedded, and compare the hash calculated from the image with the one inside the QR. If the embedded data match the evaluated hash, the card QR and the image are properly related and not forged. In order to improve the safety of the data, an encryption algorithm can be introduced to avoid falsification of data. Another additional improvement of this approach is the possibility to warp the badge image in order to make it

difficult to identify the person or object in the image. In this work, this is done by implementing two different algorithms in order to warp (and un-warp) the image: image blurring [7] and swirl [8].

The image blurring system blurs the image and provides an hardly identifiable image. According to the parameters used to generate the blurred image, it is possible to reconstruct the original image. The widely used technique of deconvolution [9,10] can be used to restore the original image from a blurred one. The image provided by the deconvolution is a good approximation of the original picture, but this procedure is lossy and the perfect reconstruction of the original image cannot be obtained.

Another implemented warping method in photoQR is the swirl. The algorithm adds swirl from the center of the image. The swirl, similarly to the blur algorithm, is lossy and the reverse procedure cannot provide a perfect reconstruction of the original image. Notwithstanding the function can be inversely applied in order the get a good approximation of the original image.

As previously mentioned, the QR code part can in principle be forged, but this can be avoided by adding an encryption algorithm. The current prototype implemented an encryption based on the GPG non symmetric keys system. The encryption system relies on a keys pair, the private key and the public one. The data embedded in the QR code are encrypted applying the private key. Everyone can decrypt the QR data with the public key, but only some external "certification authority" can create the encrypted QR with the private key.

4. PHOTOQR ALGORITHM @ WORK

The current implementation has been developed in Python3 to speed-up the development and to have an easy way to debug the source code. The main libraries used in this implementation are the following ones:

- openCV (with numpy and matplotlib for Python implementation): used to deal with all images and to process at low level the picture data and to compensate perspective;
- imagehash: to compare images and have a robust similarity metric;
- qrcode: to create QR codes;
- pyzbar: to detect QR code and the box in a robust way;
- crypto: to manage the encryption process (create key pair, encrypt, decrypt);
- skyimage: to use a swirl effect to make the image picture hardly "understandable".

Other pieces of code have been developed, integrated and improved in order to apply blurring and deblurring, by using the Wiener approach in the Fourier domain. This implementation provide fast results with low computational cost. An example output with blurring is illustrated in Figure 2. All these libraries and codes can be easily ported in a real-time implementation with C code.

There are two main modes of operation of the system: *offline mode* and *real-time mode*. In the offline and real-time mode the user can call the Python code in order to:

- create the image containing the QR code and the person picture (which can properly scrambled to make it not easily recognizable);
- detect the previously created image by un-warping the picture, decrypting and reading the info stored in the QR code and comparing the hash with the hashing of the picture.

In the offline mode the user provides an image to create the output card or provides the image to be treated in the the detection mode. In the real-time mode the algorithms are applied with a real-time video acquisition and detection.

Figure 2: photoQR ID card produced by the creation step of the algorithm with image blurring option enabled. The right part of the card contains the QR with data produced by the image hashing function of the left picture.

The user can set some properties to change the behavior of the algorithms: for example swirl, blurring and encryption can be enabled or disabled. The settings can be summarizes as follows:

- size of the output card;
- hash function to be used;
- threshold for hashing comparison;
- region size to be blurred or warped;
- margin size around the picture;
- enable/disable blurring (the user can select the blurring mask and the strength);
- enable/disable swirl;
- enable/disable data encryption.

The procedure applied in the creation and detection steps are summarized in the following sections.

4.1. PhotoQR Creation

In this step the user creates a photoQR ID card which will be checked in the detection step below.

- An image is provided (in real-time or from a static picture);
- after image check, the image is cut and resized according to the user's settings;
- if blurring or swirl are set, these functions are applied to "warp" the original image. An example output with swirl is illustrated in Figure 3;
- the hashing function is applied to the image and stored;

- if the encryption function is enabled, the hash value is encrypted and stored, otherwise the original hash value is used;
- from the hash value at the previous step, a QR image is generated;
- the image and the QR code are embedded in a single ID card and save to disk (default output file: Reference.png)

Figure 3: photQR ID card produced with the swirl option enabled.

4.2. PhotoQR Detection

In the detection (or verification) step the inverse process of creation takes place. The offline detection is more easy than the one in real-time due to the noise caused by the camera acquisition. Some processing steps are added to make the real-time process robust.

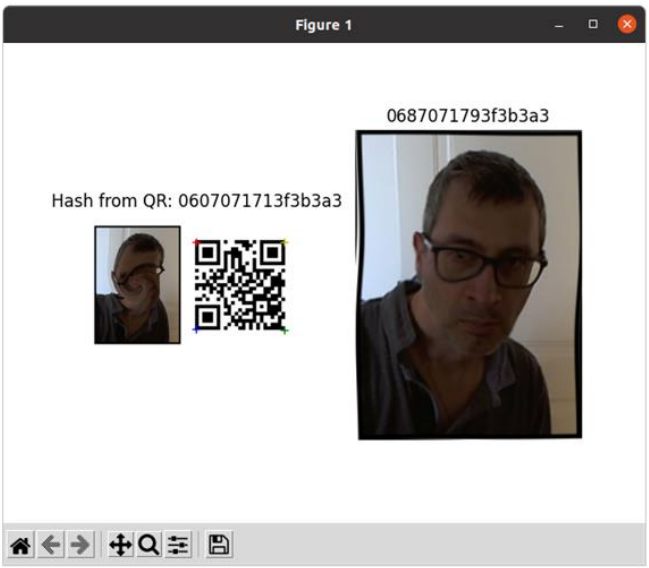

Figure 4: Detection step. The right region contains the reconstructed image (the swirl has been reversed). The hashing of the image is very similar to the hashing value stored in the QR, so the picture and the QR are assessed.

In Figure 4 the detection step is illustrated. It consists of

- image acquisition or reading from file;
- detection of QR code and identification of the surrounding box (with corners references);
- according the the position of QR corners, the perspective compensation is applied to create a flat image without deformations;
- reading of the QR code: according to the user's settings (*i.e.,* decryption is applied if required), the QR data area is read. The hash value at creation is extracted;
- the system, starting from the position and the box of the QR code, detects and extracts the associated picture;
- the hashing function is applied to the extracted picture and stored, to be compared to the hash value embedded in the QR code;
- comparison between hash from picture and hash from the QR code part: if the weighted difference is less that a threshold (default: 10), identification is ok.
- Final step, the system un-warps the picture in order to make the image recognizable.

In Figure 4 a detection from a photoQR card created with swirl property enabled is shown. In the left side of the Figure the input card can be seen. On the right, the un-wapped picture is illustrated. As mentioned in the description, inverse swirl process is lossy (since the swirl operation is non-linear), so the result is a good approximation of the original image. It is not possible to obtain a perfect recontruction of the original picture.

5. DISCUSSION

The latest open source software development and version control of the proposed photoQR system for Linux O.S. can be downloaded from GitHub: https://github.com/canessae/photoQR. In order to satisfy all requirements to use the prototype, the user has to run the following command to download all dependencies (see Section 4). The user needs to have Python3 and git already installed to download and run the prototype application:

```
user:~$ git clone https://github.com/canessae/photoQR.git
user:~$ cd photoQR/python
```

This command downloads the Python source code. In order to have a clean installation, it is suggested to use a Python virtual environment:

```
user:~$ python3 -m venv venv
user:~$ source venv/bin/activate
```

Now it is necessary to fulfill all requirements which can be simply downloaded with the following command:

```
user:~$ pip3 install -r requirements.txt
```

All needed dependencies of the prototype are stored in the requirements.txt file, and the prototype is ready to be used. The user can create a new photoQR ID card running the command:

```
user:~$ python3 testPhotoQR.py create
```

This command tries to open the camera and wait for a button to take a photo which will be used for the card creation. If the user specifies an image filename after the "create" command, the photoQR ID card is generated using that picture. The detection algorithm can be used, as for the creation, using the camera or a specified image file.

user:~$ python3 testPhotoQR.py detect <*Figure_1.png*>

If nothing is specified the code tries to open the camera and wait for a button press in order to take a photo and to perform the card assessment. If an image filename is specified after the "detect" command, that image is used for assessment (*Figure_1.png* in the example above). The photoQR algorithm has been developed while taking into account the robustness in a real environment. The perspective compensation has been added to make more robust the provided solution. However if the card in too much wrapped or it is bent, the system can fail during the detection step. A possible workaround is to print the QR code with the picture on a plastic plate, in order to avoid detection errors –as done with other standard QR ID cards. In Figures 5 and 6 below, the complete flow of the proposed system and the most important steps of the schema are summarized.

In Figure 5, the card verification schema is shown. There are two inputs, the person's or object's image, which can be acquired by the camera or loaded from a file, and some extra data. The image size can be setup by the user as mentioned in Section **Error! Reference source not found.**: this size has to be same in both creation and verification phase. The extra data input can be embedded in the final QR in order to provide details about the assessment procedure. There is not a hard limit to the extra data size, but the size of the embedded data increases the QR density. As mentioned, if the QR density is too high, the algorithm for QR code detection can fail and couldn't be able to detect and decode the QR image.

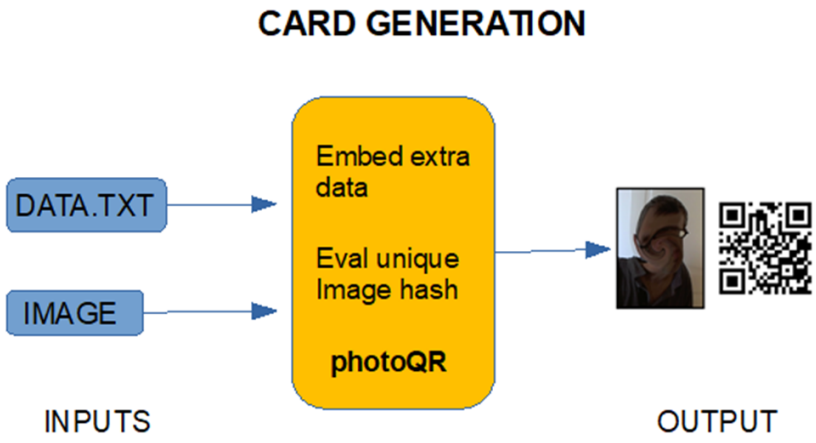

Figure 5: photoQR ID card creation flowchart.

The inverse process of card verification is summarized in Figure 6. This process involves many image processing algorithms to provide the robustness to the whole system. The photoQR ID card created in the creation step is the input and has to be divided in order to separate the picture and the QR code. The more the boxes surrounding the objects are accurate, the more the following process gives good results. The extracted picture is fed to the hashing function and could need unwrapping or deblurring in order to reconstruct the original image. In the other branch the QR is decrypted (see Section **Error! Reference source not found.**), and all embedded data are extracted: the hash value, evaluated in the creation phase, is obtained in the process.

The core of the proposed system is the comparison step where the hash from the QR and the hash evaluated from the person's picture are taken into account. If the distance from two hashes is low, the verification is ok, otherwise the verification fails and the card could be forged. We note that both the creation and the verification step is not crucial for processing time. The whole process is not time-consuming and it can be implemented in a real-time environment.

There are many hashing functions which can be used for image comparison: Average hashing, Perceptual hashing, Difference hashing, Wavelet hashing and so on (see [1]). In the proposed Python prototype various of these algorithms have been tested. We have found that Perceptual hashing shows better results than others. It is possible to change the hashing function with a limited effort. Another aspect related to the hashing function to mention is the distance operator: the Hamming distance [11] is a good choice to compare hashes and obtain a numeric value for the distance between codes.

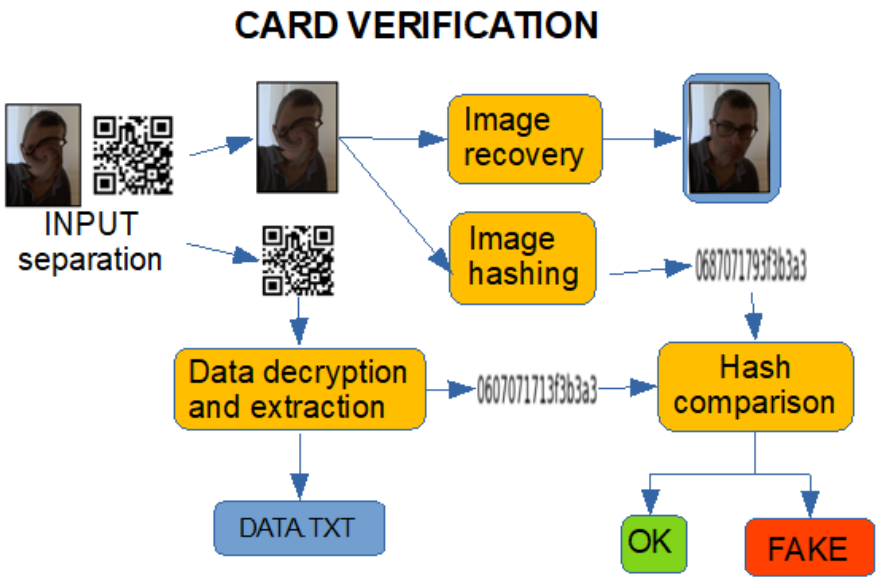

Figure 6: photoQR ID card verification flowchart: A fake warning can be given upon hash comparision.

The current system works well, but the processing in the real-time mode is a critical step and it can be affected by the acquisition quality. The picture detection step can be affected by noise, lighting and by image deformation. So the algorithm can be improved in order to make it more robust. Another critical step is the warping function. Both blur and swirl are lossy procedures so the inverse process does not provide perfect reconstruction. Both of them are affected by the detection of the box surrounding the picture. Improving the box picture detection will provide better results.

The picture position and the surrounding picture has been setup to make simple the image box detection. In order to improve the accuracy of the picture box a particular border can be introduced: a particular border composed by black and white squared boxes to surround the image. The texture of the border could simplify the image detection accuracy with an algorithm similar to that used for the usual QR detection.

6. CONCLUSIONS

We have proposed a novel photoQR ID card system with an encoded person's picture (processed via blur and/or swirl procedures) capable to assess QR data related to a person or object and to a small compressed hash string of bytes. The latter is used to assess the real person's physiognomy from fast visual inspection. To our best knowledge, no other technique of this kind has been proposed. The security of photoQR is relatively high. The encrypted hash value of the (hard-to-invert) warped image is compared with the stored hash value in the QR part. If there is consistency (within a reasonable confidence interval), the verification is passed, otherwise the verification fails with a proper signal warning. In the dynamic QR code production, bio-metric information can be easily encrypted by applying a private key. Having this added layer of protection, also helps to avoid any Cybersecurity mishaps. A photoQR ID card without the person's picture (or no other image) cannot be assessed and vice versa.

In most recent developments of the prototype (code available at github.com/canessae/photoQR) the use of the SIFT/SURF features has been introduced in the source code. Such transforms are used to identify and detect robust features inside pictures. These features are scale invariant and robust toward image rotation and these could be used instead of, or along with, the currently used hashing functions.

One of the most important development of the present system will be the porting of the photoQR algorithm to mobile environments. This step should not be so difficult due to the (smart) choice of the involved libraries. It could also be interesting to investigate new algorithms for image hashing, taking into account neural network approaches. Automatic face recognition systems could be implemented to compare in real-time the person's facial features or expression with the un-warped person's image used as input in photoQR. We hope to motivate a trend toward this approach to bring security of ID cards or Green Pass Certificates, having QR codes, to a higher level.

Data Availability

The latest code development and version control of the proposed photoQR system for Linux O.S. can be downloaded from GitHub repository at https://github.com/canessae/photoQR

REFERENCES

[1] Buchner J., "An image hashing library written in Python", https://github.com/JohannesBuchner/imagehash (last visited April 2024)

[2] Sanaul Haque Md., Dybowski R., "Advanced QR code based identity card: a new era for generating student ID card in developing countries", SIMS '14: Proc. 2014 First Int. Conf. Sys. Informatics, Modelling and Simulation (2014) 97–103. doi: 10.5555/2681970.2682472

[3] Georgoulias D., Pedersen J.M., Falch M., Vasilomanolakis E., "Covid-19 vaccination certificates in the darkweb", ACM Digital Threats: Research and Practice (April 2022) doi: 10.1145/3530877

[4] Tzer-Yeu Chen A., "How fragmentation can undermine the public health response to Covid-19", ACM Interactions 28 (2021) 64-69. doi: 10.1145/3448413

[5] Sharevski F., Devine A., Pieroni E., Jachim P., "Gone quishing: a field study of phishing with malicious QR", Preprint (2022) https://arxiv.org/abs/2204.04086

[6] Xu M., Li Q., Niu J., Su H., Liu X., Weiwei Xu et al. "ART-UP: a novel method for generating scanning-robust aesthetic QR codes", ACM Trans. Multimedia Comput. Commun. Appl. 17 (2021) Article 25. doi: 10.1145/3418214

[7] Pratt W.K., "Digital Image Processing", PIKS Scientific Inside, Fourth Edition. Print ISBN: 9780471767770 doi: 10.1002/0470097434

[8] scikit-image: "open source collection of algorithms for image processing", https://scikit-image.org/ (last visited April 2024).

[9] Jain A.K., "Fundamentals of Digital Image Processing", Ed. Pearson; 1st edition (October 3, 1988). ISBN-13: 978-0133361650

[10] Carrato S., Ramponi G., Marsi S., Jerian M., Tenze L., "FPGA implementation of the Lucy-Richardson algorithm for fast space-variant image deconvolution", 2015 9th International Symposium on Image and Signal Processing and Analysis (ISPA). doi: 10.1109/ISPA.2015.7306047

[11] Perceptual hash algorithms: "Getting Funky With pHash", https://hackerfactor.com/blog/index.php%3F/archives/432-Looks-Like-It.html (last visited April 2024).

AUTHOR

Livio Tenze (Ph.D in Electrical Engineering) and **Enrique Canessa** (Ph.D in Physics) are with the Science Dissemination Unit (SDU) of the ICTP Abdus Salam International Centre for Theoretical Physics in Trieste, Italy.