IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 1 November 2023, accepted 20 November 2023, date of publication 27 November 2023,
date of current version 30 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3337033

RESEARCH ARTICLE

Toward an Efficient Iris Recognition System on Embedded Devices

DANIEL P. BENALCAZAR[1], (Member, IEEE), JUAN E. TAPIA[1,2], (Member, IEEE), MAURICIO VASQUEZ[1], LEONARDO CAUSA[1], ENRIQUE LÓPEZ DROGUETT[3,4], AND CHRISTOPH BUSCH[2], (Senior Member, IEEE)

[1]Department of Mechanical Engineering, Universidad de Chile, Santiago 8370456, Chile
[2]da/sec-Biometrics and Internet Security Research Group, Hochschule Darmstadt, 64295 Darmstadt, Germany
[3]Department of Civil and Environmental Engineering, University of California at Los Angeles, Los Angeles, CA 90095, USA
[4]Garrick Institute for the Risk Sciences, University of California at Los Angeles, Los Angeles, CA 90095, USA

Corresponding author: Juan E. Tapia (juan.tapia-farias@h-da.de)

This work was supported in part by Agencia Nacional de Investigacion y Desarrollo (ANID) under Grant FONDEF IDEA ID19I10118; in part by the European Union's Horizon 2020 Research and Innovation Program under Grant 883356; in part by the German Federal Ministry of Education and Research; and in part by the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity German National Research Center for Applied Cybersecurity (ATHENE).

ABSTRACT Iris Recognition (IR) is one of the market's most reliable and accurate biometric systems. Today, it is challenging to build NearInfraRed (NIR) capturing devices under the premise of hardware price reduction. Commercial NIR sensors are protected from modification. The process of building a new device is not trivial because it is required to start from scratch with the process of capturing images with quality, calibrating operational distances, and building lightweight software such as eyes/iris detectors and segmentation sub-systems. In light of such challenges, this work aims to develop and implement iris recognition software in an embedding system and calibrate NIR in a contactless binocular setup. We evaluate and contrast speed versus performance obtained with two embedded computers and infrared cameras. Further, a lightweight segmenter sub-system called ''Unet_xxs'' is proposed, which can be used for iris semantic segmentation under restricted memory resources. The evaluations reveal that Unet_xxs reduces the number of parameters by 77% and duplicates the speed of state-of-the-art segmentation models with an EER drop smaller than 1% in Iris Recognition with 8.06 frame per second (fps) and Intersection Over Union (IOU) of 0.8382.

INDEX TERMS Embedding systems, iris sensor, NIR camera, hardware.

I. INTRODUCTION

Iris Recognition (IR) literature has several well-documented and evaluated approaches on how to process iris images, extract the iris code, and perform comparisons [1], [2], [3], [4], [5], [6], [7]. Some of the proposed approaches used classical methods of image processing, such as Gabor Wavelets, [1], [2], [3], Localized Binary Patterns (LBP) [4], or Binarized Statistical Image Features (BSIF) [5]. In contrast, others used Convolutional Neural Networks (CNN) for segmentation and/or encoding [7], [8], [9], [10], [11]. However, only a few previous works aimed to describe how to build a binocular iris imaging device efficiently from scratch at both hardware and software levels. As we mentioned before, today, we can use iris-captured devices not only for identification. Instead, new approaches have been proposed using a capture device as a screener filter for an occupational test to measure the worker's level of alert. Moreover, a binocular device can be used as a Fitness for Duty system from Near Infrared (NIR) iris images.

The availability of Raspberry-PI and Jetson-Nano increased the number of devices developed due to low prices and fast adoption compared to regular binocular NIR capture devices. It is essential to highlight that all hardware components in this work were developed according

The associate editor coordinating the review of this manuscript and approving it for publication was Carmelo Militello.

to the availability in our region (Latin America) during the pandemic season.

When an eye detection and segmentation subsystem are deployed on a hardware platform, it requires sufficient amounts of memory to store its parameters (weights and biases), and intermediate computational results exchanged between its deployments. This is not a trivial task.

In this work, an iris recognition system has been developed from scratch considering the increase of computational power, utilising semantic segmentation, and designing a binocular capture device that makes use of the information of both eyes. The computational power is efficiently allocated by using and comparing two modern embedded boards, the Raspberry-Pi-4B and the Jetson-Nano. Additionally, we developed a low-weight semantic segmentation network that can work on those boards with fast inference times. Finally, the proposed device arrangement can capture a face image from which both eyes are extracted with good resolution for IR. Therefore, the proposed device could also be used for face recognition as a complementary function.

The contributions of this work are the following:

- *IR Hardware Design*: the main elements needed to build a NIR iris imaging device with available components are described.
- A comparison between Raspberry-Pi and Jetson-Nano is reported.
- IR Pipeline: an end-to-end IR pipeline is implemented, which acquires information about the two eyes in a face image.
- *Iris Segmentation Network*: a lightweight iris semantic segmentation network is proposed, which finds eyes and segments the iris.
- *Efficiency*: The proposed architecture of UNet_xxs reduces the parameter number by 77% with respect to State of the art (CCNet). This makes UNet_xxs faster and easier to implement than previous works [11]. Additionally, joining available hardware and software components into a single cohesive system with a practical use case is another innovative point.
- *System Calibration and Evaluation*: the optimal distance to the camera is calibrated; additionally, the speed and performance of the proposed system are evaluated.

This paper is organised as follows: related work is described in Section II. The method is outlined in Section III. The databases and data-preprocessing are introduced in section IV, and the experiments and results are presented in section VI. Finally, conclusions are presented in section VIII.

II. RELATED WORKS

A. IRIS RECOGNITION HARDWARE

A small number of research works have detailed how to build IR capture devices from the hardware perspective. Early works focused on the implementation of IR systems on Digital Signal Processors (DSP) as well as Field Programmable Gate Arrays (FPGA) [12]. Those systems were attractive because they exploited parallel processing to reduce computational time. However, they are expensive in comparison to single-board computers.

Hentati et al. [13] implemented Osiris [3] on an FPGA device, reducing processing time and power consumption. Further on that, Avey et al. [14] reduced processing time by using a FPGAs. Processing time was reduced by a factor of 22 and 486 when compared with x86 and ARMv7 processors, respectively.

Grabowski and Napieralski [12] studied hardware efficiency for complex IR algorithms and remote camera sensors using Digital Signal Processors and FPGAs. Ma et al. [15] implemented IR Quasi-cyclic low-density parity-check (QC-LDPC) error correction with FPGA. Also, Grabowsky et al. [16] proposed an iris-on-the-move recognition system which does not require special cooperation with the user during biometric sample acquisition. This specialised architecture is mainly composed of Digital Signal Processors (DSP) and Field-Programmable Gate Arrays (FPGA). The main components include a Xilinx Spartan 3AN FPGA and four digital signal processors: two of which are fixed-point (CPU0 and CPU1) and two of which are floating-point (FPU0 and FPU1). The board is inserted into one of the PCI 32-bit slots on the ML510 platform.

Chou et al. [17] proposed a Non-orthogonal Iris recognition system based on four spectral (red, green, blue and Near-InfraRed) images. They proposed a circle rectification method to reduce the off-axis iris distortion. The images are captured at different off-axis angles using a Canon VF 75-mm lens and a Sony HVL-IRH2 camera. All the metrics are estimated on a PC running Windows XP with an Intel Core 2 Duo 4400 processor and 1 GB RAM.

Single-board computers such as Raspberry-Pi offer a small form-factor platform for IR systems. Cruz et al. [18] implemented Daugman's equations on a Raspberry-Pi-2B.

Kunik et al. [19] used the Raspberry-Pi-3B to implement Iris Recognition algorithms. Their method was based on USIT and OpenCV. Boutros et al. [20], [21] implemented IR on Head-Mounted Displays and proposed EyeMMS for segmentation with good results. Zhang et al. [22] created the CASIA-Iris-Mobile-V1.0 database to test IR performance on mobile devices. They also presented a feature fusion mechanism to increase IR performance.

Bastias et al. [23] reported a similar device, but for the purpose of 3D iris scanning. They used a Raspberry-Pi-3B+ with an infrared 8Mpx camera and NIR LEDs. The setup was mounted on a proposed Virtual Reality (VR) headset.

Fang et al. [24] proposed an end-to-end open-source system for IR with Presentation Attack Detection. They described how to build an iris capture device using general-purpose electronic components for less than 75 USD. Their system utilises a Raspberry-Pi-3B+, an infrared 5Mpx camera, a NIR optical filter, two NIR LEDs, and control circuits mounted on a breadboard. Fang et al. also produced a complete IR solution with Presentation attack detection (PAD) that runs on the restricted computational power of the Raspberry-Pi-3B+.

In the other hand, there have also been some implementations on smartphones. Raja et al. [25], and Tapia et al. [26] utilised an iPhone 5S and a Nokia Lumia 1020 as the capture device using Visible Spectrum (VS) imaging and super-resolution. The smartphone also produced segmentation inference and recognition.

Benalcazar et al. [27], [28] proposed an iris imaging device using VS images, which was used for IR in 3D [29], [30]. This device was based on a frame similar to Bastias's VR goggles; however, it was built on opaque black acrylic to suppress undesired reflections. The acquisition device was a Samsung S6 Smartphone with a macro lens, capturing close-range images of the iris at 16Mpx resolution.

Efficient iris recognition in Head-mounted embedded displays has also been proposed. Fadi et al. [20], [21] developed iris and periocular biometrics for head-mounted displays, including segmentation, recognition, and synthetic data generation focusing on Augmented Reality (AR) and Virtual Reality (VR) applications.

B. IRIS SEMANTIC SEGMENTATION

Recent works in IR use deep learning to segment and localise the pupil and the iris in a periocular image [11], [31], [32], [33]. The Criss-Cross Attention Network (CCNet), developed by Mishra et al. [33], is an iris semantic segmentation network based on U-Net [32]. This lightweight network was trained by Fang et al. [24] to predict a binary mask of the iris from NIR images. They utilised the Hough transform to localise the pupil and the iris from the binary mask. CCNet was explored and re-trained by Tapia et al. [11] to work with highly dilated images under the influence of alcohol. They also developed two faster localisation algorithms that had better performance than the Hough transform: Least Means Squares (LMS) and Mixed [11].

Tapia et al. also developed DenseNet10 [11], a reduced version of DenseNet101 [34] that can semantically segment the pupil, the iris, the sclera, and the background.[1] This network obtained greater performance than CCNet on alcohol images at the expense of doubling the number of parameters. The centre of mass was used to localise the ellipses of the pupil and the iris. A performance comparison of these methods in terms of eye detection, iris segmentation and iris recognition is presented in Section VI.

CCNet and DenseNet10 are the main networks used in this work. However, there is abundant literature on semantic segmentation networks. For instance, Wu and Zhao [35] developed a segmentation network based on U-Net with good performance. Sip-SegNet [36] employs Convolutional Neural Networks along with adaptive fuzzy filtering techniques to segment the pupil iris and the sclera. Wang et al. [37] proposed a lightweight network for iris semantic segmentation. Osorio et al. [38], [39] developed a multi-class segmentation network for VS images. Li et al. [40] developed a segmentation network that works in uncooperative scenarios.

III. METHODS

A. DEVICE DESIGN

The Raspberry PI 4 and the Jetson-Nano are the most representative platforms for embedded systems available in the market. There are several single-board computers in the market with similar features and price ranges; however, the Raspberry-PI-4 and the Jetson-Nano are the most representative and easier to acquire.[2] The two boards also allow testing the performance difference for one system with and without CUDA cores.

The proposed iris-capturing device is designed to be portable, light and contactless, acquire NIR images of both irises, show relevant information to the user, and perform IR on its own. According to previous statements, the device is composed of the following modules: Processing unit, Cooling system, Capture Device, Illumination Printed Circuit Board (PCB), Display module, Power supply, and Case. All the modules are interconnected, as illustrated in the block diagram of Figure 1.

The processing unit is in charge of commanding the rest of the modules and running the IR software. We compare the Raspberry-Pi-4B and Nvidia's Jetson-Nano for the processing unit. Both boards are readily available, have a small form factor, and permit the control of peripheral devices.

The Raspberry-Pi-4B is based on a quad-core ARM-A72 64-bit CPU that runs at 1.5GHz. The model used in this work has 8GB of RAM and no CUDA-compatible GPU.

On the other hand, The Jetson-Nano is based on a quad-core ARM-A57 64-bit CPU that runs at 1.43GHz. It has 4GB of RAM and a 128-core Maxwell CUDA-compatible GPU. Therefore, the Raspberry-Pi has a better CPU and RAM, but the Jetson-Nano has a GPU that can run neural networks. We will analyse the trade-offs of each in terms of computational performance in Section VI.

The cooling system is different for the two choices of the processing unit. For the Raspberry-Pi-4, passive heatsinks were installed on the main chips, and a fan was installed in the case. The fan blows cool air to the heatsinks, and hot air exits through holes at the sides of the case. The fan is turned on and off automatically from the OS through the General Purpose Input Output (GPIO) pins and a current amplification transistor. On the other hand, a single heatsink, installed by default, cools the main components of the Jetson-Nano. A 4-pin fan was installed to cool the heatsink, and it is controlled by the OS with Pulse Width Modulation (PWM). Both options effectively reduce the excess heat from the device and prolong the lifespan of the device.

The display module is a seven-inch touchscreen for the operator's interaction with the capture system. While capturing the image from the eyes, a mirror supports the

[1] https://github.com/Choapinus/DenseNet10

[2] During the COVID-19 pandemic under logistic restrictions.

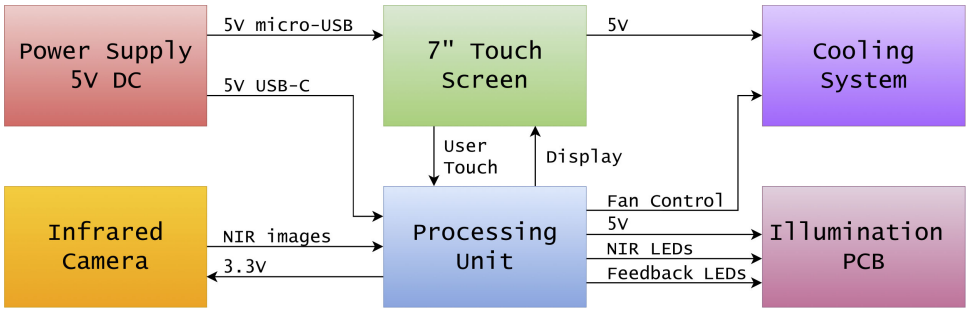


FIGURE 1. Block diagram of the hardware modules.

FIGURE 2. Image of the assembled device.

(a) Cam v1 5Mpx (b) Cam v2 8Mpx

FIGURE 3. NoIR camera modules for the raspberry-pi.

biometric attendant, showing the actions so they can centre the eyes of the capture subject inside the frame. The display is also used to provide information to the capture subject when the capture process has started and ended. Finally, it shows a welcome screen if the biometric attendant was identified or instructions to follow if otherwise. Touch information can be used to start the capture process or wake up the device if it is inactive for a long time.

Finally, the case holds all components together and facilitates cooling. A simple squared design was fabricated on laser-cut black acrylic. All the modules are secured to the acrylic frame by means of 3D-printed parts, nuts and screws. The bottom part of the device has a universal nut that can be mounted on tripods of any size. An image of the assembled device can be seen in Figure 2. It also illustrates that the camera input is a Face Region of Interest (ROI) with two eyes.

The setup of the capture device and illumination PCB was based on [24]. For the capture device, we compare the 5Mpx NoIR camera module v1 and the 8Mpx NoIR camera module v2 for Raspberry-Pi, as shown in Figure 3. The illumination PCB has two NIR LEDs to illuminate the eyes and two VL LEDs that give feedback to the user. The four LEDs are controlled by the GPIO pins of the processing unit by means of transistor-based current amplification. This enhances the brightness of the LEDs while avoiding drawing too much current from the GPIO.

The main differences with devices presented so far in the literature are the following: (i) It was designed and fabricated as custom PCB for the transistors that control the LEDs instead of using a breadboard; (ii) since PAD is not an objective for this work, both NIR LEDs are turned on and off at the same time instead of individually; (iii) we did not use a NIR optical filter since preliminary tests showed that the filter's image quality drops at night; and (iv) It was captured a face image and extract both eyes instead of capturing a single eye in close proximity.

B. SEMANTIC SEGMENTATION NETWORK

It was developed as a reduced version of U-Net [32], which is called UNet_xxs, for the purpose of iris semantic segmentation under limited computational power. Iteratively, we removed layers from CCNet's version of U-Net [24], [33] trained and tested performance until a small network with acceptable performance was obtained. Table 1 shows how reducing the input size, the number of layers, and kernel sizes produced architectures with fewer and fewer parameters. The parameter reduction process was stopped when the fps was 3.3 times larger than CCNet, but IOU was still above 0.8. This preliminary test was conducted with the 2,132 test images [11].

We trained CCNet and UNet_xxs on two tasks using the dataset from [11]. In the first task, the networks have to find the two eyes from the face-ROI image, as illustrated in Figure 6a. This is used to crop the two periocular images of the left and right eyes. The second task is to find the iris region in a periocular image, as depicted in Figure 6b.

TABLE 1. CCNet parameter reduction.

Network	Resolution	Parameters	fps	IOU
CCNet	320×240	122,514	2.40	0.9165
UNet_xs	160×120	65,202	6.11	0.8801
UNet_xxs	128×96	28,146	8.06	0.8382

The two tasks were trained separately and handled by two independent models both using the UNet_xxs architecture. The architecture of UNet_xxs is illustrated in Figure 4.

The segmentation and localisation pipeline is shown in Figure 5. It illustrates how the input Face ROI image (Figure 5a) is processed to find, crop, segment and localise the left and right irises. The output segmentation masks, as well as the localisation coordinates, are essential for IR.

First, a segmentation network (CCNet or UNet_xxs) is used to find the eyes. The network takes the Face ROI image (Figure 5a) and produces a segmentation mask where two big circles indicate the position of the eyes, as highlighted in red in Figure 5b. Then the distance d between the centroids of both circles are computed. Next, the bounding boxes of each eye are found (Figure 5c). These are rectangles placed at the centre of each circle with width $0.44 \times d$ and height $0.33 \times d$. This box size was determined empirically based on component and display sizes. After that; the bounding boxes are used to crop each eye and obtain the periocular images (Figure 5d).

Then, the segmentation networks take the periocular images as inputs and return the segmentation masks (Figure 5e), where the iris region is highlighted in white. Finally, the circles of the two pupils and irises are found (Figure 5f) using the LMS and the mixed algorithms proposed by Tapia et al. [11]. The results are the (x, y, r) triplets of each circle, where (x, y) are the coordinates of the circle's centre and r is the radius.

C. IRIS RECOGNITION

For IR, we used the method of Czajka et al. [5], implemented on the Raspberry-Pi by Fang et al. [24]. This method takes a periocular iris image, and the localisation (x, y, r) coordinates to produce a polar-coordinate normalised iris image, also known as rubber sheet [1], [41]. The rubber sheet has a size of $1 \times 64 \times 512$. A rubber sheet of the segmentation mask is also generated. Then, the BSIF-based ICA filters [5] are applied to encode the iris rubber sheet. The ICA filters are 7 filters of 15×15 and are available on open source by [24]. Finally, two encoded irises are compared using the Modified Hamming Distance, as described in Section V-A. The masks omit information from non-usable areas such as the pupils, eye leads and eyelashes from the comparison score computation.

IV. DATASETS

A. SEGMENTATION DATASETS

Both segmentation networks (CCNet and UNet_xxs) were trained for two separate tasks: finding eyes and segmenting the iris. To solve the first task using semantic segmentation, we used the face ROI images of [11], face images from the FRCG dataset, as well as our own captured images using the Raspberry-Pi. The ground-truth (GT) masks of each face and face ROI image were created by placing two big circles on each eye, as seen in Figure 6a. For this purpose, the pupil centres were marked on each image, and the pupil-to-pupil distance d was measured. Then, two circles with radius $0.2 \times d$ were placed at the pupil centres of each image. Additionally, we trained the network to produce the circles even though the eyes were closed, so we placed images of closed eyes in the training set. Finally, the network was trained to leave the image in black if no eyes were present on the input image. To accomplish this goal, we added natural images as backgrounds in the training set.

Figure 7 shows the number of images used for each source, a sample image and its GT segmentation mask. The number of images in Figure 7 accounts for offline data augmentations of rotation that were performed on each image. The total number of images was 37,645, and it was divided into 30,116 for training, 3,764 for validation and 3,765 for testing.

On the second task, the networks were trained to segment the iris region in a periocular image. Tapia et al.'s dataset [11] had GT of semantic segmentation for the pupil, the iris and the sclera. In this work, we isolated the iris label to train the networks only on the iris region. Additionally, we segmented the captured Raspberry-Pi images using the pre-trained DensNet10 [11] and used the outputs of that network as the ground truth masks.

Similarly to the first task, we trained the iris segmentation network to output a black image when no iris was present in the input image. For this purpose, we placed natural images, closed eyes and face close-ups in the dataset.

Figure 8 illustrates the number of images, an example and the corresponding segmentation mask per source. The number of images considers offline data augmentation of random rotations placed on all the sources. The total number of images is 58,599 and was divided into 44,748 images for training, 7,990 for validation and 5,861 for testing.

It is worth pointing out the fact that the first network can find and crop closed eyes while the second is not able to do so. This, in turn, helps count the number of times a person blinks as well as measure the blinking frequency when the networks perform continuous predictions on image sequences.

B. IRIS RECOGNITION DATASETS

The dataset of Tapia et al.'s [11] has a great number of face ROI images and iris images with segmentation ground truth, so it is ideal for training segmentation networks. However, it is not a good dataset for iris recognition since it has image sequences of around 75 frames per subject, and subjects might appear on alcohol and non-alcohol sets with different IDs since images were captured with different sensors. Therefore, we chose to use different datasets for IR tests.

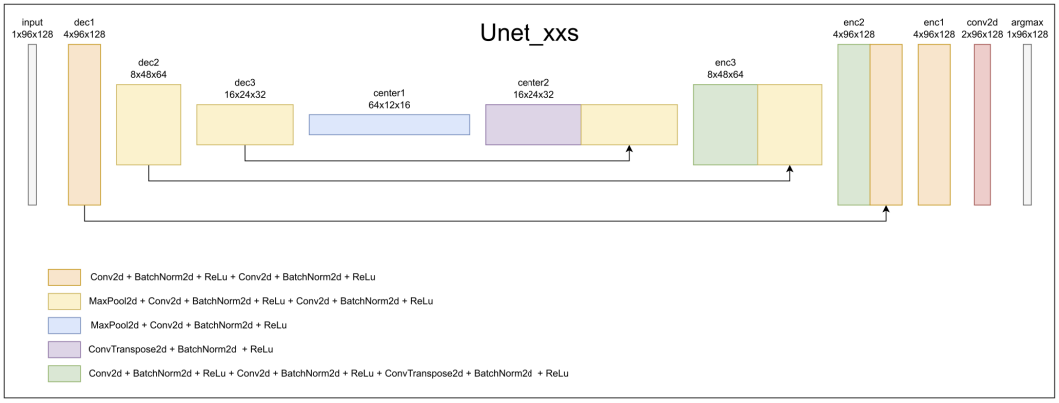


FIGURE 4. Architecture of UNet_xxs network.

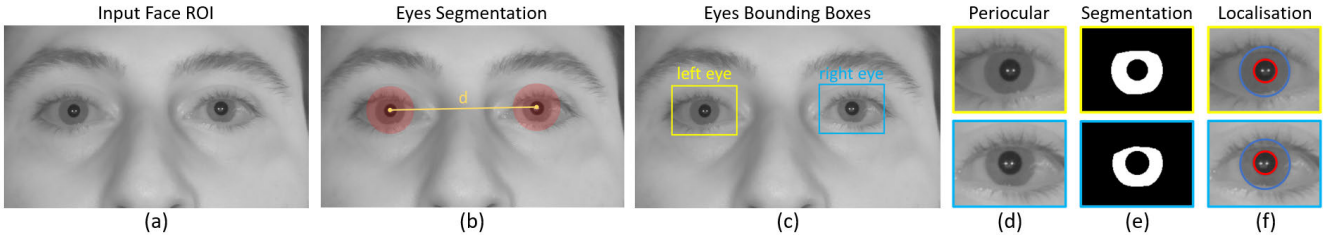


FIGURE 5. Iris segmentation and localisation pipeline.

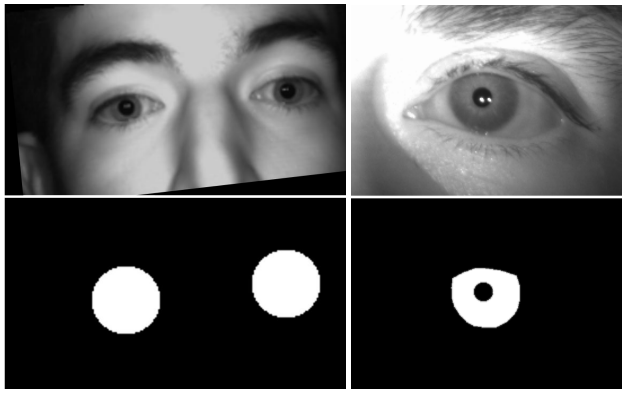


FIGURE 6. Illustration of the two trained tasks. Top: input image. Bottom: segmentation mask.

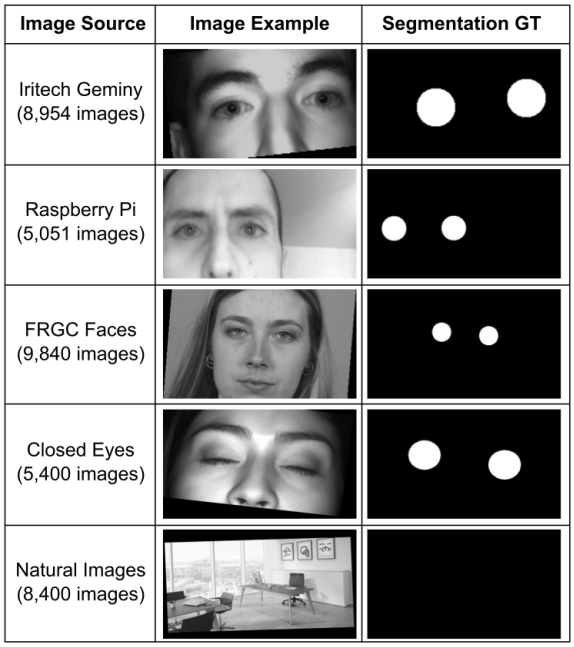


FIGURE 7. Find eyes dataset.

For fast and iterative IR tests, we used the dataset of Benalcazar et al. [27]. It has 96 subjects, and 5 NIR images of the right eye of each subject were captured, for a total of 480 images. Therefore, there are 960 mated and 114,000 non-mated comparisons in the dataset. The average image width is 388.4px, and the aspect ratio is 4:3.

For a final IR benchmark evaluation, we used the Notre Dame LG4000 dataset [42]. From the original dataset, we removed low-quality images and images with a predominant nose in the frame. The remainder of the 10,959 images were classified between the Left and Right eyes. We have different IDs on the left and right images of the same individual since the irises are different. In total, there were 811 IDs, which were partitioned in a person-disjoint manner, as 487 for training, 162 for testing, and 162 for validation. The partition corresponds to 6,663 train images, 2,090 test images and 2,203 validation images. The test set has 19,029 mated comparisons and 105,470 non-mated comparisons. We named this partitioned dataset as ND-LG400-LR and made the partition lists available on

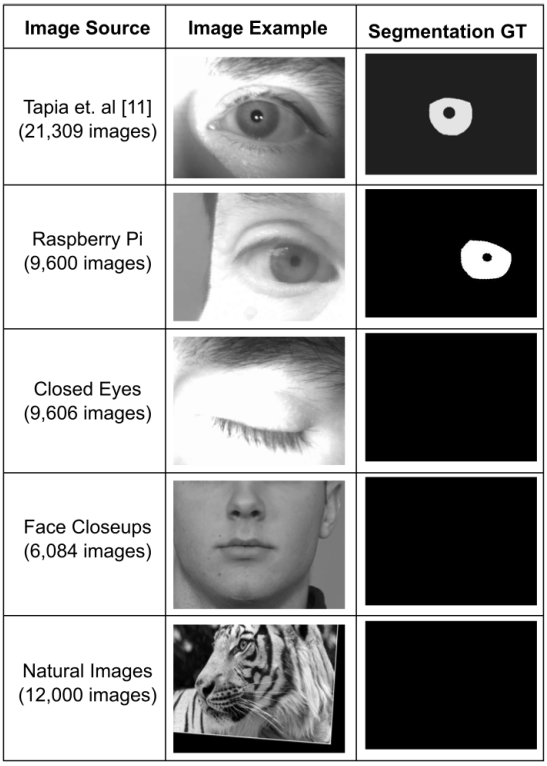


FIGURE 8. Segmentation dataset.

GitHub[3]. Additionally, we generated the segmentation GT by segmenting the 10,959 images with DenseNet10 and correcting mistakes by hand.

V. METRICS

A. MODIFIED HAMMING DISTANCE

For comparing two binary iris codes, one of the most used metrics is the Modified Hamming Distance (HD). The two encoded irises (*codeA* and *codeB*) and the rubber sheets of the segmentation masks (*maskA* and *maskB*) are used to compute the HD [1], as described in (1). The smaller the *HD* value, the more similar the two irises are.

$$HD = \frac{||(codeA \otimes codeB) \cap maskA \cap maskB||}{||maskA \cap maskB||} \tag{1}$$

B. D-PRIME SCORE

The d' score evaluates how separated the mated and non-mated comparison distributions are. It is computed score using equation (2) stemming from Daugman [1], in which μ_1 and μ_2 are the means of the mated and non-mated score distributions respectively, and σ_1 and σ_2 are the corresponding standard deviations. The greater the (d') value, the greater the separation between the two distributions.

$$d' = \frac{|\mu_1 - \mu_2|}{\sqrt{0.5 \times (\sigma_1^2 + \sigma_2^2)}} \tag{2}$$

[3]Upon acceptance

C. EQUAL ERROR RATE

The Equal Error Rate (EER) in a biometric test is the operating point at which the False Match Rate (FMR) and the False Non-Match Rate (FNMR) are equal. The smaller the EER value is, the lower the overall classification error of the system.

D. SIGNAL TO NOISE RATIO

The Signal to Noise Ratio (SNR) is the ratio between a signal's amplitude and the noise's standard deviation. In this work, we are interested in evaluating the SNR of iris images under different conditions. Therefore, we compute the SNR using (3), where the amplitude of the signal is the difference between the mean intensity values of the iris and the sclera, and the noise is considered as the standard deviation of the sclera. That is because the sclera is normally white; thus, any random variation of intensity from one pixel to the next is due to camera noise artefacts.

$$SNR = \frac{A}{\sigma} = \frac{|\mu_{iris} - \mu_{sclera}|}{\sigma_{sclera}} \tag{3}$$

E. INTERSECTION OVER UNION

The bitwise Intersection over Union (IoU) [11] metric is used to evaluate semantic segmentation accuracy. It compares the ground truth and predicted segmentation masks (M_A and M_B) by means of (4). This equation counts the number of bits that are high in a logical AND operation and divides it by the number of logical ones in the logical OR operation between the two masks.

$$IoU = \frac{\sum(M_A \wedge M_B)}{\sum(M_A \vee M_B) + \epsilon} \tag{4}$$

The range of values of the IoU is between 0 and 1, where 1 is a perfect match. A small positive value ϵ is added to the denominator to avoid division by zero.

VI. EXPERIMENTS AND RESULTS

Experiments 1, 2, 3 and 4 aim to resolve a crucial aspect in the fabrication of an iris imaging device, which is finding the optimal distance between the camera and the subject in terms of image resolution, SNR and available iris area. Since the v1 and v2 camera modules used in this work cannot automatically regulate the focal distance, the focus must be manually adjusted upon assembly. Therefore, the most crucial factor to find is the optimal distance between the subject and the camera at which to fix the focal plane.

Experiments 5, 6 and 7 evaluate the performance of the proposed segmentation network and pipeline in terms of accuracy and speed.

Finally, Experiment 8 evaluates the overall IR performance on the ND-LG400-LR dataset. It is important to highlight that all the software on both platforms, the Raspberry-Pi and the Jetson Nano, were implemented using Python 3.8, PyTorch 1.7 and OpenCV 4.5.

A. EXPERIMENT 1: IR PERFORMANCE VS IMAGE RESOLUTION

First, we explore the minimum resolution at which IR still has a reasonable performance. For this purpose, we use the dataset of [27], which is composed of 480 NIR images of the right eyes of 96 subjects. In this test, we varied the image width of each image from 388.4 px down to 25 px in 11 steps and measured the mean iris radius on each step. For each resolution, we also performed IR tests using the method of [5] and scored the EER, as well as the d' value.

Figure 9 and Figure 10 show the relationship between the iris radius and the IR metrics. The elbow in Figure 9 occurs at an iris radius of 9.2px; however, EER stabilises at 46.9px. On Figure 10, the d' also presents less rate of growth over 45px of iris radius.

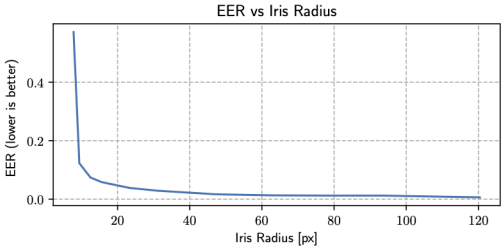


FIGURE 9. EER in IR decreases when iris radius increases.

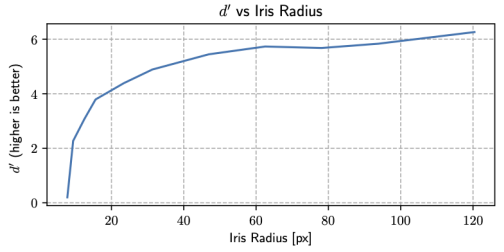


FIGURE 10. In IR tests, d' increases with iris radius.

Figure 11 shows in more detail the IR tests at three different resolutions when the iris radius is 120.4 px, 46.9 px, and 9.8 px. It can be seen that there is much more overlap at 9.2 px (Figure 11b) compared to the maximum resolution 120.4px (Figure 11a); that is why d' diminishes from 6.26 to 2.02. The Detection Error Trade-off (DET) curves in Figure 11c reveal the system behaviour for the three resolutions. It shows the expected result that errors decrement with higher resolutions; however, at an iris radius of 46.9 px, the EER increases only by 1% with respect to the maximum resolution.

Therefore, we take the value of 45px as the minimum iris radius for an acceptable IR recognition performance.

B. EXPERIMENT 2: IRIS RADIUS VS DISTANCE TO CAMERA

Now that a minimum iris radius of 45px has been established, we must correlate the iris radius with the distance to the camera. For this test, we captured four face images on the device at distances of 10, 15, 20, 25, 30, 40, 50, 60 and 70 cm from the camera. Then, the iris radii on the eight eyes contained in the four frames are manually measured, and the mean iris radius is scored for each distance value. We repeated this test three times with the following combination of cameras and processors: Raspberry-Pi with 5Mpx camera v1, Raspberry-Pi with 8Mpx camera v2, and Jetson-Nano with 8Mpx camera v2. The OpenCV [43] libraries used in the Raspberry-Pi program can only acquire images of 1,920 × 1,080 px from the cameras; however, the Jetson-Nano was able to read images of 3,264 × 1,848 px using GStreamer.

Figure 12 shows the relationship between the iris radius and the distance to the camera for each device. A red dashed line marks the 45px minimum iris radius found in the previous test. The Raspberry-Pi with the camera v1 has a similar curve to that of the Jetson-Nano with the camera v2 module. The intersection with the 45 px line occurs at 30cm and 35cm from the camera, respectively. Therefore, to work with an iris radius over 45 px, the distance to the camera must not exceed 35 cm.

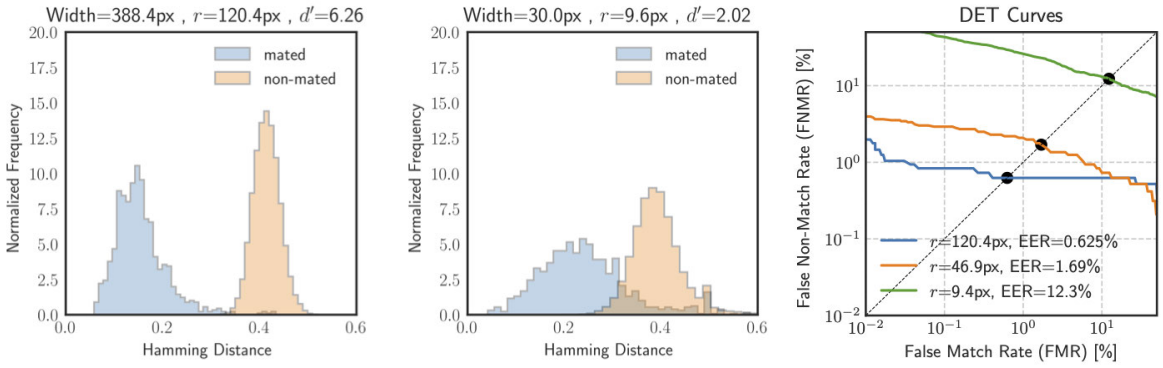
According to Figure 12, the Raspberry-Pi gets a lower resolution with the camera v2 module than using the camera v1 module at the same distance to the camera. That is because the camera v1 module possesses a magnification lens, as shown in Figure 3a. However, the Jetson-Nano is capable of getting even better iris resolutions than the Raspberry-Pi with the camera v1 module for the same distances as the camera without the magnification lens. The Jetson-Nano does it at the expense of working with higher-resolution images. Thus, we decided to use the camera v1 module exclusively with the Raspberry-Pi, and the camera v2 module with the Jetson-Nano in further tests.

C. EXPERIMENT 3: SIGNAL TO NOISE RATIO

In this experiment, we analyse the SNR at increasing distances from the camera. For this test, we used the camera v2 module with the Jetson-Nano. One image of the subject was taken at 10, 15, 20, 25, 30, 40, 50, 60 and 70cm from the camera, and the iris and the sclera of both eyes were manually segmented on each image. SNR was computed using (3) for each distance value.

In total, two capture sessions were taken, one during the day and one at night. This test can be seen in Figure 13.

In general, SNR diminishes when the distance to the camera increases both at day and night. However, SNR values are much greater during the day than at night. This means that at night the incident NIR light is not enough, and the camera has to compensate for the gain, exposure and contrast at the expense of adding extra noise. Figure 14 illustrates how at night, there is much more camera noise present in the image. According to the day curve in Figure 13, over 40 cm of distance, the SNR starts to drop more rapidly. Therefore, according to this test, the maximum distance with good SNR is 40 cm.



(a) (b) (c)

FIGURE 11. IR tests at different resolutions.

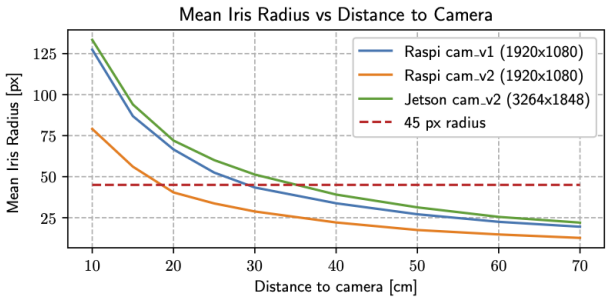


FIGURE 12. In IR tests, d' increases with iris radius.

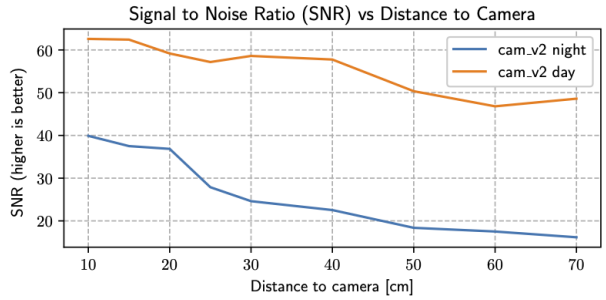


FIGURE 13. Signal to noise ratio at several camera distances.

FIGURE 14. Two Face ROI images at 60cm from the camera. Top: captured during the day. Bottom: captured at night.

D. EXPERIMENT 4: AVAILABLE IRIS AREA REGARDING SUBJECT'S GAZE

An important aspect to take into account is that the users might not look straight into the camera but instead, be distracted by the screen. Despite the camera and the screen being physically close, the subjects' gaze is different when looking at the camera than at the screen, as illustrated in Figure 15. When looking at the camera, more iris surface is present in the image, whereas when the subject directs their gaze downwards, eye leads and eyelashes obstruct the iris. We conducted the following experiment to quantify the amount of available iris surface. For the distances to the camera of 10, 15, 20, 25, 30, 40 and 50cm, an image was captured when the subject looked directly at the camera and another when looking at the screen. Then, the iris diameter along the x and y axis was measured for each eye (see Figure 15), and the mean ratio of Dy/Dx between the left and the right eye was scored for each frame. This ratio represents how much of the iris was occluded due to eye leads and eyelashes in the y axis with respect to the x axis, where the iris is never occluded. The results of this test are shown in Figure 17. When looking at the camera, the ratio settles at around 0.9 for distances over 25 cm. However, when looking at the screen, the ratio stabilises slightly below 0.8 from a distance of 30 cm. At a distance between 20 and 25 cm, the ratio is around 0.75, which might also be acceptable. Therefore, according to this test, the optimal distance to the camera is above 25 cm.

E. ANALYSIS OF EXPERIMENTS 1 THROUGH 4

In summary, the above experiments produced three different intervals to optimise different parameters. For good IR performance, the distance to the camera must not exceed 35cm. To get an acceptable SNR, the distance must be below 40cm. To obtain less eye lead and eyelash obstructions due to the subject's gaze, the distance to the camera has to be above 25cm. Figure 17 illustrates those intervals. The intersection of the three produces the optimal distance to the camera that

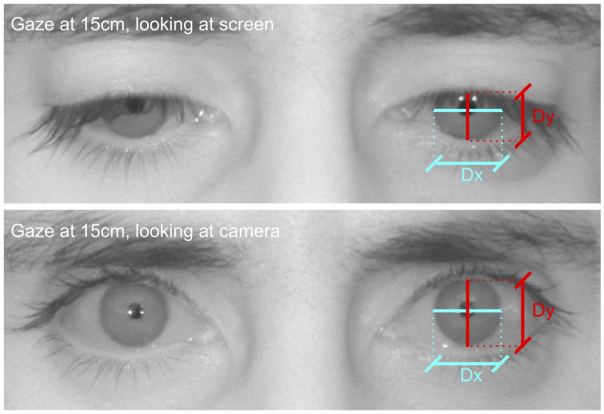


FIGURE 15. Gaze analysis at 15cm from the camera. Top: looking at the screen. Bottom: looking at the camera.

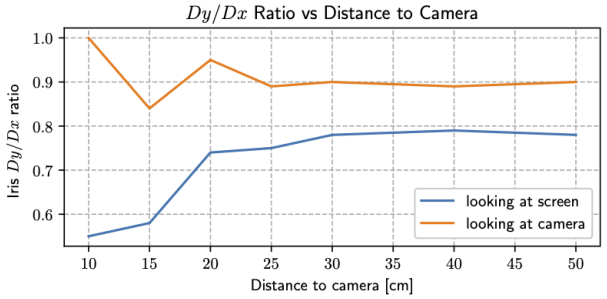


FIGURE 16. Eye aperture when looking at the camera or the screen for different distances to the camera.

satisfies the three criteria, and it is the interval between 25 and 35cm. Thus, we calibrated the focal plane of the v1 and v2 camera modules at 30cm. This is an important aspect of the hardware design that is usually overlooked when dealing with IR software.

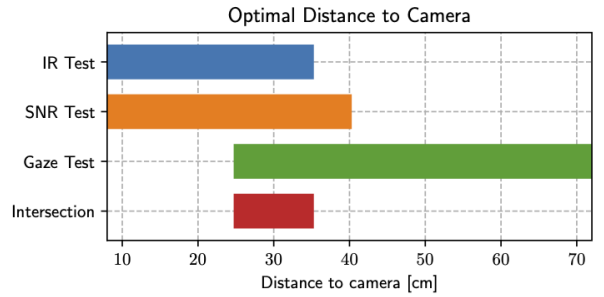


FIGURE 17. Intervals of optimal distance to the camera according to our experiments.

F. EXPERIMENT 5: EYE-FINDING ACCURACY AND SPEED

In this experiment, both segmentation accuracy and the processing speed are evaluated for the first task, which consists of finding eyes in face-ROI images. Each method takes as input an 800×480 image and scales it to the nominal resolution in which it operates. The fps measurement takes into account the resizing process.

We compared the IoU performance, Eye-tiny-yolo, and two tracking algorithms available in OpenCV [43], namely Channel and Spatial Reliability Tracking (CSRT) [44] and Kernelized Correlation Filter (KCF) [45], [46]. It was reported the mean IoU value in the test set $\pm$ is the standard deviation. It also measured the average inference speed in frames per second (fps) on two computers (PC1 and PC2), a Raspberry-Pi-4B (RasPi) and a Jetson-Nano (Jetson). PC1 has a Ryzen3 prossesor, 16 GB of RAM, and no GPU. PC2 has a Ryzen5 Processor, 16 GB of RAM and a 12 GB GPU. The input image size in this experiment was 800×480 pixels.

Table 2 shows the results of this experiment, as well as the input resolutions and the number of parameters used by each method. Because CSRT and KCF are image-processing-based methods, instead of CNN, they do not have trainable parameters. KCF obtained the fastest fps in platforms without a GPU; however, they require additional information about the eye's position in the first frame to start tracking.

In terms of IoU, Eye-tinny-yolo obtained by far the best performance, with 0.986; however, it does so at the expense of using 8 million parameters. CCNet achieved a mean IoU of 0.801 with 112,514 parameters. The mean IoU performance of the proposed UNet_xxs was 0.772, which is close to that of CCNet but using only 28,146 parameters. A 77.2% of the accurately recognised area still produces acceptable eye crops, where the entirety of the iris is inside the cropped image.

In terms of speed, UNet_xxs is by far the fastest network in all systems and the fastest method for systems with GPU. On the Jetson-Nano, CCNet doubles the speed of Eye-tiny-yolo, and UNet_xxs doubles that of CCNet. On the Raspberry-Pi, on the other hand, Eye-tiny-yolo is faster than CCNet; however, UNet_xxs is almost twice as fast as Eye-tiny-yolo.

Finally, the OpenCV algorithm KCF surpassed the speed of UNet_xxs on systems without a GPU. However, the fact that the captured subject (or biometric attendant) would have to click on the position of their eyes on the first frame makes it impractical on an end-to-end IR system.

G. EXPERIMENT 6: IRIS SEGMENTATION ACCURACY AND SPEED

We repeated Experiment 5 for the second task, which is iris segmentation. Speed (in fps) was evaluated using the same systems described in Experiment 5. The IoU metric was computed on the test set using DenseNet10, CCNet and the proposed UNet_xxs. The image size on the test set was 640×480 px.

Table 3 shows the input resolution, number of parameters and the results of this experiment. DenseNet10 produced the best IoU of 0.946, using 210,732 parameters. UNet_xxs obtained an IoU of 0.871, which is high considering that it only used 28,146 parameters to achieve it. Concerning speed, DenseNet10 was far slower than the other two, obtaining

TABLE 2. Evaluation of the eye finding task in terms of IoU accuracy and speed.

Method	N. Param.	Resolution [px]	IoU	PC1 (CPU) [fps]	PC2 (GPU) [fps]	RasPi [fps]	Jetson [fps]
OpenCV CSRT	NA	800×480	-	5.34	21.21	2.48	4.19
OpenCV KCF	NA	800×480	-	**17.14**	29.14	**4.79**	7.06
Eye-tiny-yolo	8,676,244	416×416	**0.986 ± 0.022**	3.60	42.69	1.74	4.07
CCNet	122,514	320×240	0.801 ± 0.139	6.53	22.86	1.18	8.11
UNet_xxs (ours)	28,146	160×96	0.772 ± 0.135	15.66	**80.21**	3.12	**16.18**

TABLE 3. CCNET parameters in terms of IoU accuracy and speed.

Method	N. Param.	Resolution [px]	IoU	PC1 (CPU) [fps]	PC2 (GPU) [fps]	RasPi [fps]	Jetson [fps]
DenseNet10	210,732	320 × 320	**0.946 ± 0.016**	0.77	29.47	0.32	1.87
CCNet	122,514	320 × 240	0.919 ± 0.040	7.89	188.60	1.60	12.65
UNet_xxs (ours)	28,146	128 × 96	0.871 ± 0.051	**25.78**	**343.22**	**3.52**	**38.47**

TABLE 4. Pupil and Iris localisation evaluation.

Segmentation Network	Localisation Method	Pupil Centre [px]	Pupil Radius [px]	Iris Centre [px]	Iris Radius [px]
DenseNet10	Centre of Mass	0.72 ± 4.36	0.65 ± 0.86	1.21 ± 0.91	1.18 ± 1.30
CCNet	Mixed (LMS + Hough)	2.47 ± 7.83	2.20 ± 1.89	3.83 ± 7.53	1.96 ± 1.75
UNet_xxs (ours)	Mixed (LMS + Hough)	4.46 ± 7.71	4.50 ± 2.69	5.00 ± 7.49	2.89 ± 2.92

TABLE 5. Iris recognition on ND-LG4000-LR.

Method	d'	EER	FNMR @ FMR=10%	FNMR @ FMR=1.0%	FNMR @ FMR=0.1%
DenseNet10	5.821	0.50%	0.22%	0.42%	0.70%
CCNet	4.543	1.79%	0.79%	2.19%	4.33%
UNet_xxs (ours)	3.959	2.56%	0.92%	4.35%	8.98%

speeds below 1 fps in some cases. On the other hand, CCNet and UNet_xxs were far quicker, with speeds of several fps. On the Jetson-Nano, CCNet obtained 12.65 fps, while UNet_xxs achieved 38.47. The latter is fast enough for IR frame-by-frame in real-time. Due to their lightweight, CCNet and UNet_xxs obtained hundreds of fps on a GPU machine. Finally, on the Raspberry-Pi, UNet_xxs can operate at 3.52 fps, which would produce IR results in a fraction of a second.

H. EXPERIMENT 7: EVALUATION OF PUPIL AND IRIS LOCALISATION

Once the iris surface has been segmented, the circles that best fit the pupil and iris must be computed. This is essential to obtain the Iris Code [1]. In this experiment, we measured the estimation error when predicting the pupil and iris circles. For the position of the centre of the circles, we measure the error in terms of the Euclidean Distance. On the other hand, we use the Absolute Distance for errors in the estimation of the radius. For predictions of DenseNet10, we used the centre of mass as the localisation algorithm, whereas for CCNet and UNet_xxs, we used the localisation mixed algorithm of Tapia et al. [11]. The latter uses LMS when the iris is not occluded by eye leads and Hough when it is occluded [11].

Table 4 shows the results of this experiment. DenseNet10's segmentation accuracy undoubtedly helped this method to achieve the best localisation performance with sub-pixel errors. CCNet had the most errors in the 2 px range, except for the iris centre, which was estimated with an average error of 3.83 px.

Finally, UNet_xxs produced high estimation errors because the errors are compounded from the segmentation prediction. Those errors are between 4 and 5 px in most cases. However, given that the image size is 640 × 480 px, a 5 px error is 1% of the image height, which, in this context, can be considered a small error.

I. ANALYSIS OF EXPERIMENTS 5, 6 AND 7

Experiments 5, 6 and 7 evaluated the proposed network's speed and accuracy against other state-of-the-art methods. In terms of speed, the proposed UNet_xxs was the fastest network in both tasks. This is especially true on the Jetson-Nano, which could crop eye images at 16.18 fps and segment them at 38.47 fps. With respect to performance, on the other hand, UNet_xxs did not perform as well as the other networks. Therefore, for an IR application, we recommend using UNet_xxs for the first task only and DenseNet10 or CCNet for the second task. In this way, the speed of

UNet_xxs would allow capturing several frames in real-time, cropping the eyes and choosing only the best quality images in terms of sharpness (ISO/IEC 29794-6) for the next process. The second task's speed is not as significant since a single prediction for each eye has to be performed. Therefore, DenseNet10 or CCNet would be better suited to yield results with good accuracy for iris segmentation and localisation.

J. EXPERIMENT 8: IRIS RECOGNITION PERFORMANCE

For a final benchmark evaluation of IR performance, we trained DenseNet10, CCNet and UNet_xxs on ND-LG4000-LR and performed IR tests using the ICA filters of Czajka et al. [5]. This test can reveal how the segmentation errors propagate and affect IR performance.

The DET curves shown in Figure 18 indicate that DenseNet10 has the best overall performance since its DET curve is lower than the others. CCNet has better performance than UNet_xxs for FAR smaller than 15%. However, UNet_xxs has better performance than CCNet for FAR greater than 15%.

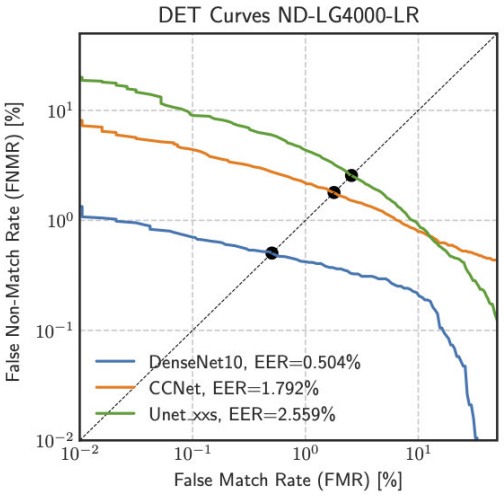


FIGURE 18. DET curves on ND-LG4000-LR dataset.

Table 5 shows the d' and EER of the three networks and the FNMR at three operating points. As in previous tests, DenseNet10 has the best performance. However, the EER of UNet_xxs is 2.56%, which is acceptable for real operations. If a 10% of False Match is permitted in the system, the three networks would have less than a 1% of False Rejections Rate. On the other hand, when the system allows only a 1% of FMR, UNet_xxs doubles the FNMR of CCNet at 4.35%, which is not as desirable. This corroborates that there is a trade-off between speed and performance when using UNet_xxs.

Finally, note that the EER and FNMR values reported in this work, even for the worst case, are smaller than those reported by Fang et al. [24] on a different partition on the Notre Dame dataset. In that work, CCNet obtained an EER of 4.39%. This indicates that UNet_xxs has competitive performance and can be used in a real-world scenario.

VII. DISCUSSION AND COMPARISONS

This work proposed building NIR hardware from scratch and adapting the algorithms with the methods available in the state of the art to be portable and included in an embedded device. While our method proposes improvements on both hardware and software levels, most papers focus only on one development aspect, such as using a powerful FPGA. Our method shows that it is feasible to create a low-cost iris sensor using different platforms such as Raspberry PI, Jetson Nano and a laptop with CPU and GPU. Throughout the paper, we show that it is not a trivial task, and it is necessary to look for the trade-off between power, speed and efficiency. The proposed segmentation network, UNet_xxs, reduces the number of parameters compared to existing models such as CCNet and DenseNet10. Tables 2, 3, 4, 5, compares the performance, speed, parameters and accuracy of the four methods. Further, the calibration process, distance, focus and resolution are explored and explained. The proposed method focuses on a non-contact solution, unlike methods that need to be in direct contact with the head of the subject. To the best of our knowledge, our method is the only one to implement IR and measure the performance of the Jetson Nano board. Finally, we give a detailed analysis of design criteria and power, which is not provided by other works. A primary limitation and future work of our proposal is to study and improve the synchronisations of NIR light with the camera using different activations such as cross-light and others.

VIII. CONCLUSION

This work comprehensively analysed the technical aspects of implementing a NIR iris imaging embedded device. The proposed device is lightweight and portable, extracts information from the two eyes, works with NIR illumination, and uses inexpensive single-board computers. We thoughtfully described the factors that affect sensor calibration (iris resolution, SNR and subject's gaze) and how to obtain the best trade-off. The optimal distance to fix the focal plane for the implemented device was 30 ± 5 cm.

The necessary processes needed for an end-to-end IR pipeline were also described. The proposed network UNet_xxs was designed as a lightweight solution for cropping eyes in a face-ROI image and then segmenting the valid iris area in those images. This network achieved the fastest speed among other CNNs for both tasks and reasonable levels of performance. However, due to their greater accuracy, we recommend using UNet_xxs for finding eyes only and DenseNet10 or CCNet for iris segmentation.

Moreover, for IR, we used the ICA BSIF filters implemented on Raspberry Pi by Fang et al. This method normalises and encodes the iris with great performance. For instance, we obtained an EER of 2.56%, 1.79% and 0.50% when segmenting with UNet_xxs, CCNet and DenseNet10, respectively. Those values are smaller than previous state-of-the-art works.

The described device and methodology make it possible to assemble an NIR imaging device using readily available

components. This expands the availability of iris devices at a low cost, which could spark new iris biometrics research in labs with limited resources. Future improvements to the proposed system include adding a PAD stage and using CNN-based iris encoding and IR.

REFERENCES

[1] J. Daugman, "How iris recognition works," in *The Essential Guide to Image Processing*. Amsterdam, The Netherlands: Elsevier, 2009, pp. 715–739.

[2] K. W. Bowyer and M. J. Burge, *Handbook of Iris Recognition*. Cham, Switzerland: Springer, 2016.

[3] N. Othman, B. Dorizzi, and S. Garcia-Salicetti, "OSIRIS: An open source iris recognition software," *Pattern Recognit. Lett.*, vol. 82, pp. 124–131, Oct. 2016.

[4] Z. He, Z. Sun, T. Tan, and Z. Wei, "Efficient iris spoof detection via boosted local binary patterns," in *Proc. Int. Conf. Biometrics*. Cham, Switzerland: Springer, 2009, pp. 1080–1090.

[5] A. Czajka, D. Moreira, K. Bowyer, and P. Flynn, "Domain-specific human-inspired binarized statistical image features for iris recognition," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 959–967.

[6] A. Gangwar and A. Joshi, "DeepIrisNet: Deep iris representation with applications in iris recognition and cross-sensor iris recognition," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 2301–2305.

[7] K. Nguyen, C. Fookes, A. Ross, and S. Sridharan, "Iris recognition with off-the-shelf CNN features: A deep learning perspective," *IEEE Access*, vol. 6, pp. 18848–18855, 2018.

[8] Z. Zhao and A. Kumar, "Towards more accurate iris recognition using deeply learned spatially corresponding features," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 3829–3838.

[9] S. Minaee and A. Abdolrashidi, "DeepIris: Iris recognition using a deep learning approach," 2019, *arXiv:1907.09380*.

[10] T. Zhao, Y. Liu, G. Huo, and X. Zhu, "A deep learning iris recognition method based on capsule network architecture," *IEEE Access*, vol. 7, pp. 49691–49701, 2019.

[11] J. E. Tapia, E. L. Droguett, A. Valenzuela, D. P. Benalcazar, L. Causa, and C. Busch, "Semantic segmentation of periocular near-infra-red eye images under alcohol effects," *IEEE Access*, vol. 9, pp. 109732–109744, 2021.

[12] K. Grabowski, W. Sankowski, M. Napieralska, M. Zubert, and A. Napieralski, "Iris recognition algorithm optimized for hardware implementation," in *Proc. IEEE Symp. Comput. Intell. Bioinf. Comput. Biol.*, Sep. 2006, pp. 1–5.

[13] R. Hentati, M. Hentati, M. Bousselmi, and M. Abid, "Hardware implementation of iris recognition algorithm," in *Proc. Int. Conf. Commun., Comput. Control Appl. (CCCA)*, Mar. 2011, pp. 1–6.

[14] J. Avey, P. Jones, and J. Zambreno, "An FPGA-based hardware accelerator for iris segmentation," in *Proc. Int. Conf. ReConFigurable Comput. FPGAs (ReConFig)*, Dec. 2018, pp. 1–8.

[15] L. Ma, C.-W. Sham, C. Y. Lo, and X. Zhong, "An effective multi-mode iris authentication system on a microprocessor-FPGA heterogeneous platform with QC-LDPC codes," *IEEE Access*, vol. 9, pp. 163665–163674, 2021.

[16] K. Grabowski and A. Napieralski, "Hardware architecture optimized for iris recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 9, pp. 1293–1303, Sep. 2011.

[17] C.-T. Chou, S.-W. Shih, W.-S. Chen, V. W. Cheng, and D.-Y. Chen, "Non-orthogonal view iris recognition system," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 3, pp. 417–430, Mar. 2010.

[18] F. Reidj G. Cruz, C. C. Hortinela, B. E. Redosendo, B. K. P. Asuncion, C. J. S. Leoncio, N. B. Linsangan, and W.-Y. Chung, "Iris recognition using Daugman algorithm on raspberry Pi," in *Proc. IEEE Region Conf. (TENCON)*, Nov. 2016, pp. 2126–2129.

[19] Z. Kunik, A. Bykowski, T. Marciniak, and A. Dabrowski, "Raspberry pi based complete embedded system for iris recognition," in *Proc. Signal Process., Algorithms, Archit., Arrangements, Appl. (SPA)*, Sep. 2017, pp. 263–268.

[20] F. Boutros, N. Damer, K. Raja, R. Ramachandra, F. Kirchbuchner, and A. Kuijper, "Iris and periocular biometrics for head mounted displays: Segmentation, recognition, and synthetic data generation," *Image Vis. Comput.*, vol. 104, Dec. 2020, Art. no. 104007.

[21] F. Boutros, N. Damer, K. Raja, R. Ramachandra, F. Kirchbuchner, and A. Kuijper, "Fusing iris and periocular region for user verification in head mounted displays," in *Proc. IEEE 23rd Int. Conf. Inf. Fusion (FUSION)*, Jul. 2020, pp. 1–8.

[22] Q. Zhang, H. Li, Z. Sun, and T. Tan, "Deep feature fusion for iris and periocular biometrics on mobile devices," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 11, pp. 2897–2912, Nov. 2018.

[23] D. Bastias, C. A. Perez, D. P. Benalcazar, and K. W. Bowyer, "A method for 3D iris reconstruction from multiple 2D near-infrared images," in *Proc. IEEE Int. Joint Conf. Biometrics (IJCB)*, Oct. 2017, pp. 503–509.

[24] Z. Fang and A. Czajka, "Open source iris recognition hardware and software with presentation attack detection," in *Proc. IEEE Int. Joint Conf. Biometrics (IJCB)*, Sep. 2020, pp. 1–8.

[25] K. B. Raja, R. Raghavendra, V. K. Vemuri, and C. Busch, "Smartphone based visible iris recognition using deep sparse filtering," *Pattern Recognit. Lett.*, vol. 57, pp. 33–42, May 2015.

[26] J. E. Tapia, A. Valenzuela, R. Lara, M. Gomez-Barrero, and C. Busch, "Selfie periocular verification using an efficient super-resolution approach," *IEEE Access*, vol. 10, pp. 67573–67589, 2022.

[27] D. Benalcazar, C. Perez, D. Bastias, and K. Bowyer, "Iris recognition: Comparing visible-light lateral and frontal illumination to NIR frontal illumination," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 867–876.

[28] D. P. Benalcazar, D. Bastias, C. A. Perez, and K. W. Bowyer, "A 3D iris scanner from multiple 2D visible light images," *IEEE Access*, vol. 7, pp. 61461–61472, 2019.

[29] D. P. Benalcazar, J. E. Zambrano, D. Bastias, C. A. Perez, and K. W. Bowyer, "A 3D iris scanner from a single image using convolutional neural networks," *IEEE Access*, vol. 8, pp. 98584–98599, 2020.

[30] D. P. Benalcazar, D. A. Montecino, J. E. Zambrano, C. A. Perez, and K. W. Bowyer, "3D iris recognition using spin images," in *Proc. IEEE Int. Joint Conf. Biometrics (IJCB)*, Sep. 2020, pp. 1–8.

[31] F. Boutros, N. Damer, F. Kirchbuchner, and A. Kuijper, "Eye-MMS: Miniature multi-scale segmentation network of key eye-regions in embedded applications," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 3665–3670.

[32] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent*. Cham, Switzerland: Springer, 2015, pp. 234–241.

[33] S. Mishra, P. Liang, A. Czajka, D. Z. Chen, and X. S. Hu, "CC-NET: Image complexity guided network compression for biomedical image segmentation," in *Proc. IEEE 16th Int. Symp. Biomed. Imag. (ISBI)*, Apr. 2019, pp. 57–60.

[34] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "DenseNet: Implementing efficient ConvNet descriptor pyramids," 2014, *arXiv:1404.1869*.

[35] X. Wu and L. Zhao, "Study on iris segmentation algorithm based on dense U-Net," *IEEE Access*, vol. 7, pp. 123959–123968, 2019.

[36] B. Hassan, R. Ahmed, T. Hassan, and N. Werghi, "SIP-SegNet: A deep convolutional encoder–decoder network for joint semantic segmentation and extraction of sclera, iris and pupil based on periocular region suppression," 2020, *arXiv:2003.00825*.

[37] Q. Wang, X. Meng, T. Sun, and X. Zhang, "A light iris segmentation network," *Vis. Comput.*, vol. 38, no. 7, pp. 2591–2601, Jul. 2022.

[38] D. Osorio-Roig, A. Morales-González, and E. Garea-Llano, "Semantic segmentation of color eye images for improving iris segmentation," in *Iberoamerican Congress on Pattern Recognition*. Cham, Switzerland: Springer, 2017, pp. 466–474.

[39] D. Osorio-Roig, C. Rathgeb, M. Gomez-Barrero, A. Morales-González, E. Garea-Llano, and C. Busch, "Visible wavelength iris segmentation: A multi-class approach using fully convolutional neuronal networks," in *Proc. Int. Conf. Biometrics Special Interest Group (BIOSIG)*, Sep. 2018, pp. 1–5.

[40] Y.-H. Li, W. R. Putri, M. S. Aslam, and C.-C. Chang, "Robust iris segmentation algorithm in non-cooperative environments using interleaved residual U-Net," *Sensors*, vol. 21, no. 4, p. 1434, Feb. 2021.

[41] J. Daugman, "How iris recognition works," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 1, pp. 21–30, Jan. 2004.

[42] U. of Notre Dame USA. (2012). *Crosssensor-Iris-2012-Data-Set*. [Online]. Available: https://cvrl.nd.edu/projects/data/#nd-crosssensor-iris-2012-data-set

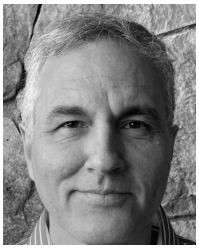
[43] G. Bradski, "The OpenCV library," *Dr. Dobb's J. Softw. Tools*, 2000, Art. no. 2236121.
[44] A. LuNežič, T. Vojíř, L. Čehovin Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter tracner with channel and spatial reliability," *Intl. J. Comput. Vis.*, vol. 126, no. 7, pp. 671–688, 2018.
[45] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2012, pp. 702–715.
[46] M. Danelljan, F. S. Khan, M. Felsberg, and J. Van De Weijer, "Adaptive color attributes for real-time visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1090–1097.

DANIEL P. BENALCAZAR (Member, IEEE) was born in Quito, Ecuador, in 1987. He received the B.S. degree in electronics and control engineering from Escuela Politecnica Nacional, Quito, in 2012, the M.S. degree in electrical engineering from The University of Queensland, Brisbane, Australia, in 2014, with a minor in biomedical engineering, and the Ph.D. degree in electrical engineering from Universidad de Chile, Santiago, Chile, in 2020. Ever since, he has been participating in various biomedical engineering and biometrics research projects. From 2015 to 2016, he was a Professor with the Central University of Ecuador. He is currently a Senior Researcher with the Department of Mechanical Engineering, Universidad de Chile, Santiago.

JUAN E. TAPIA (Member, IEEE) received the P.E. degree in electronics engineering from Universidad Mayor, in 2004, the M.S. degree in electrical engineering from Universidad de Chile, in 2012, and the Ph.D. degree from the Department of Electrical Engineering, Universidad de Chile, in 2016. In addition, he spent one year of internship with the University of Notre Dame. From 2016 to 2017, he was an Assistant Professor with Universidad Andres Bello. From 2018 to 2020, he was the Research and Development Director of Electricity and Electronics with Universidad Tecnologica de Chile—INACAP and Research and Development Director of TOC Biometrics, in November 2022. He is currently a Senior Researcher with Hochschule Darmstadt (HDA). His main research interests include pattern recognition and deep learning applied to iris biometrics, morphing, feature fusion, and feature selection. In 2016, he received the award for the best Ph.D. thesis.

MAURICIO VASQUEZ received the B.S. degree in computer engineering from the Faculty of Informatics, Universidad del Bió-Bió, Chile, in 2000. He is currently pursuing the Master of Science degree with Universidad de Santiago de Chile. His main research interests include computer vision, graphic interfaces, and UX applied to biometrics.

LEONARDO CAUSA received the P.E. degree in electrical engineering and the M.S. degree in biomedical engineering (BME) from Universidad de Chile, in 2012. He is currently pursuing the joint Ph.D. degree in electrical engineering and medical informatics by tutelage from Universidad de Chile and Universite Claude Bernard Lyon 1. His research interests include sleep pattern recognition, signal and image processing, neuro-fuzzy systems applied to the classification of physiological data, and machine and deep learning. He was engaged in research on automated sleep-pattern detection and respiratory signal analysis, fitness for duty, drowsiness, alertness, and performance.



ENRIQUE LÓPEZ DROGUETT is currently a Professor with the Civil and Environmental Engineering Department and the Garrick Institute for the Risk Sciences, University of California at Los Angeles, Los Angeles (UCLA), USA, and an Associate Editor for both the *Journal of Risk and Reliability* and the *International Journal of Reliability and Safety*. He conducts research on Bayesian inference and artificial intelligence-supported digital twins and prognostics and health management based on physics-informed deep learning for reliability, risk, and safety assessment of structural and mechanical systems. His most recent focus has been on quantum computing and quantum machine learning for developing solutions for risk and reliability quantification and energy efficiency of complex systems, particularly those involved in renewable energy production. He has led many major studies on these topics for a broad range of industries, including oil and gas, nuclear energy, defence, civil aviation, mining, renewable, and hydro energy production and distribution networks. He has authored more than 250 papers in archival journals and conference proceedings. He also serves on the Board of Directors for the International Association for Probabilistic Safety Assessment and Management (IAPSAM).

CHRISTOPH BUSCH (Senior Member, IEEE) is currently a member of the Department of Information Security and Communication Technology (IIK), Norwegian University of Science and Technology (NTNU), Norway. He holds a joint appointment with the Computer Science Faculty, Hochschule Darmstadt (HDA), Germany. Furthermore, he has been lecturing on the biometric systems course with DTU, Denmark, since 2007. On behalf of the German BSI, he has been the Coordinator for the project series BioIS, BioFace, BioFinger, BioKeyS Pilot-DB, KBEinweg, and NFIQ2.0. In the European Research Program, he was the Initiator of the Integrated Project 3D-Face, FIDELITY, and iMARS. Furthermore, he was/is a Partner in TURBINE, BEST Network, ORIGINS, INGRESS, PIDaaS, SOTAMD, RESPECT, and TReSPAsS projects. He is also a Principal Investigator with the German National Research Center for Applied Cybersecurity (ATHENE). Moreover, he was a Co-Founder of the European Association for Biometrics (www.eab.org), which was established, in 2011, and has assembled more than 200 institutional members. He has coauthored more than 600 technical papers and has been a speaker at international conferences. He is a member of the editorial board of the IET journals.