Journal of Applied Research and Technology

www.jart.icat.unam.mx

*Journal of Applied Research and Technology* 22 (2024) 102-110

Original

# Pulse-coupled neural network based on an adaptive Gabor filter for pavement crack segmentation

A. Luna Álvarez[a] • D. Mújica Vargas[a*] • J. de J. Rubio[b] • A. Rosales Silva[c]

[a]*Computer Science Department, Tecnológico Nacional de México/CENIDET, Cuernavaca, Morelos, México*
[b]*Postgraduate Studies and Research Section, ESIME Azcapotzalco, Instituto Politécnico Nacional, Ciudad de México, México.*
[c]*Postgraduate Studies and Research Section, ESIME Zacatenco, Instituto Politécnico Nacional, Ciudad de México, México.*

Received 11 08 2021; accepted 04 20 2022
Available 02 29 2024

**Abstract:** This article proposes a pulse-coupled neural network based on an adaptive Gabor filter for pavement crack segmentation in digital images. By estimating the image noise, the filter parameters that convolves the neurons of the model are adjusted. As a result, iterations were reduced to 2% with ≈ 90% precision. The algorithm was parallelized on GPU and the processing time was reduced to $\frac{n}{NM}$ regardless of the *M* and *N* dimensions of the image.

*Keywords:* Pulse-coupled neural network, adaptive Gabor filter, parallel processing, crack segmentation

[*]Corresponding author.
*E-mail address:* dante.mv@cenidet.tecnm.mx (D. Mújica Vargas).
Peer Review under the responsibility of Universidad Nacional Autónoma de México.

## 1. Introduction

To automate the crack detection task, solutions have been proposed based on artificial vision implemented in the monitoring vehicles (Shao et al., 2019; Yang et al., 2019). These solutions are made up of combinations of preprocessing, feature extraction and classification methods (Amhaz et al., 2016a; Cubero-Fernandez et al., 2017; Yang et al., 2019), segmentation through filters (Li et al., 2017; Li et al., 2020) or through training and testing of deep learning neural network model (Gopalakrishnan et al., 2017; Pauly et al., 2017; Zhang et al., 2018; Zhang et al., 2016).

Although the methods guarantee accuracy close to 90%, they have disadvantages. Methods that rely on the image processing chain take the longest time, which requires the monitoring vehicle to stop (Amhaz et al., 2016b). Filter-based methods are mostly affected by noise captured by the device, as well as minor imperfections in the pavement texture. Deep learning methods are shown to be the most dependable (Lau et al., 2020), however these depend on training and the image quality for this task, in addition to requiring a specialized team to implement them.

As an alternative to these methods, this article proposes a pulse-coupled neural network (PCNN) (Wang et al., 2010) model that bases its operation on a Gabor filter (Mukherjee & Das, 2021) that adjusts its parameters automatically from the noise estimate in the input image. This model has the advantages of being adaptive, without training and notably faster than the original PCNN model in terms of iterations, in addition to being parallelized in a graphic processing unit in CUDA language.

This document details in Section 2 the theoretical concepts necessary to understand the PCNN model and the Gabor filter, in Section 3 the proposed model is presented, in Section 4 the test data, metrics and experiments are presented to validate the proposal. Finally, Section 5 shows the results obtained and the conclusions in Section 6.

## 2. Background

### *2.1. Pulse-coupled neural network*

Pulse-coupled neural networks are biometric models inspired by the visual cortex of mammals, developed for image processing (Zhan et al., 2017), applicable to segmentation, feature generation, face extraction, motion detection, region growth and noise reduction tasks (Wang et al., 2010). As shown in Figure 1, the PCNN model is made up of four main components: the dendritic tree that contains an input $S$ of size $I \times J$, the neuron $W_{ij}$and its neighbors, the membrane potential $U$, the dynamic threshold $\Theta$ and the outputs $Y$ that become inputs for subsequent iterations. In general, the dendritic tree represents the data input filtered by a Gaussian matrix $W$ that smooths the edges of each $S$ window, then the dynamic threshold $\Theta$ is initialized, scaled by the parameters $\delta$ y $v_\theta$ with respect to the output of each neighboring neuron $Y_{kl}$ in an iteration $n-1$, which for $n=0$: $Y_{ij}=0, \Theta_{ij}=0$ (Lian et al., 2021). In the pulse generation block, the membrane potential $U_{ij}$ is updated by $U_{ij} \leftarrow S_{ij}(1+\beta L_{ij})$ where $L$ represents the input filtered and $S$ the original, it is recommended for the hyperparameter $\beta=2$. The membrane potential is activated in a binary way at the output $Y_{ij}$ through $Y_{ij}=U_{ij}>\Theta_{ij}$, this process is repeated until $Y_{ij}(n) \neq Y_{ij}(n-1)$, otherwise the image is filtered again. Finally, for the outputs of each neuron to be projected in a segmented image, a time matrix $T$ is updated by $T_{ij}(n) < T_{ij}(n-1) + nY_{ij}$, in this way the time matrix is filled with values $[0,255]$

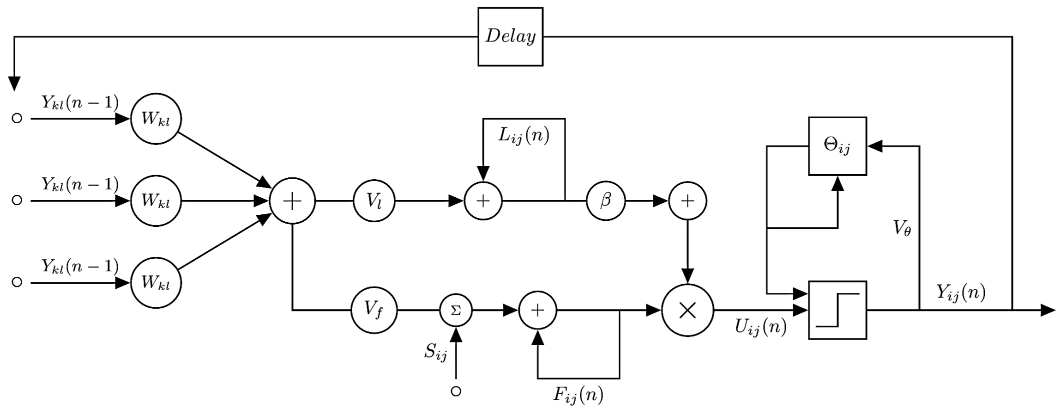

Figure 1. Simple PCNN scheme.

that can be represented as an 8-bit grayscale image. In a simplified way, the operation of the PCNN model for segmentation in the Algorithm 1 is detailed (Nie et al., 2020), considering as initialization of the matrices in $n = 0$, $Y = 0$, $T = 0, \Theta = 255$.

**Algorithm 1** PCNN model for segmentation

```
Require: $F \in \mathbb{R}^{M \times N}$
Ensure: $T \in \mathbb{R}^{M \times N}$
 1: $W_{ij} \leftarrow f(x_{ij}) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x_{ij}-\mu)^2}{2\sigma^2}} : W \in \mathbb{R}^{M \times N}$
 2: while $fn < \mathcal{N}$ do
 3:     $n \leftarrow n + 1$
 4:     $L \leftarrow W \times Y$
 5:     $\Theta \leftarrow \Theta - \delta + v_\theta Y$
 6:     while $fn < \mathcal{N}$ do
 7:         $U \leftarrow S\ (1 + \beta L)$
 8:         $Y \leftarrow U > \Theta$
 9:         if $Q = Y$ then
10:             $f \leftarrow 0$
11:         else
12:             $L \leftarrow W \times Y$
13:         end if
14:     end while
15:     $fn \leftarrow fn + \sum_{i=1}^{M} \sum_{j=1}^{N} Y_{ij}$
16:     $T \leftarrow T + nY$
17: end while
18: return $T$
```

## 2.2. Gabor filter

In image processing, it is a linear filter used for texture analysis (Mukherjee & Das, 2021), it analyzes the specific frequencies and directions in a certain region. A 2D Gabor filter is a Gaussian filter function modulated by a plane sine wave. In general, the function of the Gabor filter for a two-dimensional image is defined as:

$$G_{xy} = exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) cos\left(2\pi\frac{x'}{\lambda} + \varphi\right) \tag{1}$$

where $x' = x\ cos\ \theta + y\ sin\ \theta$ y $y' = -x\ sin\theta + y\ cos\theta$ is defined, in addition the function parameters are defined as:

- Standard deviation $\sigma$ of the Gaussian envelope.
- Wavelength $\lambda$, is the length of the filter cosine factor specified in pixels $2 \leq \lambda < X$
- Orientation θ, specifies the normal to the parallel fringes of a Gabor function specified in degrees 0≤θ<360.
- Phase offset φ, specified in degrees [-180,180]. 0 and 180 correspond to symmetric functions.
- Aspect ratio γ, specifies the ellipticity of the function support.

## 3. Proposed method

To improve the cracks segmentation, an adaptive PCNN model is proposed with the following characteristics: (1) the Gaussian filter is replaced with a Gabor filter to enhance the textures and obtain segmentation based on frequency and direction. (2) The filter requires a wavelength λ and standard deviation σ setup, therefore an automatic tuning based on noise estimation is proposed. (3) The PCNN structure processes each image pixel as an independent neuron, but depends on the surrounding region, therefore it is implemented in a parallelized methodology using GPU.

### 3.1. Noise estimation

For PCNN purposes, the pixel range should be $[0{,}255]$. The noise estimation requires an interval $[0{,}1]$, therefore the normalization is performed by:

$$x'_{ij} = \frac{x_{ij} - min(x)}{max(x) - min(x)} \tag{2}$$

considering that $min(x)$ and $max(x)$ in a normal case can be 0 and 255, however the intervals are closed as the iterations progress, and this must be adjusted according to the image values. Grayscale pavement images are altered by additive Gaussian noise, which is represented as $F_{n_{ij}} = F_{ij} + n_{ij}$. The standard deviation of the additive noise distribution can be estimated by:

$$\sigma = \left(\sum \quad \left|\left|F \times \sigma_E\right|\right|\right)\sqrt{\frac{\pi}{2}\frac{1}{6(M-2)(N-2)}} \tag{3}$$

where $M$ and $N$ represent the size of the image, and $\sigma_E$ is the Laplacian convolution operator (Kim & Shamsi, 2018) defined as:

$$\sigma_E = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \tag{4}$$

The Laplacian of an image highlights regions of rapid intensity change. From this, the standard deviation of additive Gaussian noise is estimated to be in the range of 3% to 21%. With this estimated level it is possible to adjust the deviation of the Gabor filter in $\sigma \in [0.03, 0.21]$.

### 3.2. Model adaptation

From the noise estimation, the Gabor filter becomes adaptive and must be updated every time that $L$ image is filtered and gets a pulse $Q \neq Y$ , so in Algorithm 1, Step 1 must be replaced by the estimate and update function of the filter to obtain the initial noise, as well as adding this function prior to Step 16. In Algorithm 2 the update function is detailed.

For this experimentation, the parameters for the Gabor filter detailed in (Khan et al., 2016) were followed, specifically: $\theta = 0$, $\lambda = \frac{1}{\sigma}$, $\varphi = 0$ and $\gamma = 1$. The filter G was defined as an array of $5 \times 5$. Figure 2 shows the flowchart detailing the PCNN algorithm with adaptive filter. Receives the original image $S$ and returns the time matrix $T$ , which are structured in grayscale in the value interval $[0,255]$. The $Filter\ Update(\cdot)$ blocks refer to the Algorithm 2, which receives as a parameter an array $I \in Z^{M\times N}$ returns the filter $G$ The proposed model is formalized in the diagram in Figure 3, where the filter update is highlighted at the top with the use of the external filter $\sigma_E$.

These modifications allow the PCNN to converge in fewer iterations, the recurring update allows to obtain better delimited regions. The extra function adds a linear complexity $O(n)$, where $n = M \cdot N$. However, the operations of the data structures are performed in a linear process, therefore this algorithm is easily parallelized to speed up the response time.

**Algorithm 2** Gabor filter update function

```
Require: $I \in \mathbb{Z}^{M\times N}$
Ensure: $G \in \mathbb{R}^{5\times 5}$
1: $Y \leftarrow \sigma_E \times I$
2: $\sum_{i\leftarrow 0}^{M}\sum_{j\leftarrow 0}^{N} \sigma_e \leftarrow \sigma_e + \|Y_{i,j}\|$
3: $\sigma \leftarrow \sigma_e \sqrt{\frac{\pi}{2}\frac{1}{6(M-2)(N-2)}}$
4: for $x, y \leftarrow -2$ to $x, y = 2$ do
5:     $x' \leftarrow x\cos\theta + y\sin\theta$
6:     $y' \leftarrow -x\sin\theta + y\cos\theta$
7:     $G_{x,y} \leftarrow \exp\left(-\frac{x'^2+\gamma^2+y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda}+\varphi\right)$
8: end for
9: return $G$
```

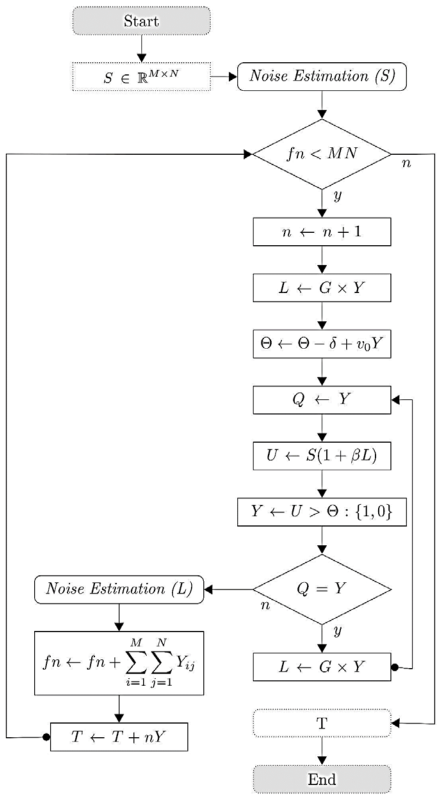
Start
$S \in \mathbb{R}^{M\times N}$
Noise Estimation (S)
$fn < MN$
n
y
$n \leftarrow n+1$
$L \leftarrow G \times Y$
$\Theta \leftarrow \Theta - \delta + v_0 Y$
$Q \leftarrow Y$
$U \leftarrow S(1+\beta L)$
$Y \leftarrow U > \Theta : \{1,0\}$
Noise Estimation (L)
$Q = Y$
n
y
$fn \leftarrow fn + \sum_{i=1}^{M}\sum_{j=1}^{N} Y_{ij}$
$L \leftarrow G \times Y$
$T \leftarrow T + nY$
T
End

Figure 2. Proposed adaptive PCNN algorithm flowchart.

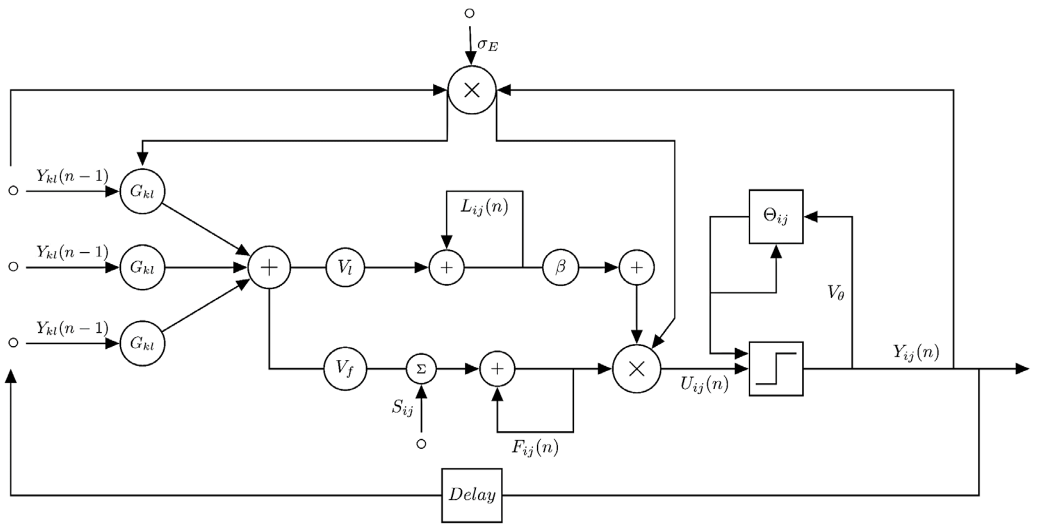

Figure 3. Proposed adaptive PCNN algorithm flowchart.

### 3.3. Parallelization

To perform GPU acceleration, the algorithm was written in CUDA kernels using vectorized data structures. This means that the image $S$ was reduced to a vector of size $M \cdot N$, therefore the matrices used have the same structure. In Algorithm 3 the sequential and parallel processing blocks are detailed.

Although $L, Y\ Q, U, Y$ and $\Theta$ are defined as $M \cdot N$ vectors, as well as $G$ and $\sigma_E$ as $K^2$ vectors, the image $S$ is read as a two-dimensional structure. To process it, it is necessary to perform vectorization from a queued transformation. That is, every $M$ value is stacked in the $N + 1$ row.

For parallel kernels, an index $i$ is defined within each procedure referring to the GPU processing thread. In the algorithm, Step 5 indicates that $i$ is obtained from the block index with respect to the total blocks and the processing thread. In practice $i \in [0, n]$, where n represents the kernel threads used, calculated from the grid parameters product $<< p1, p2 >>$ in the main thread. For each kernel, $i$ represents a PCNN neuron, so that each operation can be performed at the same time unlike the original algorithm, except for the sequential procedure since it requires a scan of the image. For convolution, the surrounding region is necessary, for this reason the local sub-indexes are calculated. Is it possible to reduce the processing time from $n$ to $\frac{n}{MN}$, without affecting the quality of the segmentation.

**Algorithm 3** Parallel PCNN algorithm

```
Require: $M, N \in \mathbb{Z}, S \in \mathbb{Z}^{M\times N}, \beta, \delta, v_0 \in \mathbb{R}$
Ensure: $T \in \mathbb{Z}^{M\times N}$
Sequential block
 1: procedure VARIANCE($Y, M, N$)
 2:     $\sigma \leftarrow \left(\sum_{i\leftarrow 0}^{MN} Y_i\right)\sqrt{\frac{\pi}{2}\frac{1}{6(M-2)(N-2)}}$
 3:     return $\sigma$
 4: end procedure
Parallel kernel block
 5: $i \leftarrow blockIdx * blockDim + threadIdx$
 6: procedure CONVOLUTION($I, W$)
 7:     $Y_i \leftarrow \sum_{m,n\leftarrow -\frac{k}{2}}^{\frac{k}{2}} I_{(\frac{i}{M}+m)*N+(i\%r+n)} * W_{k*(m+\frac{k}{2})+(n+\frac{k}{2})}$
 8:     return $Y$
 9: end procedure
10: procedure FILTER UPDATE($I$)
11:     $Y \leftarrow$ CONVOLUTION($I, \sigma_E$)
12:     $\sigma \leftarrow$ VARIANCE($Y, M, N$)
13:     $x_i' \leftarrow i - k/2 * \cos(\theta) + i\%k/2\ \sin(\theta)$
14:     $y_i' \leftarrow -(i - k/2) * \sin(\theta) + i\%k/2\ \cos(\theta)$
15:     $G_i \leftarrow \exp\left(-\frac{x_i'^2+\gamma^2+y_i'^2}{2\sigma^2}\right) * \cos\left(2\pi\frac{x_i'}{\lambda}+\varphi\right)$
16:     return $G$
17: end procedure
18: procedure THRESHOLD($Y$)
19:     $\Theta_i \leftarrow \Theta_i - \delta + v_0 Y_i$
20:     return $\Theta$
21: end procedure
22: procedure PULSES($S, Y, L$)
23:     $U_i \leftarrow S_i(1 + \beta L_i)$
24:     $Y_i \leftarrow U_i > \Theta_i$
25:     return $Y$
26: end procedure
27: procedure TIME MATRIX($T, Y, n$)
28:     $T_i \leftarrow T_i + nY_i$
29:     return $Y$
30: end procedure
Main Threat
31: $S_{M\times N} \mapsto S_{(MN)}$
32: while $fn < MN$ do
33:     $G \leftarrow$ FILTER UPDATE $<< k, k >> (S)$
34:     $n \leftarrow n + 1$
35:     $L \leftarrow$ CONVOLUTION $<< k, k >> (Y, G)$
36:     $\Theta \leftarrow$ THRESHOLD $<< M, N >> (Y)$
37:     while $Q = Y$ do
38:         $Q \leftarrow Y$
39:         $Y \leftarrow$ PULSES $<< M, N >> (S, Y, L)$
40:         $L \leftarrow$ CONVOLUTION $<< k, k >> (Y, G)$
41:     end while
42:     $fn \leftarrow fn + \sum_{i\leftarrow 0}^{MN} Y_i$
43:     $T \leftarrow$ TIME MATRIX $<< M, N >> (T, Y, n)$
44: end while
45: return $T$
```

## 4. Experimentation

The model was evaluated on a Nvidia Jetson Nano embedded card with 4 ARM @ 1.43 GHz Cores, 4GB RAM and 128 GPU Cores, to validate its viability for an embedded system. The source code and some experiments of this implementation can be accessed in the public repository of the project (Luna, 2022). To validate the segmentation, specialized metrics in the segmentation detailed in Section 4.2 were used and it was evaluated on the images detailed below.

### 4.1. Data

Two databases were used: AigleRN (Amhaz et al., 2016), consisting of a compendium of 5 pavement crack databases, including 132 samples with hand-created ground truth and 149 unlabeled samples. All images are in 8-bit ppm format and range in size from $311 \times 462$ to $991 \times 462$. Figure 4 shows two samples.

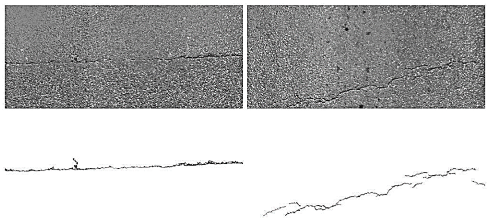

Figure 4. AigleRN samples with ground truth.

CrackForest (Shi et al., 2016) consists of 155 images with ground truth, including a. seg format file with the segmentation performed by its method (Shi et al., 2016), some images include urban scene objects such as sewers and cones. Each image is in 8-bit format of size $480 \times 320$. Figure 5 shows three examples from this database with their ground truth. In the most complex cases, the reference is shown as a stain indicating the crack area.

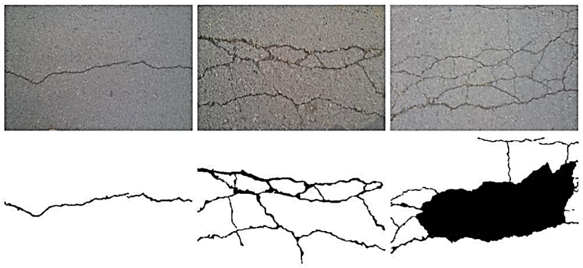

Figure 5. CrackForest samples with ground truth.

## 4.2. Metrics

To evaluate the intersection between the obtained and expected results, metrics derived from the confusion matrix precision, recall and F-score are calculated. The maximum signal to noise ratio (PSRN) (Setiadi, 2021) was used as the second metric, it defines the relationship between the maximum possible intensity of an image and the noise that affects it. To calculate, is necessary the mean squared error (MSE) and the maximum intensity, higher values represent better encoding.

$$PSNR = 10 \cdot log_{10}\left(\frac{max(x)^2}{MSE}\right) \tag{5}$$

The third metric used is the Structural Similarity Index (SSIM) (Setiadi, 2021), is a quality metric used to measure the similarity between two images.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y+C1)(2\sigma_{xy}+C2)}{(\mu_x^2+\mu_y^2+C1)(\sigma_x^2+\sigma_y^2+C2)} \tag{6}$$

where $\mu_x$ and $\mu_y$ are the means of $x$ and $y$ respectively, $\sigma_x^2$ and $\sigma_y^2$ the variances, $\sigma_{xy}$ covariance between $x$ y $y$, y $C1 = (k_1, L)^2$ , $C2 = (k_2, L)^2$ are variables to stabilize the division, $L$ is the range of pixel intensity and the constants $k_1 = 0.01$ and $k_2 = 0.03$.

## 5. Results

As a first observation, the proposed model shows an iterations reduction in the PCNN algorithm. This is not obtained by parallelization, but rather because the adaptive Gabor filter is more suitable than the Gaussian filter to extract textures, allowing faster convergence. Figure 6 shows the average iterations in the experimentation.

As seen in Figure 6, the proposed model takes 1.6% of iterations than the original PCNN. In addition to the acceleration of the convergence of the algorithm, the quality of the crack segmentation is improved. Figure 7 shows the segmentation of an image from the CrackForest database.

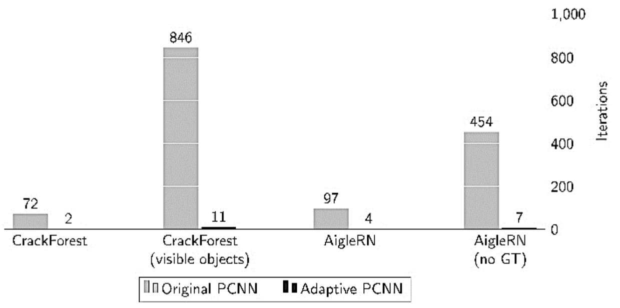

Figure 6. Iterations by the original and proposed PCNN.

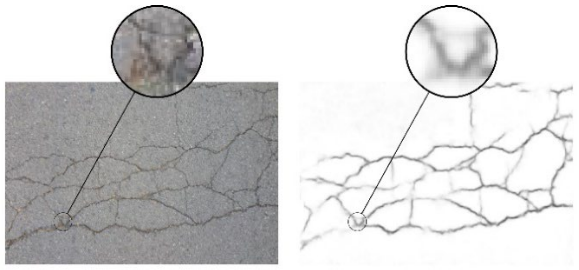

Figure 7. Output segmented image.

Segmentation showed that it is not affected by variations in the pavement texture, even in small regions and considering the filter size and the wavelength $\lambda$.

To compare, some implementations were replicated: the Otsu threshold proposed by Akagic et al. (2018), triple Canny threshold (Wang et al., 2018), optimized version of Gabor filter (Khan et al., 2016) and neural network DeepCrack (Liu et al., 2019). All methods were designed for pavement cracks detection. As a first comparison, the convergence time on the embedded card was recorded. For the proposed model, its sequential and parallel implementation was evaluated separately to show the improvement without hardware acceleration. Figure 8 shows the graph of the time consumed by each algorithm.

In this comparison, it should be noted that the methods do not require training except for the DeepCrack network, for this the pre-trained and parallelized GPU model was used, so training is not considered and presents hardware advantages. Even so, the parallel implementation of the proposed model requires $\approx 45$ ms less for each image. Even so, the metrics indicated that the proposed model presents the best quality of segmentation in most cases. In summary, Table 1 shows the quantitative comparison from the metrics obtained by the methods, as well as graphically in Figure 9 a qualitative comparison is shown.

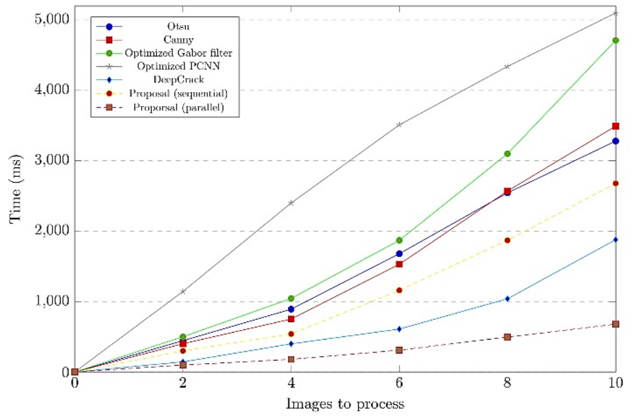

Figure 8. Convergence time graph of the evaluated methods.

Finally, in some experiments segmentation behaviors by regions are presented. As can be seen in Figure 10, the proposed model generates two regions delimited by the detection of a transverse fracture in the image.

This result is obtained by the intensity variations between both regions of the image. To solve this, the model can be complemented with a filter that matches the intensities. This also shows that the proposed model is not only functional for detection but can also be applied in non-binary segmentation by correctly implementing the adjustment of the parameters. This idea can be raised as future work.

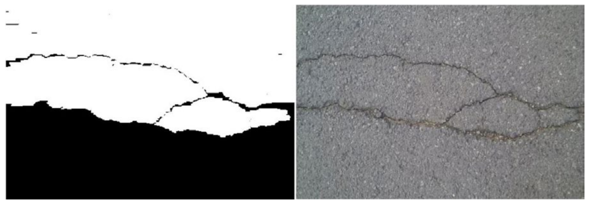

Figure 10. Segmentation.

Table 1. Summary of segmentation metrics obtained from the experimentation.

| Method | Precision | Recall | F-score | SSIM | PSNR | MSE |
|---|---|---|---|---|---|---|
| CrackForest | | | | | | |
| Canny | 0.126 | 0.238 | 0.129 | 0.573 | *51.200* | 0.068 |
| Otsu | *0.050* | 0.650 | *0.085* | *0.062* | 54.347 | *0.245* |
| Optimized Gabor | 0.375 | *0.042* | 0.088 | 0.829 | 56.170 | 0.023 |
| Optimized PCNN | 0.431 | 0.192 | 0.232 | 0.879 | 51.262 | 0.058 |
| CNN DeepCrack | **0.911** | 0.882 | 0.896 | 0.920 | **58.599** | **0.018** |
| Proposed method | **0.911** | **0.897** | **0.903** | **0.946** | 58.020 | **0.018** |
| AigleRN | | | | | | |
| Canny | *0.006* | 0.263 | *0.011* | *0.126* | 54.284 | *0.414* |
| Otsu | 0.013 | 0.391 | 0.025 | 0.325 | *51.992* | 0.294 |
| Optimized Gabor | 0.026 | *0.257* | 0.044 | 0.526 | 57.782 | 0.281 |
| Optimized PCNN | 0.043 | 0.286 | 0.079 | 0.336 | 56.899 | 0.151 |
| CNN DeepCrack | 0.855 | 0.792 | 0.822 | 0.896 | 60.154 | 0.105 |

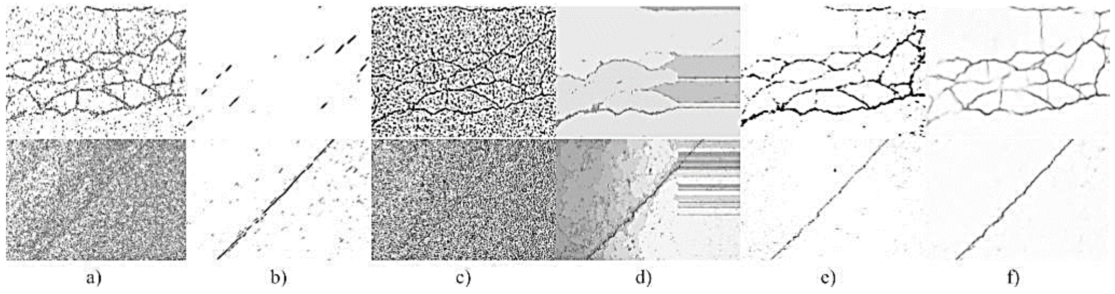

Figure 9. Images segmented by a) Canny, b) Gabor filter, c) Otsu, d) PCNN, e) DeepCrack f) Proposed PCNN.

An adaptive pulse-coupled neural network based on noise estimation was proposed. The model can adapt to the type of image to obtain the best segmentation by filtering textures without prior training. Compared to the original model, it reduces to 2% of the iterations it would take. The algorithm was parallelized on the GPU, reducing the response time in an embedded system from $x^n$ to $x$. It is capable of processing 10 images in $\approx 0.8$ seconds, which makes it feasible to implement in a real-time system.

Classic and recent methods specialized in the pavement crack segmentation were evaluated, the proposed method is 60% faster in a simple implementation and +90% parallelized, preserving the quality of the segmentation, which in most cases is superior. As future work, it is proposed to create the system in real time embedded in a vehicle.

## Conflict of interest

The authors have no conflict of interest to declare.

## Acknowledgements

The authors thank Consejo Nacional de Ciencia y Tecnología and Tecnológico Nacional de México/Centro Nacional de Investigación y Desarrollo Tecnológico for the support of this research.

## Funding

The authors received no specific funding for this work.

## References

Akagic, A., Buza, E., Omanovic, S., & Karabegovic, A. (2018). Pavement crack detection using Otsu thresholding for image segmentation. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (pp. 1092-1097). IEEE. https://doi.org/10.23919/MIPRO.2018.8400199

Amhaz, R., Chambon, S., Idier, J., & Baltazart, V. (2016a). Automatic crack detection on two-dimensional pavement images: An algorithm based on minimal path selection. *IEEE Transactions on Intelligent Transportation Systems*, *17*(10), 2718-2729. https://doi.org/10.1109/TITS.2015.2477675

Amhaz, R., Chambon, S., Idier, J., & Baltazart, V. (2016b). Automatic crack detection on 2D pavement images: An algorithm based on minimal path selection. online; Accessed 30 9 2021 https://www.irit.fr/~Sylvie.Chambon/Crack_Detection_Database.html

Cubero-Fernandez, A., Rodriguez-Lozano, F. J., Villatoro, R., Olivares, J., & Palomares, J. M. (2017). Efficient pavement crack detection and classification. *EURASIP Journal on Image and Video Processing*, *2017*, 1-11. https://doi.org/10.1186/s13640-017-0187-0

Gopalakrishnan, K., Khaitan, S. K., Choudhary, A., & Agrawal, A. (2017). Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection. *Construction and building materials*, *157*, 322-330. https://doi.org/10.1016/j.conbuildmat.2017.09.110

Khan, H. A., Salman, M., Hussain, S., & Khurshid, K. (2016). Automation of optimized gabor filter parameter selection for road cracks detection. *International Journal of Advanced Computer Science and Applications*, *7*(3). https://u-bourgogne.hal.science/hal-01431337

Kim, D. G., & Shamsi, Z. H. (2018). Enhanced residual noise estimation of low rank approximation for image denoising. *Neurocomputing*, *293*, 1-11. https://doi.org/10.1016/j.neucom.2018.02.063

Lau, S. L., Chong, E. K., Yang, X., & Wang, X. (2020). Automated pavement crack segmentation using u-net-based convolutional neural network. *Ieee Access*, *8*, 114892-114899. https://doi.org/10.1109/ACCESS.2020.3003638

Li, S., Cao, Y., & Cai, H. (2017). Automatic pavement-crack detection and segmentation based on steerable matched filtering and an active contour model. *Journal of Computing in Civil Engineering*, *31*(5), 04017045.
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000695

Li, Z., Xu, G., Cheng, Y., Wang, Z., & Wu, Q. (2020). Pavement crack detection using progressive curvilinear structure anisotropy filtering and adaptive graph-cuts. *IEEE Access*, *8*, 65020-65034.
https://doi.org/10.1109/ACCESS.2020.2985216

Lian, J., Yang, Z., Liu, J., Sun, W., Zheng, L., Du, X., ... & Ma, Y. (2021). An overview of image segmentation based on pulse-coupled neural network. *Archives of Computational Methods in Engineering*, *28*, 387-403.
https://doi.org/10.1007/s11831-019-09381-5

Liu, Y., Yao, J., Lu, X., Xie, R., & Li, L. (2019). DeepCrack: A deep hierarchical feature learning architecture for crack segmentation. *Neurocomputing*, *338*, 139-153.
https://doi.org/10.1016/j.neucom.2019.01.036

Luna, A. (2022) Pcnn with adaptive gabor filter.
https://github.com/TonnyLuna/PCNN_AdaptiveGabor_segmentation

Mukherjee, D., & Das, A. (2021). Gabor filter based automated enhancement of brain tumors. In *Advances in Medical Physics and Healthcare Engineering: Proceedings of AMPHE 2020* (pp. 71-80). Springer Singapore.
https://doi.org/10.1007/978-981-33-6915-3_8

Nie, R., Cao, J., Zhou, D., & Qian, W. (2020). Multi-source information exchange encoding with PCNN for medical image fusion. *IEEE Transactions on Circuits and Systems for Video Technology*, *31*(3), 986-1000.
https://doi.org/10.1109/TCSVT.2020.2998696

Pauly, L., Hogg, D., Fuentes, R., & Peel, H. (2017). Deeper networks for pavement crack detection. In *Proceedings of the 34th ISARC* (pp. 479-485). IAARC.
https://doi.org/10.22260/isarc2017/0066

Setiadi, D. R. I. M. (2021). PSNR vs SSIM: imperceptibility quality assessment for image steganography. *Multimedia Tools and Applications*, *80*(6), 8423-8444.
https://doi.org/10.1007/s11042-020-10035-z

Shao, C., Chen, Y., Xu, F., & Wang, S. (2019). A kind of pavement crack detection method based on digital image processing. In *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)* (pp. 397-401). IEEE.
https://doi.org/10.1109/IAEAC47372.2019.8997810

Shi, Y., Cui, L., Qi, Z., Meng, F., & Chen, Z. (2016). Automatic road crack detection using random structured forests. *IEEE Transactions on Intelligent Transportation Systems*, *17*(12), 3434-3445.
https://doi.org/10.1109/TITS.2016.2552248

Wang, G., Peter, W. T., & Yuan, M. (2018). Automatic internal crack detection from a sequence of infrared images with a triple-threshold Canny edge detector. *Measurement Science and Technology*, *29*(2), 025403.
https://doi.org/10.1088/1361-6501/aa9857

Wang, Z., Ma, Y., Cheng, F., & Yang, L. (2010). Review of pulse-coupled neural networks. *Image and vision computing*, *28*(1), 5-13.
https://doi.org/10.1016/j.imavis.2009.06.007

Yang, F., Zhang, L., Yu, S., Prokhorov, D., Mei, X., & Ling, H. (2019). Feature pyramid and hierarchical boosting network for pavement crack detection. *IEEE Transactions on Intelligent Transportation Systems*, *21*(4), 1525-1535.
https://doi.org/10.1109/TITS.2019.291059

Zhan, K., Shi, J., Wang, H., Xie, Y., & Li, Q. (2017). Computational mechanisms of pulse-coupled neural networks: a comprehensive review. *Archives of Computational Methods in Engineering*, *24*, 573-588.
https://doi.org/10.1007/s11831-016-9182-3

Zhang, A., Wang, K. C., Fei, Y., Liu, Y., Tao, S., Chen, C., ... & Li, B. (2018). Deep learning–based fully automated pavement crack detection on 3D asphalt surfaces with an improved CrackNet. *Journal of Computing in Civil Engineering*, *32*(5), 04018041.
https://doi.org/10.1061/(ASCE)CP.1943-5487.00007

Zhang, L., Yang, F., Zhang, Y. D., & Zhu, Y. J. (2016). Road crack detection using deep convolutional neural network. In *2016 IEEE international conference on image processing (ICIP)* (pp. 3708-3712). IEEE.
https://doi.org/10.1109/ICIP.2016.7533052