Brumariu Cosmin-Nicusor

Optimizarea metodei de binarizare Otsu folosind extensii DSP pe Raspberry Pi Pico 2 W

Conducător științific: Șef lucr. dr. ing. KERTÉSZ Csaba Zoltán

**Abstract**

This paper presents the implementation and optimization of the Otsu image binarization method on the Raspberry Pi Pico 2 W platform. The Otsu algorithm automatically determines the optimal threshold for separating background and foreground based on inter-class variance. Optimization was achieved using the CMSIS-DSP library, which leverages the DSP extensions of the ARM Cortex-M33 core. The optimized version was benchmarked against a non-optimized implementation, evaluating execution time and image quality metrics such as PSNR and SSIM. Results demonstrate a ~39× speed increase without compromising output quality, showcasing the potential of low-cost microcontrollers for real-time image processing applications.

# 1. Rezumat

Această lucrare propune implementarea și optimizarea metodei de binarizare Otsu pe platforma embedded Raspberry Pi Pico 2 W. Algoritmul determină automat pragul optim de separare a fundalului și obiectelor într-o imagine, pe baza varianței inter-clasă. Optimizarea a fost realizată prin utilizarea funcțiilor din biblioteca CMSIS-DSP, care valorifică extensiile de procesare semnal ale nucleului ARM Cortex-M33. Performanța metodei este comparată cu o implementare neoptimizată, analizând timpul de execuție și calitatea binarizării prin metrici PSNR și SSIM. Proiectul evidențiază potențialul microcontrolerelor low-cost pentru procesare eficientă de imagini în timp real.

# 2. Introducere

Binarizarea unei imagini este un pas esențial în procesarea imaginilor, folosit pentru extragerea contururilor sau a trăsăturilor importante dintr-un cadru vizual. Metoda Otsu este una dintre cele mai cunoscute tehnici automate de alegere a unui prag optim. Aceasta a fost introdusă de Nobuyuki Otsu în 1979 [1] și este utilizată pe scară largă în prelucrarea imaginilor. În acest proiect, algoritmul a fost implementat pe Raspberry Pi Pico 2 W, un microcontroler cu două nuclee ARM Cortex-M33 și extensii DSP. Proiectul compară două implementări, una clasică și una optimizată prin CMSIS-DSP, și analizează avantajele oferite de procesarea pe instrucțiuni specializate.

# 3. Arhitectură hardware și software

Raspberry Pi Pico 2 W este echipat cu microcontrolerul RP2350, care conține două nuclee ARM Cortex-M33 tactate la 150 MHz și suportă extensii DSP și FPU [3]. Proiectul a fost implementat în limbaj C++, folosind toolchain-ul standard arm-none-eabi-gcc și biblioteca CMSIS-DSP dezvoltată de Arm Ltd. [2]. Codul implementează generarea unei imagini artificiale, calculul histogramei intensităților, aplicarea algoritmului Otsu (optimizat și neoptimizat), și evaluarea PSNR/SSIM.

# 4. Algoritmul Otsu și optimizarea sa

Metoda Otsu determină pragul de binarizare optim prin maximizarea varianței între două clase: fundal și obiect [1]. Pentru reducerea timpului de execuție, s-au folosit funcții din biblioteca CMSIS-DSP: arm\_power\_f32 pentru MSE, arm\_mean\_f32 și arm\_var\_f32 pentru SSIM [2]. Acestea valorifică capacitățile DSP ale procesorului ARM Cortex-M33 [5].

# 5. Rezultate experimentale

Imaginea utilizată pentru prima executie este de 64x64 pixeli, cu o zonă întunecată (intensitate 50) și una luminoasă (intensitate 200).

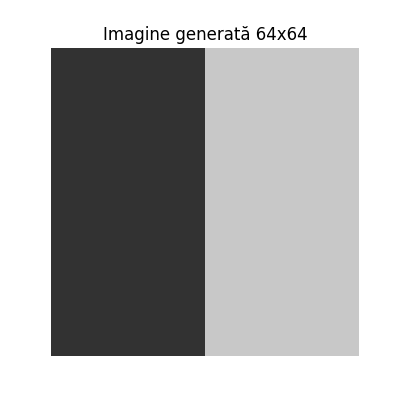


Figura 1 – Imagine 64x64 cu două benzi de intensitate.

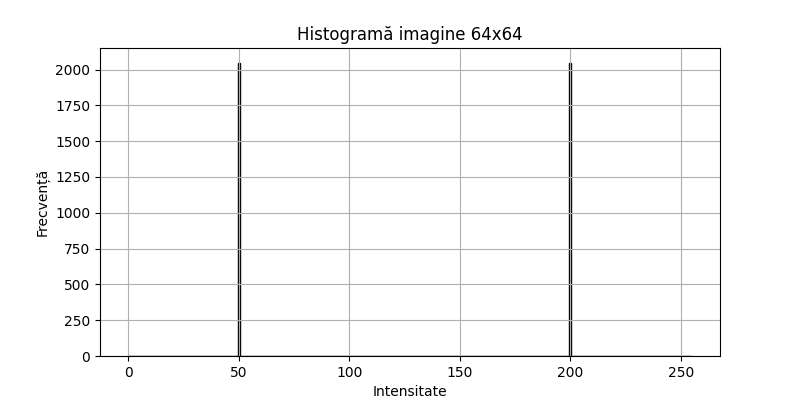


Figura 2 – Histogramă a imaginii generate.

Prin aplicarea metodei Otsu și optimizarea cu CMSIS-DSP, s-au obținut rezultate comparabile ca PSNR și SSIM față de versiunea brută, dar cu timp de execuție semnificativ mai mic.

Pentru a evalua performanța reală a algoritmului, metoda Otsu a fost aplicată pe 10 de imagini diferite, generate cu diverse niveluri de zgomot și contrast. Pentru fiecare imagine s-au măsurat timpul de execuție, PSNR și SSIM, comparând implementarea optimizată (cu CMSIS-DSP) cu cea neoptimizată (clasică, pe bucle brute). Tabelul de mai jos prezintă valorile medii obținute:

Tabel 1 – Medii obținute pe 10 de imagini cu zgomot variabil.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Versiune | Timp mediu execuție (µs) | PSNR mediu (dB) | SSIM mediu | Raport viteză (x) | Prag mediu Otsu |
| Optimizată | 642.78 | 18.89 | 0.796 | 38.86 | |  | | --- | | 62.65 |  |  | | --- | |  | |
| Neoptimizată | 24975.49 | 18.89 | 0.796 | 1.0 | |  | | --- | | 62.65 |  |  | | --- | |  | |

Rezultatele arată că varianta optimizată este în medie de aproximativ 38.9 ori mai rapidă decât cea neoptimizată, menținând aceleași niveluri de calitate (PSNR și SSIM).

# 6. Concluzii și perspective

Metoda Otsu poate fi eficient optimizată pe microcontrolere low-cost utilizând CMSIS-DSP. Acest tip de procesare este aplicabil în recunoaștere vizuală, sisteme embedded de control sau aplicații IoT. Pentru viitor, se propune extinderea proiectului prin utilizarea datelor reale din senzor, integrarea cu cameră sau rularea pe două nuclee.

# 7. Bibliografie

[1] N. Otsu, “A Threshold Selection Method from Gray-Level Histograms,” IEEE Transactions on Systems, Man, and Cybernetics, 1979.

[2] Arm Ltd., “CMSIS-DSP Software Library Documentation,” [Online]. Available: https://developer.arm.com

[3] Raspberry Pi Ltd., “Raspberry Pi Pico 2 W Datasheet,” [Online]. Available: https://datasheets.raspberrypi.com/pico2w/pico2w-datasheet.pdf

[4] S. W. Smith, The Scientist and Engineer’s Guide to Digital Signal Processing. California Technical Publishing, 1997. [Online]. Available: https://www.dspguide.com

[5] ARM, “Cortex-M33 Technical Reference Manual,” [Online]. Available: https://developer.arm.com/documentation/100235