Perfilado de rendimiento de FPS para múltiples arquitecturas computacionales usando el algoritmo de reducción de neblina DCP

FPS performance profiling for multiple computational architectures using the DCP dehazing algorithm

Allan Francisco Navarro-Brenes[1], Luis Alberto Chavarría-Zamora[2]

Fecha de recepción: 30 de abril de 2021
Fecha de aprobación: 6 de agosto de 2021

Navarro-Brenes, A.F; Chavarría-Zamora, L.A. Perfilado de rendimiento de FPS para múltiples arquitecturas computacionales usando el algoritmo de reducción de neblina DCP. *Tecnología en Marcha*. Vol. 35-3. Julio-Setiembre 2022. Pág. 73-81.

https://doi.org/10.18845/tm.v35i3.5718

1 Instituto Tecnológico de Costa Rica. Costa Rica.
Correo electrónico: allannabre01@estudiantec.cr
https://orcid.org/0000-0002-4878-7223
2 Instituto Tecnológico de Costa Rica. Costa Rica.
Correo electrónico: lachavarria@itcr.ac.cr
https://orcid.org/0000-0002-2510-5680

Palabras clave

Benchmark de FPS; procesamiento de imágenes; procesamiento en tiempo real; sistemas empotrados; arquitectura de computadores; reducción de neblina; restauración de imagen.

Resumen

Este documento presenta una prueba de rendimiento creada para evaluar el desempeño de diferentes plataformas en la ejecución de un algoritmo de reducción de niebla basado en el "Dark Channel Prior" (DCP) [1]. El parámetro utilizado para la evaluación fue el número de cuadros por segundo (FPS, por sus siglas en inglés) que el dispositivo es capaz de procesar. Con esta herramienta se logra determinar aquellas arquitecturas que son aptas para ejecutar el algoritmo en tiempo real.

El ambiente de pruebas se ejecutó en cuatro plataformas, un Google Pixel 3a, una Raspberry Pi 3B+, una GPU de NVIDIA y un procesador Intel x86. Se usaron los siguientes kits de desarrollo de software (SDK, por sus siglas en inglés) según la plataforma: Android NDK, Yocto Poky, CUDA y la cadena de herramientas GCC.

La herramienta permitió recopilar, para cada plataforma, los FPS para distintos tamaños de imagen, con estos resultados se pueden escoger la arquitectura más idónea según el área de implementación (e.g., bajo consumo o HPC).

Keywords

FPS benchmark; image processing; real-time processing; embedded systems; computer architecture; image dehazing; image restoration.

Abstract

In this document we present a benchmark to evaluate the performance of different platforms in the execution of a dehazing algorithm based on the Dark Channel Prior (DCP) [1]. The parameter used for the evaluation was the number of frames per second (FPS) that the device was able to process. This tool allows to determine which architectures can execute the algorithm in real time.

The testing environment was executing in four platforms, a Google Pixel 3a, a Raspberry Pi 3B+, a GPU by NVIDIA, and an Intel x86 processor. The following software development kits (SDK's) where used for each of the platforms: Android NDK, Yocto Poky, CUDA, and the GCC toolchain.

The tool allowed us to collect, for each platform, the FPS for different image sizes, these results allow the selection of an ideal architecture depending on a specific application (e.g., low power, HPC).

Introducción

La niebla ha tenido un impacto significativo en actividades de la humanidad desde las edades tempranas y especialmente en las últimas décadas como consecuencia del aumento de la circulación aérea, marina, y el tráfico terrestre [2]. Las consecuencias de este fenómeno han generado pérdidas económicas y humanas comparables con los efectos de otros eventos meteorológicos como tornados o huracanes [2].

Un ejemplo de este impacto se refleja en las cifras de accidentes relacionados con la niebla, como lo indica la Administración Federal de Carreteras (FHWA) de EE. UU, la niebla causó el 2% de las muertes en accidentes automovilísticos entre 2007 y 2016. Además, del total de

muertes relacionadas con el clima, un 9% se debió a este fenómeno [3]. El tener un sistema informático que ayude a reducir los efectos negativos de la niebla en imágenes digitales puede ser beneficioso en el sector transporte con posibles aplicaciones en otras áreas como: sistemas de seguridad, industria fotográfica y cinematográfica.

Para implementar este sistema, se debe realizar un análisis previo de las plataformas candidatas en función de su capacidad para implementar un algoritmo de reducción de niebla en tiempo real. En este documento se proporciona una forma sistemática de evaluar diferentes plataformas y recopilar los datos generados durante tiempo de ejecución con el fin de encontrar la plataforma idónea según el área de aplicación. Adicionalmente, se desarrolla una herramienta compilable para diferentes plataformas dado un mismo código fuente capaz de transmitir datos en tiempo real a una computadora para visualizar los resultados.

Este documento está organizado de la siguiente manera: en el marco teórico se presentan los conceptos fundamentales del algoritmo DCP y otros trabajos similares al desarrollado. En la sección "Materiales y Métodos", se explica la metodología empleada para desarrollar la herramienta y el diseño utilizado. En la sección de resultados, se muestra la información captada por las distintas plataformas y por último se presentan las conclusiones y recomendaciones.

Marco teórico

Algoritmo DCP

El algoritmo utilizado para realizar el procesamiento de las imágenes es el "*Dark Channel Prior*" (DCP) [1] [4]. Este algoritmo se basa en la premisa que, para imágenes de exteriores sin niebla, al menos un canal de color tiene algunos píxeles cuya intensidad es cercana a cero para cuadros que no contengan cielo [4]. El modelo de formación de niebla utilizado en ese trabajo es el descrito por la ecuación (1).

$$\mathbf{I}(x) = \mathbf{J}(x)t(x) + \mathbf{A}\big(1 - t(x)\big) \tag{1}$$

donde $\mathbf{I}(x)$ es la intensidad de la imagen de entrada, $\mathbf{J}(x)$ es la luminosidad de la escena, $\mathbf{A}$ es la luz atmosférica global y $t(x)$ es la transmisividad del medio, este modelo resulta en un sistema de ecuaciones indeterminado, con 3N variables conocidas y 4N + 3 incógnitas [4]. Para lograr resolver el sistema, a partir de una imagen arbitraria **Y**, se determina su canal oscuro (DCP) o J^{dark} como se muestra en [4].

$$J^{\text{dark}}(x) = \min_{y \in \Omega(x)} \left(\min_{c \in \{r,g,b\}} J^c(\boldsymbol{Y}) \right) \tag{2}$$

Donde J^{dark} es una imagen en escala de grises, se escoge el 0.1% de pixeles con mayor intensidad. Utilizando la posición de los pixeles seleccionados, se selecciona el color que tenga la mayor intensidad en la imagen original, este color corresponde a $\mathbf{A}$. Luego se calcula una aproximación de $t(x)$ llamada $\tilde{t}(\boldsymbol{x})$, de la siguiente manera:

$$\tilde{t}(\boldsymbol{x}) = 1 - \min_{y \in \Omega(\boldsymbol{x})} \left(\min_{c} \frac{I^c(\boldsymbol{y})}{A^c} \right) \tag{3}$$

Este cálculo es en realidad el DCP de $\mathbf{I}$ normalizado por **A**, como se menciona en [4]. Para reconstruir la imagen deseada $\boldsymbol{J}$, se utiliza la siguiente fórmula:

$$\boldsymbol{J}(\boldsymbol{x}) = \frac{\boldsymbol{I}(\boldsymbol{x}) - \boldsymbol{A}}{\max(t(\boldsymbol{x}), t_0)} + \boldsymbol{A} \quad (4)$$

Múltiples métodos propuestos para reducir niebla en imágenes son en realidad variaciones u optimizaciones del DCP (por ejemplo [1], [4]). Además, existen trabajos para aplicaciones agrícolas en imágenes aéreas, donde el DCP muestra una mejora de calidad en imágenes para fotogrametría utilizando un brillo modificado en los espacios HSB y HSL antes de la aplicación del algoritmo DCP [5].

Una revisión extensiva de distintos métodos de procesamiento menciona al DCP como el concepto preferido por los investigadores según [6]. Como este algoritmo es usado extensivamente en múltiples trabajos de estudio de neblina, se propone el estudio de múltiples plataformas ejecutando el DCP.

Trabajos relacionados

Hay varios trabajos relacionados con la reducción de neblina de imágenes en sistemas empotrados de tiempo real. Por ejemplo, en [7] se muestra un dispositivo integrado de bajo costo para reducción de neblina en imágenes [7]. Otro algoritmo propuesto basado en DCP se discute en [8] donde se usa una imagen de entrada reducida para calcular J^{dark} y , utilizando un tamaño de ventana de un píxel, lo cual permite reducir en gran medida el tiempo de ejecución del algoritmo sin cambios significativos en la calidad en los resultados.

En cuanto a pruebas de desempeño (*benchmarks*), en [9] se evalúan diferentes algoritmos, mencionando al DCP como uno de los métodos que mejores resultados lograron en un ambiente submarino. En [10] se realizó un *benchmark* para dos plataformas embebidas diferentes, el cual tiene similitudes con los objetivos de este estudio. Sin embargo, este *benchmark* se enfocan en arquitecturas especializadas mientras que la intención detrás de este trabajo es tratar con una amplia gama de plataformas, analizando más dispositivos y produciendo una prueba que se puede ejecutar sin cambios significativos en el código fuente, al proveer una funcionalidad para incluir otras arquitecturas no contempladas en este trabajo.

Materiales y métodos (metodología)

En esta sección, se explica la metodología desarrollada en el trabajo propuesto. Primero, se inició con la selección de las plataformas a comparar, se escogieron cuatro plataformas, las cuales son mostradas en el cuadro 1. Además, la herramienta puede ser compilada para cualquier otra plataforma, agregado a la pantalla de visualización siempre y cuando cumpla con el formato definido.

Cuadro 1. Plataformas utilizadas para analizar el desempeño de el algoritmo DCP.

Plataforma	Núcleos de procesamiento	RAM	Sistema Operativo
Broadcom BCM2837B0 (Raspberry Pi 3B+)	4 núcleos Cortex-A53 (ARMv8) 64-bit @ 1.4GHz	1GB LPDDR2	Yocto Poky 3.1
Snapdragon 670 (Google Pixel 3a smartphone)	8 núcleos 2@2.0 GHz 360 Gold y 6@1.7 GHz Kryo 360 Silver	4GB LPDDR4x	Android 11
GeForce GTX 1050 Ti [3]	768 núcleos CUDA	4GB GDDR5	Ubuntu 20.04
Intel Core i7-8750H	6 núcleos (12 hilos) x86 64-bit @ 2.2 GHz	32GB DDR4	Ubuntu 20.04

Diseño y herramientas utilizadas

La figura 1 muestra, en alto nivel, la arquitectura utilizada para la herramienta, donde se tienen múltiples nodos los cuales se comunican mediante el protocolo de control de transmisión (TCP) con la máquina del usuario, la cual es la que almacena la información y muestra los gráficos en pantalla.

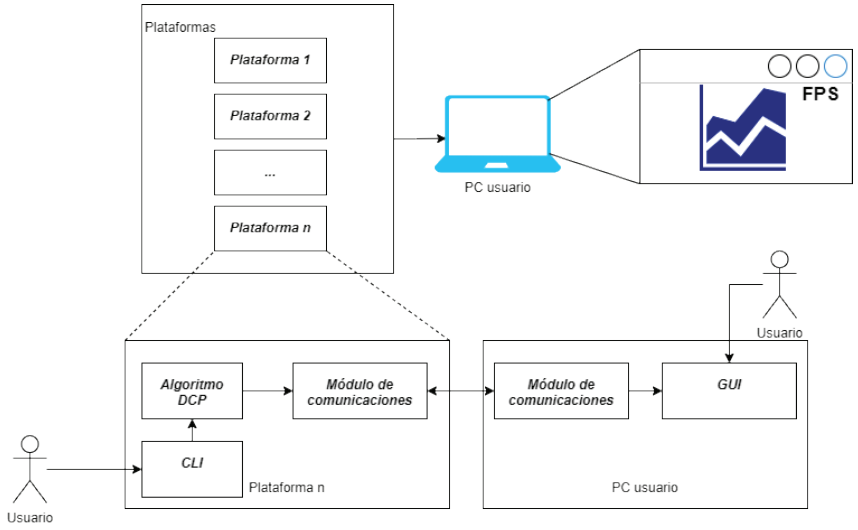

Figura 1. Arquitectura de alto nivel de la herramienta propuesta en este trabajo.

Las principales tecnologías y herramientas utilizadas en este documento para desarrollar el sistema fueron las siguientes:

- Lenguaje de programación: Se utilizó C++ tanto para el *backend* como para la interfaz gráfica.
- Compilador: GCC 9.3.0.
- Interfaz de usuario: Qt 5, para la visualización de los datos.

3 Sistema heterogéneo CPU/GPU junto con el Intel Core i7 8750H

- Programación multi hilo: Para las plataformas que no utilizan GPU, se utilizó Open MP versión 4.3, la cual forma parte del compilador GCC.
- Programación con GPU: CUDA 11.1, esta herramienta provee una interfaz sencilla para ejecutar programas en entornos heterogéneos (GPU+CPU).
- Procesamiento de Imágenes: OpenCV 4.4, para simplificar la codificación y decodificación, así como manipulación básica de imágenes.
- Protocolo de comunicación: Se utilizó TCP/IP mediante "sockets" de Linux para recopilar la información procedente de las plataformas y centralizarla en la computadora del usuario.

Resultados

La figura 2 muestra una ventana de la herramienta con los resultados de la ejecución en las cuatro plataformas consideradas procesando una imagen con una resolución QHD (960x540 píxeles). Todas las plataformas lograron procesar a esta resolución a una tasa de al menos 20 FPS, a esta velocidad aún se puede visualizar un video de manera fluida.

Figura 2. Ejecución de la herramienta con cuatro plataformas procesando una imagen en QHD (resolución) con el algoritmo DCP.

La interfaz gráfica es capaz de agregar dinámicamente las plataformas deseadas para ocupar menos espacio de ventana como se muestra en la figura 3. En esta imagen se observa como para una GPU CUDA se obtiene un rendimiento de 20.3919 FPS procesando una imagen UHD.

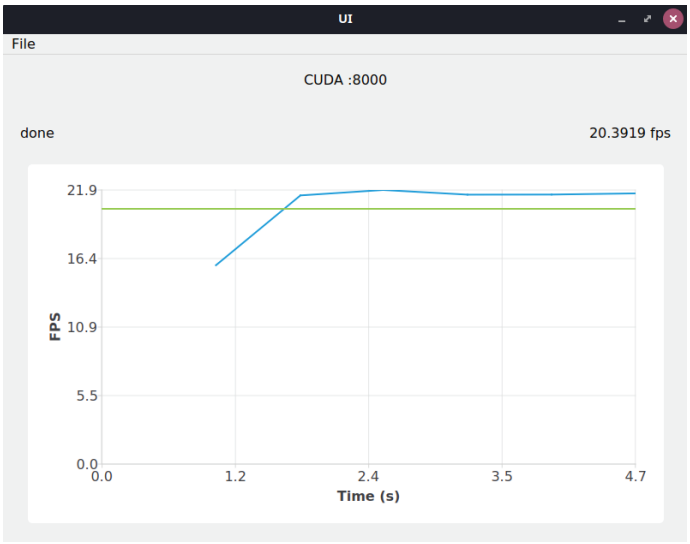

Figura 3. GPU ejecutando el algoritmo DCP para una imagen UHD.

En la figura 4 se muestra el procesamiento del algoritmo DCP para reducir la neblina. Hacia la izquierda de la imagen se observan las entradas, a la derecha, se muestra la misma imagen procesada por las plataformas.

Figura 4. Imágenes procesadas por las plataformas. Imagen libre de derechos de autor tomada de pexels.com.

Por otra parte, en la figura 5. Muestra de fotogramas de uno de los videos procesados. Video libre de derechos de autor tomado de pexels.com.figura 5, se muestra un extracto de tres fotogramas de uno de los videos procesados por la herramienta.

Figura 5. Muestra de fotogramas de uno de los videos procesados. Video libre de derechos de autor tomado de pexels.com.

Conclusiones y recomendaciones

En este artículo se presentó una manera sistemática de analizar el comportamiento de distintas plataformas en la ejecución del algoritmo DCP, teniéndose que el tamaño máximo de imagen procesado en el ambiente de pruebas fue 4k (UHD), el cual pudo ser procesado por la GPU a una velocidad promedio de aproximadamente 20 FPS, mientras que la Raspberry Pi 3B+ pudo manejar un video QHD a esa misma velocidad.

Como se observa en la figura 2, las plataformas Snapdragon e Intel tuvieron un desempeño similar en la ejecución del algoritmo. Es importante recalcar que la segunda es la PC del usuario en la cual se ejecuta la GUI junto con otros procesos no relacionados directamente con el algoritmo DCP. Es por esto por lo que se considera que el rendimiento de la plataforma debería aumentar si se ejecuta en un entorno mínimo.

La figura 3. muestra los resultados de CUDA para el video de mayor resolución considerado con un tamaño de con una resolución UHD, a una tasa de aproximadamente 20 FPS, lo cual se puede visualizar continuamente. El uso de una GPU para el paralelismo a nivel de datos demuestra ser el método de mejor rendimiento en comparación con el uso de programación multihilo en la CPU, esto es de esperar debido a que el algoritmo DCP es altamente paralelizable, además, la plataforma CUDA utilizada cuenta con 768 núcleos. Este método es aproximadamente tres veces más rápido que la siguiente mejor solución que usa solo los 12 núcleos de la CPU x86 en la misma máquina.

La prueba de rendimiento permite la evaluación de múltiples plataformas ejecutando el algoritmo de depuración DCP utilizando un único código fuente. Además, se demostró la viabilidad de utilizar un sistema empotrado de pocos recursos como la Raspberry Pi con una resolución media de 960x540 píxeles logrando un rendimiento de hasta 20 FPS. Una GPU en un entorno heterogéneo es capaz de procesar calidades mucho más altas de hasta 4k conservando una buena velocidad de fotogramas.

Esta herramienta permite contar con información en tiempo real acerca de la ejecución del algoritmo DCP en distintas plataformas, con estos datos es posible escoger de una manera objetiva la plataforma adecuada dependiendo la aplicación objetivo, explorando potenciales soluciones a un reto ingenieril.

Se recomienda analizar el comportamiento del procesador x86 utilizando otro equipo distinto al que se usa para ejecutar el *benchmark*. Preferiblemente en un entorno mínimo mediante el uso de un sistema operativo personalizado como lo es Yocto. Con esto se evita que se impacte negativamente el desempeño del algoritmo DCP.

Agradecimientos

Un especial agradecimiento al PhD. Juan Pablo Soto Quirós, por la retroalimentación brindada.

Referencias

[1] K. He, J. Suny X. Tang, “Single image haze removal using dark channel prior,” in 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 1956–1963. DOI: 10.1109/CVPR.2009.5206515.

[2] I. Gultepe, R. Tardif, S. C. Michaelides, J. Cermak, A. Bott, J. Bendix, M. D. M¨uller, M. Pagowski, B. Hansen, B. Ellrod, W. Jacobs, G. Tothy S. G. Cober, “Fog research: A review of past achievements and future perspectives,” in Pure and Applied Geophysics, vol. 164, 2007, pp. 1121–1159. DOI: https://doi.org/10.1007/s00024-007-0211-x.

[3] How do weather events impact roads? [En línea]. Disponible: https://ops.fhwa.dot.gov/weather/q1 roadimpact.htm.

[4] K. He, J. Sun y X. Tang, “Single image haze removal using dark channel prior,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 12, pp. 2341–2353, 2011. DOI: 10.1109/TPAMI.2010.168.

[5] L. A. Chavarria-Zamora, S. Arriola-Valverdey R. Rimolo-Donadio, “Evaluation of fog reduction algorithms for photogrammetric applications in agriculture,” in 2018 IEEE International Work Conference on Bioinspired Intelligence (IWOBI), 2018, pp. 1–9. DOI: 10.1109/IWOBI.2018.8464186.

[6] G. Harish Babu and N. Venkatram, “A survey on analysis and implementation of state-of-the-art haze removal techniques,” Journal of Visual Communication and Image Representation, vol. 72, p. 102 912, 2020, ISSN: 1047-3203. DOI: https://doi.org/10.1016/j.jvcir.2020.102912. [En línea]. Disponible: http://www.sciencedirect.com/science/article/pii/S1047320320301504.

[7] T. Koga, A. Yasuda, S. Furukaway N. Suetake, “A simplified and fast dehazing processing suitable for embedded systems,” in 2018 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), 2018, pp. 492–497. DOI: 10.1109/ISPACS.2018.8923385.

[8] Y. Iwamoto, N. Hashimotoy Y. Chen, “Fast dark channel prior based haze removal from a single image,” in 2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2018, pp. 458–461. DOI:10.1109/FSKD.2018.8686854.

[9] J. Perez, P. J. Sanz, M. Brysony S. B. Williams, “A benchmarking study on single image dehazing techniques for underwater autonomous vehicles,” in OCEANS 2017 - Aberdeen, 2017, pp. 1–9.

[10] P. Soma and R. K. Jatoth, “Implementation of a novel, fast and efficient image de hazing algorithm on embedded hardware platforms,” Circuits, Systems, and Signal Processing, Aug. 2020. DOI: 10.1007 / s00034-020-01517-4. [En línea]. Disponible: http://dx.doi.org/10.1007/s00034-020-01517-4.