elektron ámbar

Original Article

Revista elektron, Vol. 6, No. 1, pp. 36-40 (2022)
https://doi.org/10.37537/rev.elektron.6.1.149.2022

Implementación en FPGA de algoritmo para análisis parasitario

FPGA Algorithm Implementation for Parasitic Analysis

Rombolá Guido [1], Leiva Lucas [2], Vazquez Martin [3], Toloza Juan [4], Sagües Federica [5], Saumell Carlos [6]

[1,2,3,4] *Facultad de Ingeniería - Universidad Nacional de Tres de Febrero - Caseros - Buenos Aires - Argentina*

[2,3,4] *LabSET - Universidad Nacional del Centro de la Provincia de Buenos Aires - Tandil - Buenos Aires - Argentina*

[5,6] *CIVETAN - Universidad Nacional del Centro de la Provincia de Buenos Aires - Tandil - Buenos Aires - Argentina*

[1] grombola@untref.edu.ar

[2,3,4] {lleiva,mvazquez,jmtoloza]@labset.exa.unicen.edu.ar

[5] federica@vet.unicen.edu.ar

[6] saumell@vet.unicen.edu.ar

Recibido: 01/03/22; Aceptado: 04/05/22

***Resumen*— Un control parasitario eficiente permite reducir pérdidas significativas en la agroindustria. Los métodos actuales con los que se realiza este tipo de controles imponen costos y demoras. Por ello, se propone el desarrollo de un dispositivo portátil que automatice esta tarea. En este trabajo se presenta la implementación hardware de un algoritmo de conteo automático de huevos de parásitos utilizando síntesis de alto nivel. Los resultados demuestran la factibilidad de la implementación, con un 87% de precisión operando a una tasa de hasta de 65 frames por segundo y una ocupación de LUTs menor al 45%, considerando dos kits comerciales (PYNQ-Z1 y ULTRA96V2).**

Palabras clave: Análisis Parasitario, Procesamiento de Imágenes, HLS, FPGA.

***Abstract*— An efficient parasite control reduces significant losses in the agribusiness, but current methods involve costs and delays. Therefore, the development of a portable device to automates this task is proposed. This work presents a hardware implementation of an automatic parasite egg counting algorithm using high-level synthesis. The results demonstrate the feasibility of the implementation, with an 87% accuracy operating at a rate of up to 65 frames per second and an occupation of LUTs less than 45%, considering two commercial kits (PYNQ-Z1 and ULTRA96V2).**

Keywords: Parasitic Analysis, Image Processing, HLS, FPGA

I. Introducción

El sector agropecuario y agroindustrial es el principal mercado de exportación de Argentina. En esta industria un buen control parasitario permite mitigar grandes pérdidas económicas.

Los planes sanitarios en el ganado bovino incluyen el diagnóstico y control de potenciales enfermedades. La acción de parásitos internos en los animales es un trastorno constante en la eficiencia productiva ocasionando grandes pérdidas [1]. Los parásitos internos producen huevos que son eliminados con la materia fecal; en el ambiente desarrollan a larvas infectantes, contaminando los pastos y de esta manera reiniciando el ciclo parasitario cuando son ingeridas por el animal. El diagnóstico parasitológico de rutina se realiza a través de la determinación de huevos en la materia fecal pudiendo detectar el problema y anticiparse a las consecuencias clínicas evitando que los animales enfermen. Las pérdidas subclínicas vinculadas a la ganancia de peso son las que el productor no consigue detectar. La pérdida de peso (o dejar de ganarlo) en bovinos puede alcanzar el rango de 15 a 40 kg por animal en pastoreo. La pérdida de peso subclínica no es posible compensar debido a que ocurre en la etapa de crecimiento y desarrollo del animal, provocando serias consecuencias económicas. Para Argentina, se estimaron pérdidas ocasionadas por parásitos en aproximadamente 200 millones de dólares anuales [1]. Para minimizar el impacto de las parasitosis subclínicas, se realizan en los programas de control sanitario, monitoreos regulares mediante análisis de materia fecal, donde a través del conteo de huevos, se puede estimar indirectamente la carga parasitaria sobre él animal. La técnica de laboratorio que se utiliza tiene muchas limitaciones.

Actualmente la detección de los parásitos gastrointestinales en bovinos se realiza de forma indirecta, a través del análisis de materia fecal utilizando la técnica Mcmaster modificada [2]. Este método consiste en la observación, identificación y conteo de huevos de parásitos gastrointestinales utilizando microscopio óptico, tarea que requiere de personal experimentado, equipamiento de laboratorio y tiempo que incrementa los costos de su realización. Asimismo, los resultados de los análisis están sujetos a la fatiga del personal encargado de realizar esta tarea. Se estima que el error promedio es del 20% cuando es realizada por un experto.

Las nuevas tecnologías permitieron comenzar a generar soluciones para conteo automático. Algunos de estos métodos se detallan en [3,4] los cuales realizan el procedimiento a través del procesamiento de imágenes sobre muestras pre-procesadas en laboratorio, e incluso existen soluciones utilizando smartphones vinculado con equipamiento especializado [5]. Se pueden encontrar en la literatura también soluciones basadas en Deep Learning, como la presentada en [6]. Sin embargo, todas las técnicas requieren de equipamiento costoso y muchas de ellas no son portables.

En este contexto, los FPGA son cada vez más utilizados para la implementación de aplicaciones de procesamiento de imágenes, principalmente en aplicaciones embebidas de tiempo real, donde la latencia y el consumo de potencia son consideraciones importantes [7]. Además, la incorporación de nuevas metodologías de desarrollo, como las herramientas de síntesis de alto nivel (HLS), proporcionan beneficios significativos para implementar algoritmos de procesamiento de imágenes en FPGA y reduce los tiempos de desarrollo [8, 9].

En este trabajo se presenta el desarrollo de la implementación hardware de un algoritmo para la detección, identificación y conteo de huevos de parásitos. El objetivo es la automatización de la tarea de conteo, incidiendo además en un incremento de la precisión. Este algoritmo es parte de una solución tecnológica, un dispositivo portátil, que evita la necesidad de lidiar con el proceso de tomar las muestras, transportarlas al laboratorio, aguardar por los resultados, y con los resultados poder aplicar las medidas de control y/o terapéuticas correspondientes teniendo en cuenta el tiempo que lleva cada análisis. El flujo de diseño fue realizado utilizando herramientas de síntesis de alto nivel. El trabajo contiene las métricas de implementación obtenidas para el IP Core en un dispositivo Zynq (PYNQ-Z1) y un dispositivo UltraScale+ (Ultra96V2), ambos de Xilinx. Considerando que la tecnología de ambas plataformas son diferentes (SoC y MPSoC) resulta interesante evaluar las prestaciones ofrecidas en ambas plataformas de acuerdo a los requerimientos del problema.

En la sección II se presentan los detalles de implementación del algoritmo, en la sección III se presentan los resultados obtenidos tanto en precisión como en análisis de tiempos y área. Finalmente, en la sección IV se presentan las conclusiones y trabajos futuros.

II. Desarrollo

La implementación del IP core para el conteo de huevos de parásitos se realizó utilizando Vivado HLS 2019.2. Una de las principales ventajas de este entorno es el amplio soporte ofrecido por el fabricante [10], y además permite el uso de una biblioteca de funciones para el procesamiento de imágenes (xfOpenCV). Esta biblioteca contiene un subconjunto de funciones de la biblioteca OpenCV, optimizadas para la síntesis.

Se realizaron síntesis de Vivado HLS para su exportación del IP core a dos kits de desarrollo comerciales: PYNQ-Z1 y ULTRA96V2. Se escogen estas dos plataformas, ya que si bien permiten generar una solución sólo en su lógica programable, son de bajo costo y ambas tienen soporte para PYNQ, facilitando tanto el desarrollo de aplicaciones a nivel de usuario, como la incorporación de dispositivos de captura USB. Esto último resulta útil en este caso para conectar un microscopio digital USB de bajo costo, que permite la entrada de las imágenes a analizar [11]. Estos microscopios alcanzan hasta 1000X de Zoom óptico, con foco ajustable, iluminación led con intensidad controlada, y permiten entregar imágenes en resoluciones de 640x480 y 800x600 píxeles. Además, pueden capturar los huevos de parásitos considerando que el tamaño promedio de los mismos es de 85 micrómetros.

A. Descripción del algoritmo

La implementación del algoritmo está basada en un pipeline de imagen, y consta de tres etapas: preprocesamiento, segmentación y clasificación.

La primera etapa de preprocesamiento convierte la imagen de entrada de un espacio de color RGB a escala de grises, y aplica una binarización de la imagen, dando como resultado una imagen con los objetos candidatos a ser analizados en etapas siguientes en color blanco, y el fondo en color negro. La Figura 1 presenta los resultados de las operaciones que ejecuta esta etapa, para una imagen de entrada de ejemplo. El umbral de la binarización fue establecido en 116, con base en la experticia del desarrollador de software y del experto de laboratorio, y es fijo para todas las imágenes. El algoritmo permite fijar este valor ya que todas las imágenes son capturadas en un ambiente estructurado: cámara de depósito de la muestra, porcentaje de solución y muestra, e iluminación.

La etapa de segmentación identifica cada uno de los objetos de interés (blobs) y extrae sus características. Cada blob se conforma por un conjunto de píxeles blancos conectados, y aislados del resto. Para la detección de blobs, se realizó una implementación basada en una única pasada, que enumera los blobs con sus características, calculadas incrementalmente [12]. Esto evita el almacenamiento de la imagen completa en memoria, y solo necesita la información de las últimas dos filas en cada instante de tiempo. No guardar la imagen completa en memoria permite destinar menos recursos de memoria del dispositivo y que la solución pueda implementarse en un FPGA de menores prestaciones. Considerando que en esta etapa opera sobre una imagen binaria, se representa cada fila como una lista de secuencias con píxeles consecutivos de color blanco. De cada secuencia, son relevantes los números de columna de inicio y de fin. El resto de los valores no contemplados corresponden a píxeles negros. Se conoce a esta representación con el nombre de run-length encoding.

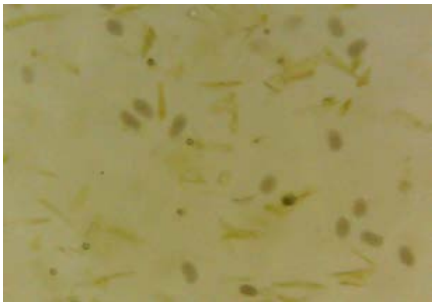
Para identificar los blobs, se deben comparar las filas de acuerdo a las siguientes reglas:

- Si una secuencia de píxeles de una fila no se solapa con ninguna secuencia de la fila anterior, se crea un nuevo blob cuyo contenido es únicamente el de esta secuencia.
- Si una secuencia de píxeles de una fila se solapa con alguna secuencia de la fila anterior, entonces esas dos secuencias corresponden al mismo blob. Este ejemplo se ilustra en la figura 2 (a).
- Si hay dos secuencias de píxeles en una misma fila que originalmente corresponden a dos blobs, y en la fila siguiente una secuencia se solapa con las dos de la fila anterior, entonces se combinan los dos blobs ya que son el mismo objeto. Este ejemplo se ilustra en la figura 2 (b).

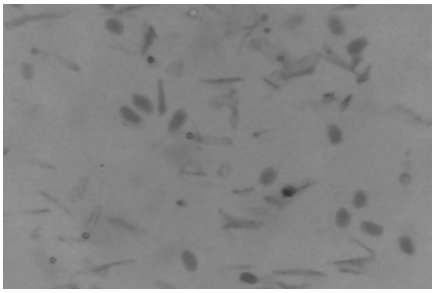
Para cada uno de los blobs detectados se calculan las características que se utilizarán posteriormente para identificar a un objeto [13]. Estas características son:

- *Área (A)*: suma de todos los píxeles blancos de un blob.
- *Mínimo rectángulo que contiene al blob (bounding box)*: rectángulo más pequeño que contiene la totalidad del blob.
- *Momentos geométricos de orden 1 y 2*: promedios ponderados de las intensidades de los píxeles, que

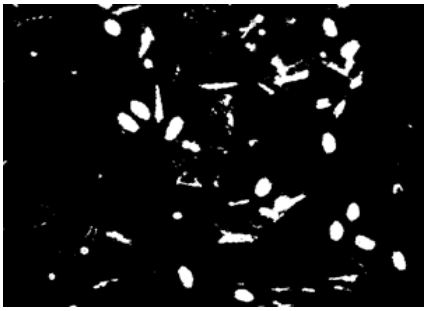
permiten identificar algunos aspectos de la forma de una imagen.



(a)

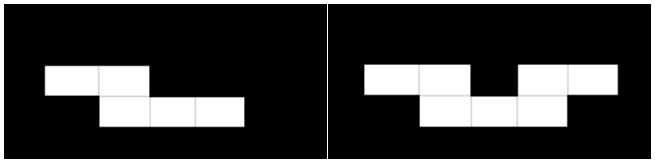


(b)



(c)

Fig. 1. (a) imagen de entrada, (b) conversión a escala de grises, (c) binarización.



(a) (b)

Fig. 2. (a) ejemplo de dos secuencias que al solaparse son parte del mismo blob, (b) ejemplo de dos secuencias independientes que se combinan para formar el mismo blob

La salida de esta etapa consiste en un arreglo de los blobs identificados con sus respectivas características.

La tercera etapa (clasificación) determina si el blob se corresponde o no con un huevo parasitario. Para ello, utiliza las características calculadas en la etapa de segmentación, en conjunto con características obtenidas a partir de los valores de la etapa anterior. Las nuevas características generadas son:

- *Excentricidad (E)*: representa el grado de elongación de un blob, siendo 1 para un círculo, y valores mayores que 1 para las regiones elongadas.
- *Relación de área (RA)*: cociente entre área de la elipse equivalente al blob, y el área del blob, siendo esta elipse equivalente aquella cuyo centro se corresponde con el centro de gravedad del blob, y la orientación de su semieje mayor es la del eje mayor del blob.

Si los valores de las características se encuentran contenidos dentro de ciertos intervalos, entonces el blob es identificado como positivo. Los valores de los intervalos se establecieron mediante prueba y error a fin de maximizar los resultados, quedando definidos como: A=[250, 700], E=[1.5, 6) y RA=[1, 1.06]. Considerando que la toma de imágenes se realiza en un ambiente estructurado, estos valores son fijos.

La salida de esta etapa, que coincide con la salida de la solución completa, se corresponde con la cantidad de huevos detectados en la imagen.

B. Optimizaciones para la síntesis del IP Core

Se utilizaron representaciones de datos de precisión arbitraria para obtener mejores resultados de ocupación de recursos y tiempos de procesamiento. Para aquellas operaciones que requieren de números decimales, se utilizaron representaciones en punto fijo.

Se aplicó además una optimización en la función encargada de incrementar los momentos geométricos para un blob determinado. La ecuación para calcular los momentos geométricos de un blob utilizando la técnica incremental del algoritmo de segmentación se define como:

$$m_{pq} = \sum_{i=0}^{k} \sum_{x=x1[i]}^{x2[i]} x^{p} y_{i}^{q} \quad (1)$$

donde k es el número de caminos (secuencias de píxeles contiguos en blanco) que componen al blob, *x1[i]* y *x2[i]* son las coordenadas de inicio y fin de un camino, e *yi* es el número de fila. Se puede reescribir a la Ecuación (1) como:

$$m_{pq} = \sum_{i=0}^{k} \sum_{x=x1[i]}^{x2[i]} x^{p} y_{i}^{q} = \sum_{i=0}^{k} y_{i}^{q} \sum_{x=x1[i]}^{x2[i]} x^{p} \quad (2)$$

Entonces, considerando las equivalencias de las ecuaciones (3), (4) y (5), los momentos geométricos de orden 1 y 2 pueden ser calculados evitando realizar la suma interna como un bucle. En ese sentido, se reduce la comparación e incremento de la variable iteradora, además la cantidad de operaciones deja de depender de la longitud de cada camino.

$$\sum_{i=1}^{n} i = \frac{n*(n+1)}{2} \quad (3)$$

$$\sum_{i=1}^{n} i^{2} = \frac{n*(n+1)*(2n+1)}{6} \quad (4)$$

$$\sum_{i=r}^{n} f(i) = \sum_{i=1}^{n} f(i) - \sum_{i=1}^{r-1} f(i),\ con\ r <= i \quad (5)$$

Adicionalmente, se aplicaron las siguientes directivas de síntesis:

- PIPELINING en el loop de la función que verifica si los caminos de una fila se solapan con los de la fila anterior para formar un blob más grande. Permite la ejecución concurrente de operaciones.
- INLINE a la función de cálculo de momentos geométricos.
- DATAFLOW en la función principal, y en la etapa de preprocesamiento en donde la imagen recibida como un stream de datos pasa por una serie de modificaciones por las distintas funciones de procesamiento. Permite que las funciones se solapen en su ejecución.
- LOOP UNROLL en el loop que copia los caminos de la fila actual al arreglo con los caminos de la fila anterior posición a posición, permitiendo ejecutar en paralelo estas copias y disminuyendo la latencia.

III. Resultados experimentales

El algoritmo fue verificado con un dataset compuesto por 30 imágenes obtenidas a partir del uso de un microscopio digital USB compatible con los kits de desarrollo propuestos (PYNQ-Z1 y ULTRA96V), que contienen un total 234 huevos de parásitos. Las capturas se realizaron en el Laboratorio de Parasitologia de la Facultad de Ciencias Veterinarias,UNICEN, asociado a los centros CIVETAN (UNICEN-CIC-CONICET) y CISAPA (UNICEN [14], tal como se presenta en la Fig. 3.

Se analizaron los resultados del algoritmo en base a la precisión, definida en la ecuación (6), y sensibilidad (*Recall*) definida en la ecuación (7), teniendo en cuenta los resultados obtenidos para:

- Verdaderos Positivos (VP): objetos que son huevos y son detectados como tal.
- Falsos Negativos (FN): objetos que son huevos y no son detectados como tal.
- Falsos Positivos (FP): objetos que no son huevos y son detectados como tal [15].

$$Precisión = \frac{VP}{VP+FP} \quad (6)$$

$$Sensibilidad = \frac{VP}{VP+FN} \quad (7)$$

Fig 3. Proceso de captura de imágenes de validación del algoritmo.

Los valores de *VP*, *FN* y *FP* fueron obtenidos a partir de una comparación manual de los resultados generados del algoritmo, respecto a los indicados por el experto. Para poder hacer esto, fue necesaria una implementación alternativa de la solución, que, además de indicar la cantidad de huevos detectados, marque las ubicaciones en la imagen. Este proceso se realizó únicamente para la simulación de alto nivel y no se incluye como parte de la solución final apta para hardware debido a que la lógica de marcado implicaría un costo adicional de recursos. Un ejemplo de cómo se ven estas imágenes resaltadas se muestra en la Figura 4. En ella, los elementos rodeados por un recuadro azul se corresponden con huevos parasitarios.

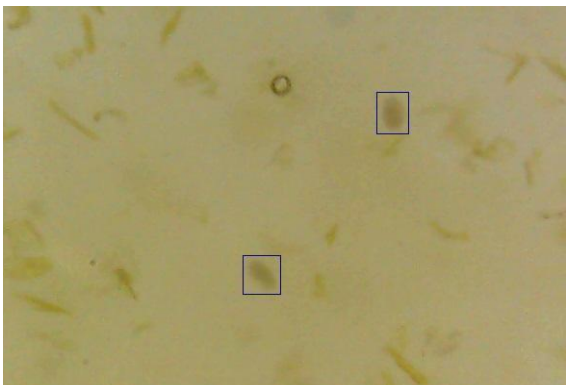


Fig 4. salida de simulación de alto nivel del algoritmo.

Para obtener los resultados de precisión y sensibilidad, se realizó una suma de los VP, FN y FP de cada imagen, para luego aplicar las ecuaciones (6) y (7) sobre los totales. Como resultado, se obtuvo una precisión de 87% y una sensibilidad del 77%, lo que indica una buena efectividad del método.

El core fue sintetizado con una frecuencia de reloj de entrada de 100MHz y cosimulado con la herramienta. Esto último se refiere a realizar una comparativa entre los resultados obtenidos de la simulación de alto nivel y los resultados obtenidos con la solución sintetizada.

En la Tabla I se presentan los resultados de porcentaje de ocupación de recursos obtenidos de las síntesis para ambos kits de desarrollo, demostrando la factibilidad de implementación sobre ambas plataformas.

TABLA I.
Ocupación de Recursos (Resultados de Síntesis)

Kit	*BRAM_18K*	*DSP48E*	*FF*	*LUT*
Pynq Z1	10 %	19 %	11 %	44%
Ultra96V2	6 %	15 %	6 %	41 %

Respecto a los tiempos, se obtuvo una latencia de 22 milisegundos para PYNQ-Z1 y 16 milisegundos para ULTRA96V2, dando una capacidad de procesamiento de 45 y 62 frames por segundo respectivamente.

IV. Conclusiones

En este trabajo se presentó el desarrollo de algoritmo en hardware para el conteo automático de huevos de parásitos para ser aplicado en veterinaria ganadera. Los resultados obtenidos demuestran la factibilidad de la implementación

de un sistema portable de bajo costo, capaz de ser utilizado en el campo por especialistas, dada la robustez de la tecnología seleccionada y las ventajas de los FPGAs respecto al bajo consumo. Asimismo, el algoritmo implementado alcanza un nivel de precisión aceptable (87%) y los resultados de síntesis demuestran la factibilidad de implementación en dos plataformas tanto en tiempo, como utilización de recursos.

Además se destaca que la metodología de desarrollo seleccionada fue adecuada. Se puede afirmar que el uso de síntesis de alto nivel contribuye positivamente tanto al tiempo de desarrollo, como a la verificación y validación de las soluciones generadas.

Se pretende que el algoritmo desarrollado forme parte del desarrollo de un equipamiento de análisis automático, con la intención de cubrir las necesidades de los más de 100 laboratorios actualmente habilitados por el Gobierno de la Provincia de Buenos Aires. Se espera continuar con el desarrollo del prototipo de sistema funcional y la evaluación del mismo por parte de expertos. Además se pretende incluir la clasificación taxonómica de huevos de parásitos, y analizar la adaptación del algoritmo a otras especies de animales.

Agradecimientos

Este trabajo fue parcialmente financiado por la Secretaría de Investigación y Desarrollo de UNTREF (Código de Proyecto 32/19 80120190100183TF) y la SeCAT de UNICEN (Código de Proyecto 03/C287).

Referencias

[1] Bulman, M. "Pérdidas económicas directas e indirectas por parasitos internos y externos de los animales domésticos en Argentina". Anales de la Academia Nacional de Agronomia y Veterinaria. 2012. Tomo LXVI. Buenos Aires. Argentina. ISSN. 0327-8093 pag. 76-176

[2] Fiel, C. A.; Steffan, P. E.; Ferreyra, D. A. 2011. Diagnóstico de las parasitosis más frecuentes en rumiantes: técnicas de diagnóstico e interpretación de resultados. Primera edición - Tandil: Abad Benjamín, 2011. ISBN 978-987-33-1502-2

[3] T.H.M. Mes, M. Eysker, H.W. Ploeger. "A simple, robust and semi-automated parasite egg isolation protocol." Nature protocols 2.3: 486-489, 2007.

[4] Ghazali, Kamarul H., Raafat S. Hadi, Z. Mohamed. "Automated system for diagnosis intestinal parasites by computerized image analysis." Modern Applied Science 7.5: 98, 2013.

[5] Slusarewicz, Paul, et al. "Automated parasite faecal egg counting using fluorescence labelling, smartphone image capture and computational image analysis." International journal for parasitology 46.8: 485-493, 2016.

[6] A. Akintayo, G.L. Tylka, A.K. Singh, B. Ganapathysubramanian, A. Singh, S. Sarkar, S "A deep learning framework to discern and count microscopic nematode eggs." Scientific reports 8.1, 2018.

[7] D.G. Bailey. "Image processing using FPGAs.", 2019.

[8] D.G. Bailey. "The advantages and limitations of high level synthesis for FPGA based image processing." Proceedings of the 9th International Conference on Distributed Smart Cameras. 2015.

[9] A. Cortes, I. Velez, A. Irizar. "High level synthesis using Vivado HLS for Zynq SoC: Image processing case studies." 2016 Conference on Design of Circuits and Integrated Systems (DCIS). IEEE, 2016.

[10] S. Lahti, P. Sjovall, J. Vanne, y T. D. Hamalainen, «Are We There Yet? A Study on the State of High-Level Synthesis», IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 38, n.o 5, pp. 898-911, may 2019, doi: 10.1109/TCAD.2018.2834439.

[11] USB Microscope Cameras, https://www.dinolite.us/en/products/digital-microscopes

[12] J. Trein, A. Th Schwarzbacher, B. Hoppe. "FPGA implementation of a single pass real-time blob analysis using run length encoding." MPC-Workshop, February. 2008.

[13] W. Burger, M. Burge, “Regions in binary images“ en Digital Image processing, 2da edición. Londres, Springer, cap 10, pp 209-252, 2016.

[14] CIVETAN, http://www.civetan-conicet.gob.ar

[15] “Classification: Precision and Recall | Machine Learning Crash Course.” Google Developers, 10 Febrero de 2020. Accedido el 9 de Noviembre de 2021. [Online], Disponible en: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall?hl=es_419.