# Computer vision aplicada al control de calidad sobre confites

Alexis Aranda Ocampo[1], Carla Espinola Hamm[1], Lara Cavicchioli[1], Patricia Sandagorda[1], Valeria Silvestri[1]

[1]Universidad Nacional de La Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina
alexis_aranda_ocampo@hotmail.com, carlae.hamm@gmail.com, laracavic132@gmail.com, patriciasandagorda@gmail.com, valeria.ayelen@gmail.com

**Resumen.** En el desarrollo de este paper se buscará analizar cómo utilizar técnicas de procesamiento de imágenes, a través de la librería OpenCV, el uso de una cámara 4K y un embebido Jetson, para realizar el análisis de calidad sobre un confite de marca Rocklets. De esta forma podemos aprovechar el poder del paralelismo de este embebido con GPU incorporada para el procesamiento. Este análisis consiste en examinar y validar no solo la forma del confite, sino también su estado de conservación, entiéndase: sin quebraduras, con el brillo correcto, color y tono correcto. Esto se da en el marco de la adición de esta funcionalidad al actual embebido "Rocky", un catalogador por color de Rocklets. El objetivo inicial es poder añadir la función de detección de Rocklets en mal estado. El objetivo secundario es añadir la posibilidad de realizar un análisis de múltiples Rocklets a la vez.

**Palabras claves:** GPU, paralelismo, OpenCV, calidad, Rocklets, computer vision, procesamiento de imágenes, Jetson, sistema embebido

## 1 Introducción

Actualmente poseemos un sistema de clasificación de confites marca Rocklet, apodado Rocky. Este sistema tiene un sensor de color para tomar una muestra del color del Rocklet y decidir en qué recipiente depositarlo. Planeamos utilizar una cámara 4K y un embebido con GPU integrada para tomar una fotografía de alta resolución del Rocklet y hacer un análisis de calidad más minucioso, que permita reconocer más propiedades de un Rocklet y con ellas tomar mejores decisiones. Utilizaremos técnicas de procesamiento de imágenes y computer vision.
Actualmente se utiliza el procesamiento de imágenes y olores para determinar la calidad del mango de forma no destructiva[1]. Otro estudio propone utilizar OpenCV

[1] Puede ver la investigación completa en la referencia 4.

sobre Android para analizar daños en una estructura[2], mientras que en el campo de medicina se utiliza para detectar fracturas en los huesos[3].

## 2 Desarrollo

Nuestra investigación surge como ampliación de nuestro sistema Rocky. Actualmente el mismo recibe un rocklet, realiza la clasificación respecto a su color y luego lo despacha al recipiente correspondiente. No tiene en cuenta ningún parámetro de calidad.
La idea es utilizar un embebido externo con GPU integrada que sea capaz de procesar imágenes, junto a una cámara de alta resolución capaz de darnos datos precisos sobre el rocklet, para poder tomar mejores decisiones sobre su calidad.
Investigamos el uso de la tecnología OpenCV para procesar imágenes con el objetivo de reconocer color, brillo, posibles quebraduras y forma del rocklet, mediante cálculos de gradientes y contorno de la figura. Para esto, utilizamos las funciones de contorno que además, nos permiten corroborar cuánto se desvía la forma de un rocklet de un círculo ideal para decidir si cumple con los requisitos de calidad.
Para ello, agregaremos una cámara 4K al sistema embebido (por ejemplo: la cámara See3CAM_130 de e-com Systems, que actualmente se vende a US$99) y una GPU NVIDIA Jetson. Actualmente existen los modelos TK1, TX1 y TX2 con arquitecturas Kepler, Maxwell y Pascal respectivamente, que cuestan entre US$366 y US$468.

## 3 Explicación del algoritmo.

### 3.1 Algoritmo para un solo rocklet

El algoritmo en alto nivel consiste en unos pocos pasos:

1. Tomar la foto de un rocklet con la cámara 4K
2. Analizar la foto para obtener parámetros que describan al rocklet
3. Determinar la calidad del rocklet
4. Decidir en qué recipiente se debe colocar el rocklet en base a su calidad y color

El cuello de botella se encuentra en el paso 2, ya que una imagen en resolución 4K (4096 x 2160 px) cuenta con una gran cantidad de datos. Por eso se decidió ejecutarlo sobre la GPU del módulo Jetson elegido.

---

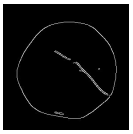
[2] Puede ver la investigación completa en la referencia 5.
[3] Puede ver la investigación completa en la referencia 6.

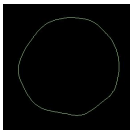
Se debe tener en cuenta que los bloques soportan un máximo de 1024 hilos en las arquitecturas Kepler, Maxwell y Pascal[4]. En los pasos donde se deba procesar toda la imagen y se pueda tratar cada píxel de forma independiente, utilizaremos bloques de 1024 hilos en una dimensión. Cada uno tomará un cuarto de cada columna de 4096 píxeles. Serán organizados en una grilla en dos dimensiones de 4 x 2160 bloques para cubrir toda la imagen.

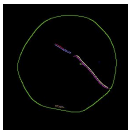
1. Hallar el contorno del rocklet a partir de la imagen original. Para ello es necesario:
   a. Pasar la imagen a escala de grises, procesando cada píxel de forma independiente.
   b. Generar una imagen binaria. Contamos con dos métodos posibles:
      - Threshold: separa el rocklet del fondo. Esto generará menos contornos mejor definidos, lo que será más simple de procesar, pero nos perderá los detalles internos del rocklet. Es un proceso simple, paralelizable a nivel de píxeles, pero tiene la desventaja de que un cambio en la iluminación podría variar el resultado.
      - Canny edge detection: encuentra los bordes de la imagen. Es un proceso más complejo, ya que debe calcular el gradiente por cada píxel y luego procesarlo en su búsqueda. Generará más contornos que también deberemos procesar, pero no nos perderá los detalles internos del rocklet.
   c. Hallar los contornos con cv.findContours().


Imagen original


(1) Threshold


(2) Canny edge detection


Contornos basados en (1)


Contornos basados en (2)

2. Obtener las propiedades del rocklet a partir de su contorno. Estas funciones se pueden ejecutar en paralelo con diferentes cantidades de hilos y bloques según el tamaño del contorno y el tipo de función. Las propiedades que podemos obtener son:
   a. Área del contorno con la función cv.contourArea()
   b. Círculo mínimo que lo encierra con cv.minEnclosingCircle(). La diferencia entre su área y el área del contorno indica cuánto se aleja del círculo ideal.
   c. Forma convexa que lo encierra con cv.convexHull(). Podemos hallar su área y compararla con el área del contorno, o podemos usar cv.convexityDefects() para encontrar todos los puntos donde el contorno se desvía de la forma convexa junto con su distancia.[5]

[4] Especificado en las referencias 1 y 2.

[5] cv.convexityDefects() es una función muy precisa, encuentra incluso las mínimas desviaciones e informa una serie de propiedades por cada una. Podemos usar la propiedad fixpt_depth para descartar las desviaciones mínimas que no nos interesan.

d. Rectángulo de mínima área que lo encierra con cv.minAreaRect(). Esta función devuelve los vértices de un rectángulo que puede estar rotado. Con estos datos podemos calcular si el rocklet se encuentra ovalado y su orientación.

3. Obtener las propiedades del color.
   a. Generar una máscara a partir del contorno que sólo permita pasar los píxeles internos del rocklet: los que tienen su color.
   b. Con la imagen original y la máscara aplicar la función cv.minMaxLoc() para obtener los valores mínimo y máximo del conjunto de píxeles seleccionado, y cv.mean() para obtener el color promedio.
   c. Si aplicamos el punto b sobre una imagen en escala de grises, obtendremos las intensidades mínima, máxima y un promedio.

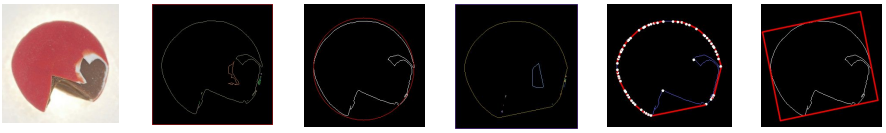


Imagen original | Contornos | Círculo mínimo | Formas convexas | Desviaciones de la forma convexa | Rectángulo mínimo

Con todas estas propiedades estamos listos para el siguiente paso: determinar la calidad del rocklet.

### 3.2 Algoritmo para múltiples rocklets

Es posible analizar en una foto varios rocklets a la vez trabajando con una pequeña adaptación al algoritmo anterior. El primer paso será identificar y aislar cada rocklet para poder aplicar el algoritmo anterior a cada uno por separado. Para separarlos usamos la función cv.watershed(), que nos indica los bordes para cada rocklet incluso en el caso de que se toquen entre sí. Sin embargo, es conveniente que no se toquen, ya que en ese caso se los puede separar con threshold, que es un proceso más simple.

## 4 Pruebas que pueden realizarse

Una vez implementado el sistema, se recomienda realizar pruebas con gran variedad de rocklets. Es importante que las variaciones se den en su color, brillo, forma y quebraduras. Sería interesante encontrar rocklets manchados, cuadrados u ovalados, o recortar parte de los confites para generar formas no convexas.
Se debe tener en cuenta que: la cámara debe ver los rocklets desde arriba, no en perspectiva, sino reconocería rocklets ovalados donde no los hay; y la iluminación ambiente puede afectar a las lecturas, por lo que se debe probar en distintos ambientes.
En la adaptación para reconocer muchos rocklets a la vez, se puede probar ubicándolos de forma desordenada, variando la distancia entre ellos y los colores

presentes, probando qué ocurre cuando dos o más rocklets se tocan y cómo cambia el resultado cuando son del mismo o distintos colores los que se tocan.

Por último, se recomienda armar casos de prueba que permitan comparar el nuevo sistema contra el anterior, analizando tiempos de procesamiento por rocklet y tasas de error.

## 5 Conclusiones

A lo largo de este trabajo nos enfocamos en cómo mejorar el clasificador de rocklets Rocky, incorporando procesamiento de imágenes para que sea capaz de tomar más y mejores decisiones. Aprendimos gracias a los casos similares citados, que se pueden extraer muchas propiedades para evaluar los rocklets, además del color.

Entonces consideramos incluir una cámara de alta resolución y una GPU para procesar la gran cantidad de datos. Notamos que sus elevados precios no lo hacen viable como aparato doméstico. Sin embargo, analizando el funcionamiento de los distintos métodos de OpenCV podemos concluir que no es necesaria la alta resolución para encontrar las propiedades buscadas. Reduciendo la cantidad de datos, no hará falta la potencia de una GPU y el análisis podría resolverlo otra plataforma de menor costo, como una Raspberry Pi. OpenCV existe para diversas plataformas, por lo que se mantendría el algoritmo. Sugerimos investigar estas opciones para un próximo trabajo.

Nuestra solución se ve prometedora en el contexto de la producción industrial de rocklets, donde una imagen de alta resolución puede incluir a cientos de rocklets sin perder información, permitiendo evaluar la calidad de los confites sin la necesidad de supervisores humanos.

## 6 Referencias


1. NVIDIA Tesla P100 - The Most Advanced Datacenter Accelerator Ever Built Featuring Pascal GP100, the World's Fastest GPU. (WP-08019-001_v01.2 ed.) NVIDIA (2016)
2. NVIDIA's Next Generation CUDA Compute Architecture: Kepler TM GK110/210. (whitepaper v1.1) NVIDIA (2014)
3. Documentación de OpenCV para GPU. https://docs.opencv.org/2.4/modules/gpu/doc/gpu.html
4. X. Huang, R. Lv, S. Wang, J. H. Aheto, C. Dai: Integration of computer vision and colorimetric sensor array for nondestructive detection of mango quality (2018).
5. ZhiQiang Chen, Jianfei Chen: Mobile Imaging and Computing for Intelligent Structural Damage Inspection. Ren-Jye Dzeng (2014)
6. Samuel Fibrianto Kurniawan, I Ketut Gede Darma Putra, A.A Kompiang Oka Sudana: Bone fraction detection using OpenCV(2014)