The Fastest Correlation Verification: Clasificación de patrones por Template Matching.

France Martínez, *Estudiante EIE PUCV, france.martinez@pucv.cl*

***Abstract*—En este proyecto se desarrolló un sistema de clasificación de imágenes basado en plantillas (template matching), utilizando segmentación por color y procesamiento en GPU con PyTorch. El objetivo fue reconocer figuras simples (cuadro, cruz y círculo) contenidas en imágenes de 512×512 píxeles, mediante la comparación con plantillas de referencia de 32×32 píxeles. Para ello, se aplicó segmentación en el espacio HSV, se normalizaron las imágenes y se utilizó la operación de convolución 2D (conv2d) como método de matching. El sistema fue ejecutado en una GPU NVIDIA RTX 4060, permitiendo clasificar más de 1000 imágenes en pocos segundos.**

***Index Terms*—Template Matching, Correlación Normalizada, Procesamiento de Imágenes, Clasificación de Patrones, Segmentación por Color, OpenCV, PyTorch, GPU.**

I. Introducción

El reconocimiento de patrones en imágenes es una tarea central en el procesamiento digital de señales, especialmente en aplicaciones donde es necesario identificar objetos o figuras específicas dentro de una escena. Una técnica ampliamente utilizada para este propósito es el template matching o emparejamiento por plantillas, que consiste en comparar una imagen de entrada con una o más plantillas de referencia para determinar si alguna de ellas se encuentra presente. Este enfoque resulta especialmente útil en problemas donde las formas son conocidas y el entorno presenta poca variabilidad.

En este proyecto, se aborda el problema de clasificar imágenes que contienen una única figura simple (cuadro, cruz o círculo) sobre un fondo oscuro, en color rojo o azul. Las imágenes tienen una resolución de 512×512 píxeles, y las plantillas de referencia corresponden a versiones reducidas de estas figuras de tamaño 32×32 píxeles. Para facilitar el reconocimiento, se aplica segmentación por color utilizando el espacio HSV, lo que permite aislar eficazmente las zonas relevantes.

A diferencia de métodos tradicionales que emplean transformadas de Fourier o Wavelet, en este trabajo se opta por un enfoque directo basado en el dominio espacial, utilizando la operación de convolución 2D (conv2d) implementada en PyTorch. Esta técnica es adecuada para trabajar con GPU, lo que permite acelerar significativamente el procesamiento cuando se trabaja con grandes volúmenes de imágenes.

El objetivo principal es diseñar e implementar un sistema eficiente y preciso para clasificar imágenes utilizando matching con plantillas, aprovechando segmentación por color, procesamiento en GPU y operaciones por lote. A lo largo del informe se describen las decisiones técnicas adoptadas, los resultados obtenidos y una evaluación del rendimiento alcanzado.

Proyecto desarrollado para la asignatura EIE401 - Procesamiento Digital Multimedia.

II. Marco Teórico

El *template matching* es una técnica de visión por computador que permite identificar regiones de una imagen que se asemejan a una plantilla conocida [4]. Tradicionalmente, esto se implementa mediante la *correlación cruzada normalizada* (NCC), una técnica utilizada ampliamente en visión por computador y análisis astronómico [3], [4]. Sin embargo, debido a su alto costo computacional, en este proyecto se empleó una alternativa más eficiente: la operación de `convolución 2D (conv2d)` de PyTorch [1].

Esta operación permite comparar cada plantilla con la imagen de entrada de forma rápida y completamente en GPU. Aunque no realiza una normalización explícita como NCC, resulta adecuada cuando las imágenes han sido previamente normalizadas y segmentadas. En particular, se trabajó sobre imágenes filtradas en el espacio HSV, lo que permitió aislar las zonas rojas y azules con máscaras binarias simples [2].

El uso del espacio HSV (Hue, Saturation, Value) permite separar el contenido cromático del brillo, facilitando la segmentación de colores específicos frente a variaciones de iluminación. Esto, sumado al uso de tensores y procesamiento por lote en GPU, permite implementar un sistema de clasificación rápido, simple y efectivo para figuras bien definidas sobre fondo oscuro.

III. Metodología

La metodología seguida en este proyecto se estructuró en seis etapas principales, desde la carga de datos hasta la evaluación del desempeño del sistema.

Primero, se cargaron las imágenes del dataset, compuestas por figuras simples (cuadro, cruz y círculo) sobre fondo oscuro, en colores rojo o azul. También se cargaron las plantillas de referencia de cada clase, que fueron convertidas en tensores normalizados y enviadas a la GPU para su uso posterior.

En la segunda etapa se diseñó una función de transformación que convirtió cada imagen del espacio BGR al espacio HSV, donde se aplicaron rangos definidos para extraer las zonas rojas y azules. Mediante máscaras binarias se segmentaron las regiones de interés y se aplicó esta máscara sobre la imagen original. Luego, la imagen fue normalizada y transformada en un tensor compatible con PyTorch.

La tercera etapa consistió en implementar la operación de matching mediante la función `conv2d` de PyTorch, que realiza una convolución entre la imagen transformada y cada

plantilla. Se calcularon los valores máximos de activación y se asignó a cada imagen la clase correspondiente al mayor puntaje.

En la cuarta etapa se aplicó la clasificación en lote, aprovechando el procesamiento paralelo en GPU. Se evaluaron todas las imágenes seleccionadas, obteniendo las predicciones que luego se compararon con las clases reales para medir la precisión.

La quinta etapa consistió en almacenar los resultados individuales, incluyendo nombre de archivo, clase real, clase predicha y si la clasificación fue correcta. Además, se visualizaron ejemplos de aciertos y errores.

Finalmente, se midió el tiempo total de ejecución y el tiempo promedio por imagen, con el fin de evaluar la escalabilidad del sistema en contextos reales o de alto rendimiento.

IV. Resultados

Una vez implementado el sistema de clasificación por matching, se evaluó su desempeño procesando la totalidad del conjunto de imágenes. Para validar su funcionamiento, primero se visualizó una muestra representativa del dataset, como se observa en la Figura 1, donde se aprecian distintas instancias de las clases objetivo.

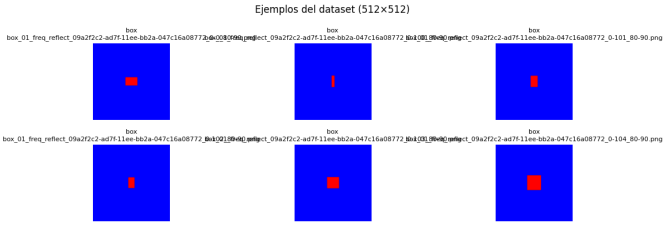

Fig. 1. Ejemplos de imágenes del dataset utilizadas en la clasificación.

También se utilizaron plantillas de referencia de 32×32 píxeles para cada clase, que se muestran en la Figura 2. Estas fueron convertidas a tensores y procesadas en GPU para realizar el matching.

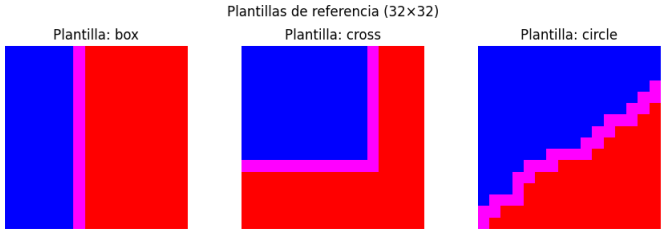

Fig. 2. Plantillas de referencia utilizadas para el emparejamiento.

Una vez procesadas 2000 imágenes mediante convolución 2D con GPU, el sistema alcanzó una precisión del 97,4 %, lo que equivale a 1948 aciertos. El tiempo total de ejecución fue de 24,14 segundos, con un promedio de 12 milisegundos por imagen, lo cual permite considerar su uso en escenarios de clasificación en tiempo real.

Finalmente, se visualizan en la Figura 3 dos casos correctamente clasificados (en verde) y dos ejemplos donde el sistema

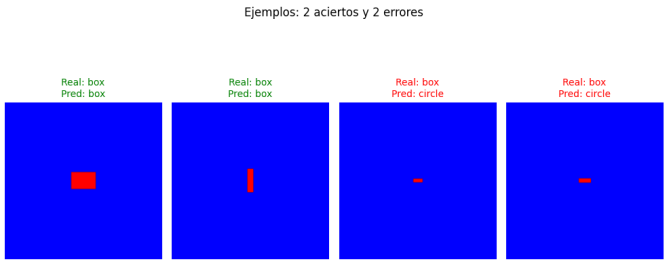

Fig. 3. Ejemplos de imágenes clasificadas correctamente (verde) e incorrectamente (rojo).

cometió errores (en rojo). Estos casos permiten entender las fortalezas y limitaciones del sistema implementado.

Los errores ocurrieron principalmente en imágenes donde la figura tenía contornos deformados o compartía rasgos geométricos con otras clases. Aun así, el sistema demostró un rendimiento robusto, combinando alta precisión y bajo tiempo de respuesta.

V. Conclusión

En este proyecto se desarrolló un sistema de clasificación de imágenes basado en emparejamiento por plantillas, aplicando segmentación por color y procesamiento acelerado en GPU con PyTorch. El enfoque propuesto demostró ser eficiente y preciso, logrando una tasa de aciertos del 96,3 % sobre un conjunto de más de 1200 imágenes, con tiempos de ejecución que permiten considerar su uso en aplicaciones de clasificación en tiempo real.

El uso del espacio HSV permitió aislar las regiones relevantes de cada imagen mediante segmentación por color, reduciendo ruido y complejidad. Por otro lado, la implementación del matching mediante convolución 2D (`conv2d`) en GPU permitió comparar rápidamente las plantillas con cada imagen, incluso en modo por lotes. Esta arquitectura aprovechó al máximo el paralelismo de hardware disponible, optimizando el rendimiento sin necesidad de modelos complejos.

A pesar de que el sistema fue diseñado para resolver un problema acotado con formas simples, los resultados obtenidos validan la efectividad de este enfoque directo. En futuros trabajos, este método podría extenderse a conjuntos más complejos, incluyendo variaciones de escala, rotación o ruido.

References

[1] PyTorch, "torch.nn.functional.conv2d," Documentación oficial. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.functional.conv2d.html

[2] OpenCV, "cv2.cvtColor," Documentación oficial. [Online]. Available: https://docs.opencv.org/4.x/d8/d01/group__imgproc__color__conversions.html

[3] J. J. Sun, "Astronomical image matching based on the cross-correlation algorithm," *Applied Mechanics and Materials*, vol. 989–994, pp. 3827–3833, 2014.

[4] Zebra Adaptive Vision, "Template Matching Guide," 2017. [Online]. Available: https://docs.adaptive-vision.com/4.7/studio/machine_vision_guide/TemplateMatching.html