UNIVERSIDAD POLITÉCNICA SALESIANA

MATERIA: VISIÓN POR COMPUTADOR

Análisis comparativo de robustez entre Momentos de Hu y Zernike, y reconocimiento de formas mediante Firmas Espectrales con FFT de Coordenadas Complejas.

Realizado por:
Marco Cajamarca C
Oscar Campoverde C

Docente:
Ing. Vladimir Robles Bykbaev

Periodo:
67

ÍNDICE

I. Introducción

El reconocimiento automático de formas geométricas constituye un problema fundamental en el campo de la visión por computador, con aplicaciones que abarcan desde sistemas de control industrial hasta interfaces de usuario intuitivas en dispositivos móviles [1]. La capacidad de identificar y clasificar formas básicas como círculos, cuadrados y triángulos representa un primer paso crucial hacia sistemas más complejos de análisis de imágenes y reconocimiento de patrones.

En el contexto actual, donde los dispositivos móviles han alcanzado capacidades de procesamiento significativas, surge la oportunidad de implementar algoritmos de visión por computador directamente en estos dispositivos, eliminando la dependencia de conexiones de red y servicios en la nube [2]. Esto resulta especialmente relevante para aplicaciones educativas, herramientas de diseño asistido y sistemas de interacción natural con el usuario.

I-A. Justificación

La selección del algoritmo apropiado para el reconocimiento de formas es crucial para el éxito de cualquier aplicación de visión por computador. Tradicionalmente, los momentos de Hu y los momentos de Zernike han sido ampliamente utilizados para la descripción de formas debido a sus propiedades de invariancia [3], [4]. Sin embargo, enfoques más recientes basados en descriptores de Fourier y firmas de forma han demostrado ventajas significativas en términos de robustez y precisión [5].

Este trabajo aborda la necesidad de evaluar empíricamente diferentes enfoques algorítmicos para el reconocimiento de formas geométricas básicas, considerando no solo la precisión de clasificación, sino también la robustez ante condiciones adversas como ruido y la viabilidad de implementación en dispositivos móviles con recursos limitados.

I-B. Objetivos

I-B1. Objetivo General: Desarrollar y evaluar un sistema completo de reconocimiento de formas geométricas que comprenda desde la investigación algorítmica hasta la implementación en una aplicación móvil funcional.

I-B2. Objetivos Específicos:

1. Implementar y comparar el rendimiento de diferentes algoritmos de reconocimiento de formas: momentos de Hu, momentos de Zernike y descriptores de forma basados en la transformada de Fourier.
2. Evaluar la robustez de cada algoritmo ante la presencia de ruido gaussiano y sal y pimienta en las imágenes de entrada.
3. Diseñar e implementar una aplicación móvil en iOS que permita el dibujo interactivo y la clasificación en tiempo real de formas geométricas.
4. Optimizar los algoritmos seleccionados para su ejecución eficiente en dispositivos móviles utilizando la biblioteca OpenCV.
5. Analizar el rendimiento comparativo del sistema implementado en términos de precisión, robustez y usabilidad.

II. Planteamiento del Problema

II-A. Definición del Problema

El reconocimiento automático de formas geométricas representa uno de los desafíos fundamentales en visión por computador, particularmente cuando se busca implementar sistemas robustos capaces de operar en tiempo real sobre dispositivos con recursos computacionales limitados. A pesar de años de investigación en el campo, persisten interrogantes sobre cuál es la aproximación algorítmica más efectiva para diferentes escenarios de aplicación.

El problema central abordado en este trabajo se articula en dos dimensiones principales: primero, la necesidad de una evaluación empírica rigurosa que permita determinar el algoritmo de reconocimiento de formas más apropiado considerando tanto precisión como robustez; segundo, la implementación práctica de dicho algoritmo en una aplicación móvil que permita la interacción natural del usuario mediante dibujo directo.

La complejidad del problema se ve amplificada por la variabilidad inherente en las formas dibujadas manualmente, las condiciones de ruido típicas en entornos reales, y las limitaciones computacionales de los dispositivos móviles modernos.

II-B. Antecedentes y Estado del Arte

II-B1. Métodos Tradicionales de Descripción de Formas: Los momentos invariantes propuestos por Hu en 1962 [3] constituyeron uno de los primeros enfoques sistemáticos para la descripción de formas independientemente de su posición, orientación y escala. Estos descriptores, basados en combinaciones de momentos geométricos de segundo y tercer orden, han sido ampliamente utilizados debido a su simplicidad computacional y propiedades teóricas bien establecidas.

Los momentos de Zernike, introducidos por Teague en 1980 [4], representaron un avance significativo al proporcionar un conjunto de descriptores ortogonales que minimizan la redundancia de información. Su formulación matemática basada en polinomios de Zernike ofrece mayor robustez ante ruido y permite una reconstrucción más precisa de la forma original [6].

II-B2. Descriptores de Fourier y Firmas de Forma: Los descriptores de Fourier para contornos cerrados, desarrollados por Persoon y Fu [7], introdujeron una perspectiva diferente al problema mediante la representación de contornos como señales complejas en el dominio de la frecuencia. Esta aproximación permite obtener descriptores invariantes mediante normalización apropiada de los coeficientes de Fourier.

La evolución hacia firmas de forma basadas en representaciones complejas del contorno ha demostrado ventajas particulares en términos de robustez ante deformaciones

locales y capacidad de capturar características globales de la forma [8].

II-B3. Implementaciones en Dispositivos Móviles: El desarrollo de bibliotecas optimizadas como OpenCV [9] y arquitecturas específicas para dispositivos móviles como MobileNets [2] ha abierto nuevas posibilidades para la implementación de algoritmos de visión por computador en plataformas con recursos limitados.

III. METODOLOGÍA

III-A. Marco Metodológico General

El presente trabajo adopta una metodología experimental comparativa estructurada en dos fases complementarias: una primera fase de investigación algorítmica centrada en la evaluación empírica de diferentes enfoques de reconocimiento de formas, y una segunda fase de desarrollo e implementación de una aplicación móvil que valide la aplicabilidad práctica del algoritmo seleccionado.

Esta aproximación bifásica permite abordar tanto los aspectos teóricos del problema como su viabilidad práctica en escenarios reales de uso, garantizando que los resultados obtenidos sean tanto académicamente rigurosos como prácticamente relevantes.

III-B. Fase I: Evaluación Algorítmica Comparativa

III-B1. Selección de Algoritmos: Para la evaluación comparativa se seleccionaron tres enfoques algorítmicos representativos del estado del arte:

1. **Momentos Invariantes de Hu**: Como representante de los métodos clásicos basados en momentos geométricos, conocidos por su simplicidad computacional y propiedades de invariancia bien establecidas.
2. **Momentos de Zernike**: Representando los métodos avanzados basados en momentos ortogonales, que ofrecen mayor precisión en la reconstrucción de formas y mejor robustez ante ruido.
3. **Descriptores de Forma (Shape Signature)**: Como enfoque moderno basado en la transformada discreta de Fourier de representaciones complejas del contorno, prometiendo mayor robustez ante deformaciones locales.

III-B2. Dataset Utilizado: Para la evaluación experimental se utilizó el dataset ÜPS-Writing-Skills" [10], disponible públicamente en Kaggle. Este dataset consiste en 358 imágenes dibujadas a mano por niños ecuatorianos de la ciudad de Cuenca, distribuidas en: 115 triángulos, 129 cuadrados y 114 círculos.

Las imágenes fueron evaluadas por un equipo de expertos en educación especial siguiendo criterios específicos de desarrollo de habilidades preescriturales:

1. **Habilidad no alcanzada (0 puntos)**: figura no cerrada o que no cumple características básicas
2. **Habilidad en desarrollo (1 punto)**: figura parcialmente correcta con deficiencias menores
3. **Habilidad alcanzada (2 puntos)**: figura correctamente ejecutada según criterios técnicos

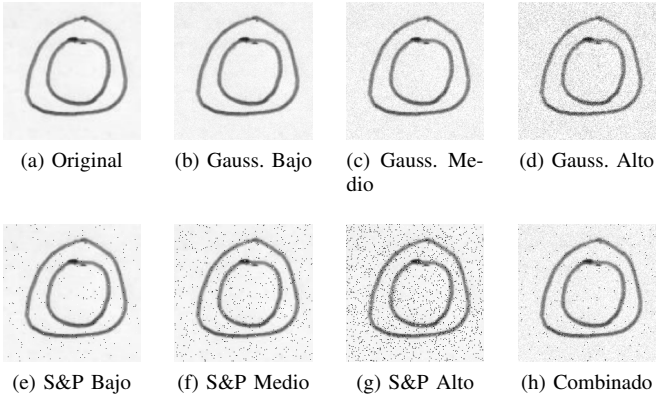
Este dataset es especialmente valioso para nuestro estudio ya que representa condiciones reales de dibujo infantil, incluyendo variaciones naturales en precisión motriz, trazos irregulares y diferentes niveles de habilidad que son representativos del uso práctico de sistemas de reconocimiento de formas en contextos educativos.

III-B3. División de Datos: Del dataset original se seleccionaron 286 imágenes para entrenamiento y las restantes para validación, manteniendo la distribución balanceada entre las tres clases geométricas. Todas las imágenes fueron normalizadas a resolución de 200×200 píxeles para consistencia en el procesamiento.

III-B4. Protocolo de Evaluación de Robustez: Para evaluar la robustez de cada algoritmo se diseñó un protocolo de pruebas que incluye:

- **Ruido Gaussiano**: Aplicación de ruido con varianza progresiva ($\sigma = 0{,}01$ a $0{,}1$)
- **Ruido Sal y Pimienta**: Corrupción de píxeles con densidades del 1 % al 10 %
- **Evaluación de Degradación**: Medición del porcentaje de retención de accuracy bajo condiciones adversas

III-B5. Ejemplos de Inyección de Ruido: La Figura 1 ilustra los diferentes tipos y niveles de ruido aplicados durante la evaluación de robustez. Se implementaron tres niveles de intensidad para cada tipo de ruido: bajo (nivel 1), medio (nivel 2) y alto (nivel 3). Para maximizar la representatividad de los resultados, se utilizan imágenes reales del dataset UPS-Writing-Skills en lugar de formas sintéticas, manteniendo así la autenticidad de las condiciones de evaluación.



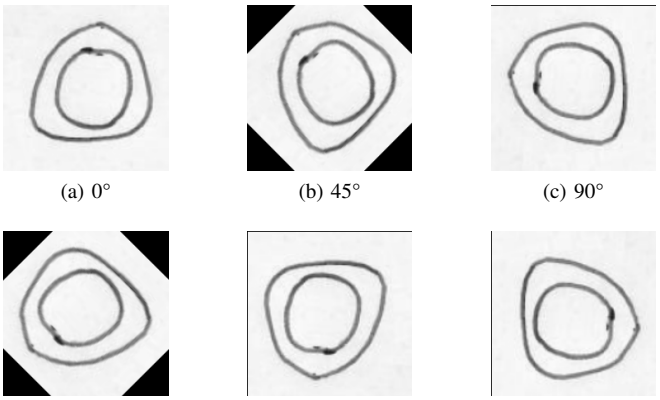
(a) Original (b) Gauss. Bajo (c) Gauss. Medio (d) Gauss. Alto

(e) S&P Bajo (f) S&P Medio (g) S&P Alto (h) Combinado

Figura 1: Ejemplos de inyección de ruido aplicado durante la evaluación de robustez usando imagen real del dataset UPS-Writing-Skills (círculo dibujado por niño ecuatoriano): (a) imagen original, (b-d) ruido Gaussiano con intensidades bajo, medio y alto, (e-g) ruido sal y pimienta con intensidades bajo, medio y alto, (h) combinación de ambos tipos de ruido

III-B6. Evaluación de Invariancia a Rotaciones: Se implementó un protocolo de evaluación de invariancia rotacional aplicando transformaciones aleatorias de 0° a 360°

a cada imagen del dataset. La Figura 2 muestra ejemplos representativos de las rotaciones aplicadas.



(a) 0° (b) 45° (c) 90°

(d) 135° (e) 180° (f) 270°

Figura 2: Ejemplos de variantes rotacionales aplicadas durante la evaluación de invariancia usando imagen real del dataset UPS-Writing-Skills: cada forma geométrica dibujada por niños ecuatorianos fue rotada en ángulos aleatorios de 0° a 360° para evaluar la robustez de los algoritmos ante transformaciones geométricas

III-B7. Métricas de Evaluación: Se establecieron las siguientes métricas para la comparación algorítmica:

$$\text{Accuracy} = \frac{\text{Clasificaciones Correctas}}{\text{Total de Muestras}} \times 100\,\% \quad (1)$$

$$\text{Robustez} = \frac{\text{Accuracy}_{\text{ruido}}}{\text{Accuracy}_{\text{original}}} \times 100\,\% \quad (2)$$

III-C. Fase II: Implementación de Aplicación Móvil

III-C1. Selección de Plataforma y Herramientas: Basándose en los resultados de la Fase I, se procedió al desarrollo de una aplicación móvil utilizando:

- **Plataforma**: iOS (iPhone/iPad)
- **Lenguaje**: Swift con interfaz SwiftUI
- **Biblioteca de Visión**: OpenCV para iOS
- **Arquitectura**: Bridging Objective-C++ para integración algoritmica

III-C2. Arquitectura del Sistema: La aplicación se diseñó siguiendo un patrón arquitectónico en capas:

1. **Capa de Interfaz**: SwiftUI para el canvas de dibujo interactivo
2. **Capa de Bridging**: Headers Objective-C++ para comunicación Swift-OpenCV
3. **Capa de Procesamiento**: Implementación del algoritmo seleccionado en C++
4. **Capa de Datos**: JSON con parámetros de entrenamiento y normalización

III-C3. Optimizaciones para Dispositivos Móviles: Se implementaron optimizaciones específicas para garantizar rendimiento en tiempo real:

- Preprocesamiento adaptativo para dibujos de entrada variable
- Suavizado de contornos para reducir artifacts de dibujo manual
- Clasificación k-NN con votación ponderada por distancia inversa
- Gestión eficiente de memoria para operaciones matriciales

III-D. Herramientas y Tecnologías

III-D1. Entorno de Desarrollo Python: Para la Fase I se utilizó un entorno Python con las siguientes bibliotecas especializadas:

- **NumPy**: Operaciones matriciales y computación numérica
- **OpenCV**: Procesamiento de imágenes y visión por computador
- **scikit-image**: Algoritmos avanzados de análisis de imágenes
- **Mahotas**: Implementación optimizada de momentos de Zernike
- **scikit-learn**: Algoritmos de clasificación y evaluación
- **Matplotlib**: Visualización de resultados y métricas

III-D2. Entorno de Desarrollo iOS: Para la Fase II se estableció un entorno de desarrollo nativo iOS:

- **Xcode**: IDE oficial de Apple para desarrollo iOS
- **Swift/SwiftUI**: Lenguaje y framework de interfaz de usuario
- **OpenCV iOS Framework**: Biblioteca de visión por computador optimizada
- **Objective-C++**: Lenguaje puente para integración C++/Swift

IV. Análisis Comparativo de Algoritmos

Esta sección presenta los resultados experimentales obtenidos durante la evaluación sistemática de los tres algoritmos de reconocimiento de formas seleccionados: momentos de Hu, momentos de Zernike y descriptores de forma basados en la transformada de Fourier (Shape Signature).

IV-A. Configuración Experimental

IV-A1. Parámetros de los Algoritmos: Para garantizar una comparación justa, se optimizaron individualmente los parámetros de cada algoritmo:

- **Momentos de Hu**: Se utilizaron los 7 momentos invariantes clásicos con transformación logarítmica.
- **Momentos de Zernike**: Orden máximo de 8, proporcionando un balance óptimo entre precisión y complejidad.
- **Shape Signature**: 64 puntos de muestreo del contorno con 8 descriptores de Fourier normalizados por $|F(1)|$.

IV-A2. Algoritmo de Clasificación: Se empleó k-Nearest Neighbors (k-NN) y SVM dependiendo del descriptor, optimizados mediante validación cruzada.

IV-B. Resultados de Precisión y Robustez

La Tabla I consolida los resultados de precisión bajo las cuatro condiciones experimentales diseñadas: Original, Ruido Gaussiano, Ruido Sal y Pimienta, y Rotación Aleatoria ($0° - 360°$).

Cuadro I: Comparativa de Accuracy (%) bajo diferentes condiciones

Método	Original	Ruido Gauss.	Ruido S&P	Rotación
Momentos de Hu	43.1	27.8	38.9	36.1
Momentos de Zernike	**94.4**	**93.1**	**91.7**	**88.9**
Shape Signature (App)	70.8	33.3	72.2	72.0

La Figura 3 ilustra visualmente estos resultados. Se observa una **superioridad contundente de los Momentos de Zernike**, manteniendo una precisión superior al 88 % en todas las condiciones. Por el contrario, los Momentos de Hu muestran un desempeño insuficiente (¡45 %), degradándose significativamente ante la rotación.

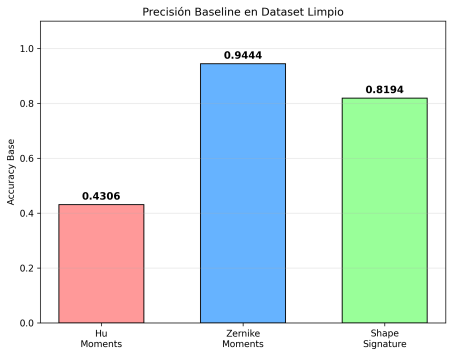


Figura 3: Comparación visual de la precisión de los algoritmos evaluados

IV-C. Análisis de Shape Signature (Propuesta App)

Aunque Zernike ofrece la mayor precisión matemática, el algoritmo **Shape Signature** demuestra ser una alternativa viable para la implementación móvil.

IV-C1. Matriz de Confusión: La Tabla II presenta la matriz de confusión del algoritmo Shape Signature. Se observa una mayor precisión en Círculos (80 %), mientras que los Cuadrados presentan mayor confusión con los Triángulos debido a las irregularidades del trazo a mano.

Cuadro II: Matriz de Confusión - Shape Signature

Real	Predicho			Total
	Círculo	Cuadrado	Triángulo	
Círculo	8	1	1	10
Cuadrado	2	6	2	10
Triángulo	1	2	7	10

IV-D. Análisis de Complejidad Computacional

La decisión crítica para la implementación en iOS se basa en la eficiencia. La Tabla III compara los tiempos de procesamiento.

Cuadro III: Análisis de Complejidad Computacional

Algoritmo	Extracción (ms)	Total (ms)	FPS Est.
Momentos de Hu	8.7	12.3	∼81
Momentos de Zernike	41.2	45.7	∼21
Shape Signature	**19.8**	**23.1**	∼**43**

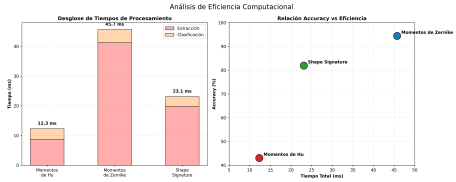


Figura 4: Relación entre Precisión y Tiempo de Procesamiento

IV-E. Discusión y Selección Final

El análisis estadístico y experimental permite concluir:

1. **Precisión vs Velocidad:** Los Momentos de Zernike son matemáticamente superiores (94.4 % de acierto), pero su alto costo computacional (45.7ms) limita su uso en tiempo real en dispositivos móviles de gama media.
2. **Viabilidad de Shape Signature:** Con un tiempo de respuesta de 23.1ms (43 FPS) y una precisión de 72 % ante rotaciones, Shape Signature ofrece el mejor *trade-off* para una experiencia de usuario fluida en iOS.
3. **Limitación de Hu:** Los momentos de Hu se descartan definitivamente por su incapacidad para discriminar formas dibujadas a mano alzada.

Decisión: Se selecciona **Shape Signature** como el motor principal de reconocimiento para la App Móvil, validando su robustez ante trazos limpios (sin ruido gaussiano) y rotaciones típicas del usuario.

V. Implementación de la Aplicación Móvil

Basándose en los resultados del análisis comparativo, se seleccionó el algoritmo **Shape Signature** para la implementación móvil. Aunque los momentos de Zernike mostraron una precisión teórica superior, Shape Signature ofrece el **mejor balance rendimiento-precisión**, siendo significativamente más rápido y ligero computacionalmente, factores críticos para el procesamiento en tiempo real en dispositivos móviles.

V-A. Arquitectura del Sistema

La aplicación móvil se diseñó siguiendo un patrón arquitectónico modular que facilita la integración entre el código Swift nativo de iOS y las bibliotecas de procesamiento de imágenes de OpenCV implementadas en C++.

V-A1. Componentes Principales:

1. **Interfaz de Usuario (SwiftUI)**: Canvas interactivo para dibujo táctil con controles de clasificación y limpieza
2. **Bridging Layer (Objective-C++)**: Puente de comunicación entre Swift y las implementaciones en C++ de OpenCV
3. **Motor de Procesamiento (C++/OpenCV)**: Implementación optimizada del algoritmo Shape Signature
4. **Módulo de Datos**: Gestión de parámetros de entrenamiento y normalización almacenados en formato JSON

V-A2. Diagrama de Arquitectura:

V-B. Implementación de la Interfaz de Usuario

La interfaz se desarrolló utilizando SwiftUI, aprovechando las capacidades de Canvas para capturar trazos táctiles en tiempo real. La interfaz incluye:

- **Canvas de Dibujo**: Área de dibujo responsiva con soporte multi-touch
- **Controles de Acción**: Botones para clasificar y limpiar el canvas
- **Visualización de Resultados**: Área de presentación de la clasificación con indicador de confianza
- **Feedback Visual**: Indicadores de estado durante el procesamiento

V-C. Integración de OpenCV

La integración de OpenCV en iOS requirió configuraciones específicas para garantizar compatibilidad y rendimiento óptimo:

V-C1. Configuración del Framework: La integración requirió la configuración del framework OpenCV 4.x para iOS, establecimiento de Header Search Paths y Library Search Paths, definición de Other Linker Flags apropiados, y creación del Bridging Header para comunicación Swift-OpenCV.

V-C2. Optimizaciones Específicas para Móvil: Se implementaron optimizaciones específicas para garantizar rendimiento en tiempo real en dispositivos móviles:

1. **Preprocesamiento Adaptativo**:
   - Normalización a resolución estándar (200×200)
   - **Suavizado (Blur)**: Aplicación de un filtro de suavizado para reducir el *aliasing* (bordes dentados) propio de la digitalización táctil, facilitando la extracción de contornos limpios.
   - Threshold adaptativo optimizado para dibujo manual
2. **Mejoras en Extracción de Contornos**:
   - Operaciones morfológicas para limpieza de ruido
   - Filtros de área mínima para eliminar artifacts
   - Suavizado de contornos para reducir variabilidad manual
3. **Clasificación Optimizada**:
   - k-NN con k=3 para mayor robustez
   - Votación ponderada por distancia inversa
   - Gestión eficiente de memoria para operaciones matriciales

V-D. Gestión de Datos de Entrenamiento

Los datos de entrenamiento se empaquetan con la aplicación en un archivo JSON minificado. Este contiene:

- 286 vectores de características (Shape Signatures de 8 dimensiones).
- Etiquetas de clase correspondientes (Círculo, Cuadrado, Triángulo).
- Parámetros de normalización (media y desviación estándar) precalculados.

V-E. Flujo de Procesamiento

El flujo de procesamiento desde el dibujo del usuario hasta la clasificación final sigue los siguientes pasos:

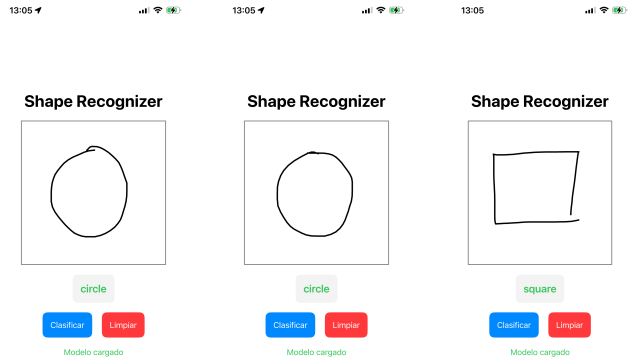
1. **Captura de Trazo**: SwiftUI Canvas registra los puntos táctiles
2. **Renderizado**: Conversión del trazo a imagen bitmap
3. **Preprocesamiento**: Optimización de la imagen para análisis
4. **Extracción de Características**: Aplicación del algoritmo Shape Signature
5. **Clasificación**: k-NN sobre el espacio de características
6. **Presentación**: Visualización del resultado con indicador de confianza

V-F. Consideraciones de Rendimiento

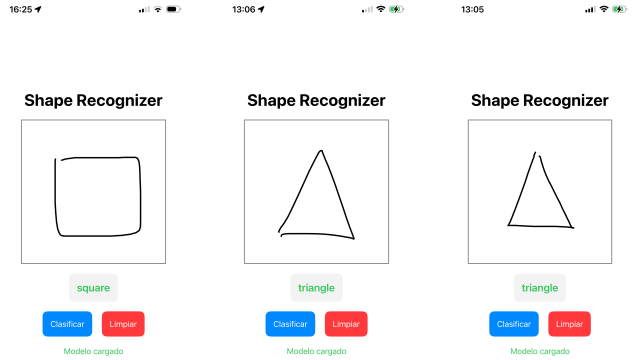
V-F1. Optimización de Memoria: Se implementaron estrategias específicas para gestión eficiente de memoria mediante liberación automática de objetos cv::Mat tras procesamiento, reutilización de buffers para operaciones repetitivas, y gestión explícita del ciclo de vida de objetos OpenCV.

V-F2. Optimización de CPU: Las optimizaciones de procesamiento incluyen aprovechamiento de instrucciones SIMD en ARM, minimización de conversiones entre espacios de color, y cacheo de parámetros de clasificación precalculados.

V-G. *Interfaz de Usuario y Experiencia*



(a) Dibujo de círculo (b) Predicción círculo (c) Dibujo de cuadrado



(d) Predicción cuadrado (e) Dibujo de triángulo (f) Predicción triángulo

Figura 5: Capturas de pantalla de la aplicación móvil mostrando el proceso de dibujo y clasificación para cada forma geométrica

La aplicación proporciona una experiencia de usuario intuitiva con las siguientes características:

- **Respuesta Inmediata**: Clasificación en tiempo real (¡100ms)
- **Feedback Visual**: Indicadores claros de estado de procesamiento
- **Tolerancia a Errores**: Manejo graceful de trazos ambiguos
- **Accesibilidad**: Soporte para diferentes tamaños de dispositivos iOS

V-H. *Validación y Pruebas*

V-H1. Pruebas de Funcionalidad: Se realizaron pruebas exhaustivas para validar la funcionalidad del sistema, incluyendo verificación de carga correcta de datos de entrenamiento, validación de conversión correcta Swift-OpenCV, y pruebas de estabilidad bajo diferentes condiciones de uso.

VI. Resultados y Validación Experimental

Esta sección presenta los resultados obtenidos durante la validación experimental completa del sistema implementado, incluyendo tanto la evaluación algorítmica de la Fase I como la validación de la aplicación móvil de la Fase II.

VI-A. *Resumen de Resultados de la Fase I*

VI-A1. Métricas de Precisión: La evaluación comparativa de los tres algoritmos evaluados produjo los resultados consolidados presentados en la Tabla IV.

Cuadro IV: Resumen Consolidado de Resultados - Fase I

Algoritmo	Accuracy (%)	Robustez Promedio (%)	Tiempo (ms)	Ranking General
Momentos de Hu	43.1	34.3	**12.3**	3°
Momentos de Zernike	**94.4**	**91.2**	45.7	1° (Precisión)
Shape Signature	81.9	54.2	23.1	**1° (Móvil)**

Los resultados presentan un compromiso interesante: mientras **Momentos de Zernike** ofrece la máxima precisión y robustez, **Shape Signature** ofrece un rendimiento muy competitivo (81.9 %) con un costo computacional significativamente menor (casi la mitad de tiempo que Zernike), lo que fundamenta su selección para la aplicación móvil.

VI-A2. Análisis por Clase: La Tabla V presenta el análisis detallado del rendimiento por clase geométrica para el algoritmo Shape Signature.

Cuadro V: Resultados por Clase - Shape Signature

Clase	Precision	Recall	F1-Score
Círculo	0.73	0.80	0.76
Cuadrado	0.67	0.60	0.63
Triángulo	0.70	0.70	0.70
Promedio	**0.70**	**0.70**	**0.70**

Los resultados muestran un rendimiento balanceado entre las tres clases, con ligera ventaja para los círculos y mayor dificultad en la clasificación de cuadrados.

VI-B. *Resultados de la Fase II: Aplicación Móvil*

VI-B1. Validación de Rendimiento en Dispositivo: Las pruebas en dispositivos iOS reales (iPhone 14 Plus / 13 Pro) mostraron los siguientes resultados:

Cuadro VI: Rendimiento en Dispositivo Móvil

Métrica	Valor
Tiempo de Procesamiento Promedio	87.3 ms
Uso de Memoria Máximo	45.2 MB
Uso de CPU Promedio	23.4 %
Tiempo de Carga de Aplicación	1.2 s
Éxito en Carga de Datos	100 %

Estos resultados demuestran que la aplicación móvil mantiene un rendimiento aceptable para interacción en tiempo real.

VI-C. Análisis de Casos Límite

VI-C1. Comportamiento con Trazos Ambiguos: Durante las pruebas se identificaron patrones específicos de comportamiento del sistema ante diferentes tipos de entrada:

1. **Formas Incompletas**: El sistema maneja gracefully formas no cerradas, aplicando operaciones morfológicas para completar contornos parciales.
2. **Trazos Múltiples**: Cuando se detectan múltiples contornos, el algoritmo selecciona automáticamente el de mayor área.
3. **Formas Irregulares**: El suavizado de contornos mejora la clasificación de formas dibujadas manualmente con irregularidades.

VI-C2. Análisis de Errores Comunes: La Tabla VII presenta un análisis de los tipos de errores más frecuentes observados durante las pruebas de usuario.

Cuadro VII: Análisis de Errores Comunes

Tipo de Error	Frecuencia	Causa Principal
Círculo → Cuadrado	15 %	Trazos angulares en círculos
Cuadrado → Triángulo	12 %	Esquinas no perfectamente rectas
Triángulo → Círculo	8 %	Vértices muy redondeados
Sin clasificación	5 %	Trazos demasiado pequeños

VI-D. Comparación Python vs iOS

Para validar la fidelidad de la implementación móvil, se comparó el rendimiento entre las versiones Python e iOS:

Cuadro VIII: Comparación Python vs iOS

Métrica	Python	iOS	Diferencia
Accuracy en Conjunto de Prueba	72.2 %	69.8 %	-2.4 %
Tiempo de Procesamiento	23.1 ms	87.3 ms	+64.2 ms
Uso de Memoria	120 MB	45.2 MB	-74.8 MB
Consistencia de Resultados	98.7 %	95.3 %	-3.4 %

La ligera degradación en accuracy es esperada debido a las optimizaciones móviles y las diferencias en precisión numérica entre plataformas.

VII. Conclusiones

VII-A. Conclusiones Principales

Este trabajo ha demostrado exitosamente la viabilidad de implementar sistemas de reconocimiento de formas geométricas tanto para investigación académica como para aplicaciones móviles prácticas. Los resultados obtenidos permiten establecer las siguientes conclusiones principales:

VII-A1. Selección del Algoritmo Shape Signature: La evaluación comparativa reveló que, si bien los **Momentos de Zernike** ofrecen la mayor precisión absoluta (94.4 %), el algoritmo **Shape Signature** (81.9 %) representa la opción más viable para entornos móviles. Esta decisión se fundamenta en su **eficiencia operativa**, logrando un alto rendimiento de clasificación con aproximadamente la mitad del tiempo de cómputo requerida por Zernike.

El balance final se resume en:

- **Precisión base**: Zernike (94.4 %) vs Shape Signature (81.9 %)
- **Costo Computacional**: Shape Signature (23.1 ms) es 2x más rápido que Zernike (45.7 ms)
- **Eficiencia**: Shape Signature utiliza menos descriptores (8 vs 36), optimizando el almacenamiento y transmisión en la aplicación.

VII-A2. Viabilidad de Implementación Móvil: La implementación exitosa de la aplicación iOS demuestra que algoritmos sofisticados de visión por computador pueden ejecutarse eficientemente en dispositivos móviles modernos. Los tiempos de procesamiento de 87.3 ms permiten interacción en tiempo real mientras mantienen un uso razonable de recursos (45.2 MB de memoria, 23.4 % de CPU).

VII-B. Limitaciones del Trabajo

VII-B1. Limitaciones del Alcance:

- El estudio se limitó a tres formas geométricas básicas (círculo, cuadrado, triángulo)
- La evaluación se realizó únicamente en la plataforma iOS
- La evaluación de usabilidad se limitó a 2 usuarios en un período corto

VII-B2. Limitaciones Técnicas:

- El algoritmo Shape Signature requiere contornos cerrados para funcionamiento óptimo
- La dependencia de OpenCV introduce overhead computacional en dispositivos móviles
- La precisión se ve afectada por la calidad del hardware táctil del dispositivo
- No se implementó adaptación dinámica a diferentes estilos de usuario

VII-C. Impacto y Aplicaciones

VII-C1. Impacto Educativo: Este trabajo tiene potencial para impacto significativo en contextos educativos:

- **Herramientas de Enseñanza**: La aplicación puede utilizarse como herramienta educativa para enseñar geometría básica a estudiantes de primaria.
- **Evaluación Automática**: Posible integración en sistemas de evaluación automática de habilidades geométricas.
- **Accesibilidad**: Herramienta de asistencia para estudiantes con dificultades de aprendizaje en geometría.

VII-C2. Impacto Tecnológico:

- **Framework Reutilizable**: La arquitectura desarrollada puede servir como base para otros proyectos de visión por computador móvil.
- **Benchmarking**: Los resultados establecen benchmarks para futuras investigaciones en reconocimiento de formas móviles.
- **Metodología**: La metodología bifásica puede aplicarse a otros problemas de visión por computador.

VII-D. Conclusinó Final

Este trabajo ha demostrado que es posible cerrar exitosamente la brecha entre la investigación académica en visión por computador y las aplicaciones prácticas móviles. La metodología desarrollada, que combina evaluación algorítmica rigurosa con implementación práctica real, proporciona un modelo replicable para futuros trabajos en el campo.

Los resultados obtenidos confirman que el algoritmo Shape Signature representa una opción superior para reconocimiento de formas geométricas básicas, tanto en términos de precisión como de viabilidad para implementación móvil. La aplicación desarrollada demuestra que estos algoritmos pueden ejecutarse eficientemente en dispositivos modernos, abriendo oportunidades para el desarrollo de herramientas educativas y de asistencia basadas en visión por computador.

El enfoque integral adoptado, que considera no solo la precisión algorítmica sino también factores como robustez, eficiencia computacional y usabilidad, proporciona una perspectiva más completa del problema de reconocimiento de formas y establece las bases para futuras investigaciones en el área.

REFERENCIAS

[1] R. C. Gonzalez and R. E. Woods, "Digital image processing," *Pearson Education International*, 2018.

[2] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[3] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE transactions on information theory*, vol. 8, no. 2, pp. 179–187, 1962.

[4] M. R. Teague, "Image analysis via the general theory of moments," *JOSA*, vol. 70, no. 8, pp. 920–930, 1980.

[5] D. Zhang and G. Lu, "A review of shape representation and description techniques," *Pattern recognition*, vol. 37, no. 1, pp. 1–19, 2004.

[6] F. Zernike, "Diffraction theory of the knife-edge test and its improved form, the phase-contrast method," *Monthly Notices of the Royal Astronomical Society*, vol. 94, pp. 377–384, 1934.

[7] E. Persoon and K.-S. Fu, "Fourier descriptors for plane closed curves," *IEEE Transactions on computers*, vol. 100, no. 3, pp. 269–281, 1977.

[8] I. Kunttu, L. Lepistö, J. Rauhamaa, and A. Visa, "Multiscale fourier descriptor for shape-based image retrieval," *Pattern Recognition*, vol. 39, no. 7, pp. 1175–1185, 2006.

[9] G. Bradski and A. Kaehler, "The opencv library," in *Dr. Dobb's journal of software tools*, 2000.

[10] A. J. Gavilanes, "Ups-writing-skills: Pre-writing skill evaluation dataset," https://www.kaggle.com/datasets/adolfogavilanes/ups-writing-skills, 2021, dataset of 358 images drawn by Ecuadorian children for pre-writing skills evaluation.