Clasificador Knn y segmentación por reconocimiento de bordes basada en K-means

Juan Manuel Alvarado .
E-mail: jalvarados1900@alumno.ipn.mx

Abstract—In the project titled "Knn Classifier and Edge Recognition-Based Segmentation Using K-means," a novel image classification approach is presented, integrating the Knn (k-nearest neighbors) classifier with K-means based segmentation. Employing Sobel filters for edge detection and Hu moments for shape analysis, the model effectively distinguishes between images of washers and coins, objects that are challenging to differentiate due to their similar circular shapes. This approach is particularly valuable in industrial sorting systems, requiring precise classification. The results demonstrate high accuracy in classifying these geometrically similar items, showcasing the efficacy of combining edge recognition with Knn classification in complex image processing tasks.

Index Terms—Knn Classifier, Edge Recognition ,K-means Segmentation ,Machine Learning, Washer and Coins, C++.

1 INTRODUCCIÓN

La clasificación y segmentación de imágenes representan desafíos significativos en el campo del procesamiento de imágenes y el aprendizaje automático. Este proyecto introduce un enfoque innovador que combina el Clasificador Knn (k-nearest neighbors) y una técnica de segmentación basada en el reconocimiento de bordes utilizando el algoritmo K-means. Con el objetivo de diferenciar con precisión entre imágenes de rondanas y monedas, dos objetos con formas circulares similares, se aplica un filtro de Sobel para la detección efectiva de bordes, complementado con momentos de Hu para una análisis detallado de la forma. Esta metodología no solo mejora la precisión en la clasificación de imágenes, sino que también tiene aplicaciones prácticas en sistemas de clasificación industrial, donde la distinción rápida y precisa entre estos objetos es fundamental. Nuestros resultados demuestran la eficacia de esta combinación de técnicas en la resolución de tareas complejas de procesamiento de imágenes.

Enero 01, 2024

PLACE PHOTO HERE

Juan Alvarado Estudiante de Ingeniería en Inteligencia
Artificial en la Escuela Superior de Cómputo .

Operador de Sobel

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A$$

Momentos de Hu$_i$

$$\mathrm{Hu}_i = f_i(\{\eta_{pq}\}) \tag{1}$$

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{(1+\frac{p+q}{2})}} \tag{2}$$

AGRADECIMIENTOS

Dr. Benjamín Luna Benoso

Alexander Iain Crombie Esquinca
Hernández del Ángel Erick Gustavo

1.1 Referencias

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms," 3rd ed. MIT Press, 2009.
J. E. Rodríguez Rodríguez, E. A. Rojas Blanco, and R. O. Franco Camacho, "Clasificación de datos usando el método k-nn," Facultad Tecnológica, Universidad Distrital Francisco José de Caldas, [Fecha de publicación si está disponible].

2 ANÁLISIS A PRIORI

Algorithm 1 Función de Distancia

function DISTANCIA(pxl: PIXEL, $centroide$: CENTROIDE)
    return $\sqrt{(pxl[0] - centroide.pxl[0])^2 + (pxl[1] - centroide.pxl[1])^2 + (pxl[2] - centroide.pxl[2])^2}$ ▷ $O(1)$
end function

Algorithm 2 Función Asignar a Cluster

function ASIGNARACLUSTER(p: PIXEL, $centroides$: ARRAY OF CENTROIDE, k: INTEGER)
    $indiceMin \leftarrow 0$ ▷ $O(1)$
    $distMin \leftarrow$ MAX_INTEGER ▷ $O(1)$
    for $j \leftarrow 0$ **to** $k - 1$ **do** ▷ $O(k)$
        $dist \leftarrow$ DISTANCIA$(p, centroides[j])$ ▷ $O(1)$
        if $dist < distMin$ **then** ▷ $O(1)$
            $distMin \leftarrow dist$ ▷ $O(1)$
            $indiceMin \leftarrow j$ ▷ $O(1)$
        end if
    end for
    return $indiceMin$ ▷ $O(1)$
end function

Algorithm 3 Procedimiento K-Means

procedure KMEANS($mapa$: MATRIX OF PIXEL, $centroides$: ARRAY OF CENTROIDE, k: INTEGER, n: INTEGER)
    $cambio \leftarrow$ FALSE ▷ $O(1)$
    $iteraciones \leftarrow 0$ ▷ $O(1)$
    $maxIteraciones \leftarrow 10$ ▷ $O(1)$
    INICIALIZAR($centroides$) ▷ $O(k)$
    repeat ▷ Hasta $O(maxIteraciones)$
        $cambio \leftarrow$ FALSE ▷ $O(1)$
        for each $pixel$ **in** $mapa$ **do** ▷ $O(n)$
            $indice \leftarrow$ ASIGNARACLUSTER$(pixel, centroides, k)$ ▷ $O(k)$
            Acumular valores para recalculo de centroides ▷ $O(1)$
            Actualizar color del pixel según centroide asignado ▷ $O(1)$
        end for
        for $i \leftarrow 0$ **to** $k - 1$ **do** ▷ $O(k)$
            if $centroides[i].count > 0$ **then**
                Recalcular posición de centroides ▷ $O(1)$
                if hay cambio en la posición **then**
                $cambio \leftarrow$ TRUE ▷ $O(1)$
                end if
            end if
        end for
        $iteraciones \leftarrow iteraciones + 1$ ▷ $O(1)$
    until $\neg cambio$ **or** $iteraciones \geq maxIteraciones$

end procedure

$\therefore$ **K-Means** $\in \mathbf{O(maxIteraciones \cdot (nk + k))}$

Algorithm 4 Funciones de gradiente

function GRADIENTE($kernel$: MATRIX 3x3, $matrizDeEnteros$: MATRIX OF INTEGER, x: INTEGER, y: INTEGER) $O(1)$
  Expandir $matrizDeEnteros$ $O(xy)$
  Aplicar el $kernel$ sobre la matriz expandida $O(xy)$
  return matriz gradiente
end function

function GRADIENTETOTAL($grad\_x$: MATRIX OF INTEGER, $grad\_y$: MATRIX OF INTEGER, x: INTEGER, y: INTEGER) $O(1)$
  $total \leftarrow$ nueva_matriz(x, y) $O(xy)$
  for cada elemento en $total$ **do** $O(xy)$
    $total[i][j] \leftarrow \sqrt{grad\_x[i][j]^2 + grad\_y[i][j]^2}$ $O(1)$
  end for
  return $total$
end function
$\therefore$ **Gradiente** $\in O(xy)$

Algorithm 5 Dependencias para Object Detection y (KNN)

function CALCULOHUMOMENTS($imagen$: MATRIX OF INTEGER, x: INTEGER, y: INTEGER) $O(xy)$
  Calcular momentos de la imagen $O(xy)$
  Calcular centroides $O(1)$
  Calcular momentos centrales $O(xy)$
  return Calcular y devolver momentos de Hu
end function
$\therefore$ **Complejidad de CalculoHuMoments es** $\in O(xy)$

function DISTANCIAEUCLIDIANA($v1$: ARRAY OF DOUBLE, $v2$: ARRAY OF DOUBLE) $O(n)$
  $suma \leftarrow 0.0$ $O(1)$
  for cada elemento en $v1$ y $v2$ **do** $O(n)$
    $suma \leftarrow suma + (v1[i] - v2[i])^2$ $O(1)$
  end for
  return $\sqrt{suma}$ $O(1)$
end function
$\therefore$ **Complejidad de DistanciaEuclidiana es** $\in O(n)$

function DISTANCIAMINIMA($vector$: ARRAY OF DOUBLE, $features$: ARRAY OF ARRAYS OF DOUBLE) $O(mn)$
  $minDist \leftarrow$ MAX_DOUBLE $O(1)$
  for cada $feature$ en $features$ **do** $O(m)$
    $dist \leftarrow$ DISTANCIAEUCLIDIANA($vector$, $feature$) $O(n)$
    if $dist < minDist$ **then** $O(1)$
      $minDist \leftarrow dist$
    end if
  end for
  return $minDist$
end function
$\therefore$ **Complejidad de DistanciaMinima es** $\in O(mn)$

Algorithm 6 Detector

function DETECTOR(actual: VECTOR of DOUBLE, clase1: Features, clase2: Features) ▷ Total: $O(n)$
  $distMin1 \leftarrow$ DISTANCIAMINIMA($actual, clase1$) ▷ $O(n)$
  $distMin2 \leftarrow$ DISTANCIAMINIMA($actual, clase2$) ▷ $O(n)$
  $distMax1 \leftarrow$ DISTANCIAMAXIMA($actual, clase1$) ▷ $O(n)$
  $distMax2 \leftarrow$ DISTANCIAMAXIMA($actual, clase2$) ▷ $O(n)$
  if $distMin1 < distMin2$ **and** $distMin1 < distMax1$ **then**
    print "PERTENECE A LA CLASE MONEDAS" ▷ O(1)
  else if $distMin2 < distMin1$ **and** $distMin2 < distMax2$ **then**
    print "PERTENECE A LA CLASE RONDANAS" ▷ O(1)
  else
    print "INCIERTO" ▷ O(1)
  end if
end function
Therefore, Big O: $O(n)$

Algorithm 7 Aplicación principal

1: **procedure** APLICACIÓN PRINCIPAL
2: **if** `imagen es válida` **then** ▷ $O(1)$
3: `obtener dimensiones de la imagen` ▷ $O(1)$
4: `leer datos de la imagen` ▷ $O(m \cdot n)$
5: **end if**
6: `convertir datos de imagen a matriz de enteros` ▷ $O(m \cdot n)$
7: `calcular gradiente en dirección x` ▷ $O(m \cdot n)$
8: `calcular gradiente en dirección y` ▷ $O(m \cdot n)$
9: `combinar gradientes para obtener gradiente total` ▷ $O(m \cdot n)$
10: `calcular momentos de Hu a partir del gradiente total` ▷ $O(m \cdot n)$
11: `asignar escala de grises a la imagen basada en el gradiente total` ▷ $O(m \cdot n)$
12: `generar centroides para K-means` ▷ $O(n)$
13: `aplicar K-means a la imagen` ▷ $O(\text{maxIteraciones} \cdot (nk + k))$
14: `escribir la imagen procesada` ▷ $O(m \cdot n)$
15: `detectar clase de objeto en la imagen` ▷ $O(n)$
16: `limpiar variables temporales` ▷ $O(1)$
17: **end procedure**
18:
19:
20:

$$\textbf{Aplicación principal} \in O(\textbf{maxIteraciones} \cdot (nk+k) + m \cdot n) \tag{3}$$

EVIDENCIA

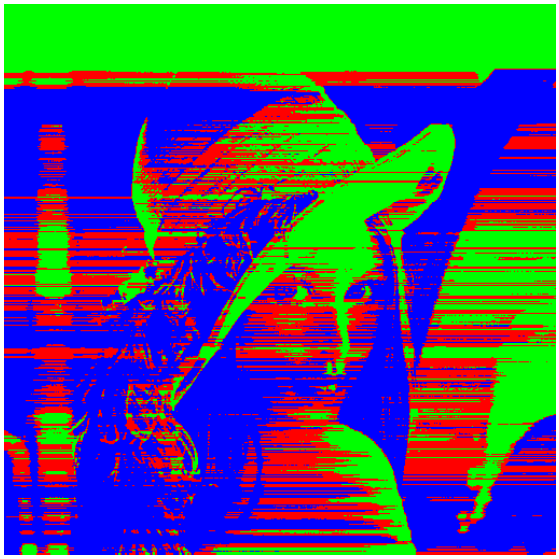

Fig. 1: Imagen de Lenaa segmentada con K-means donde K=3.

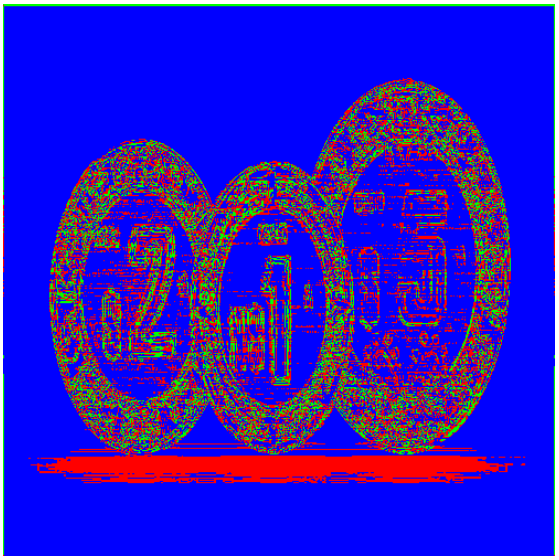

Fig. 3: Imagen con gradiente y BN segmentada.

Fig. 2: Imagen original de monedas sin procesar.

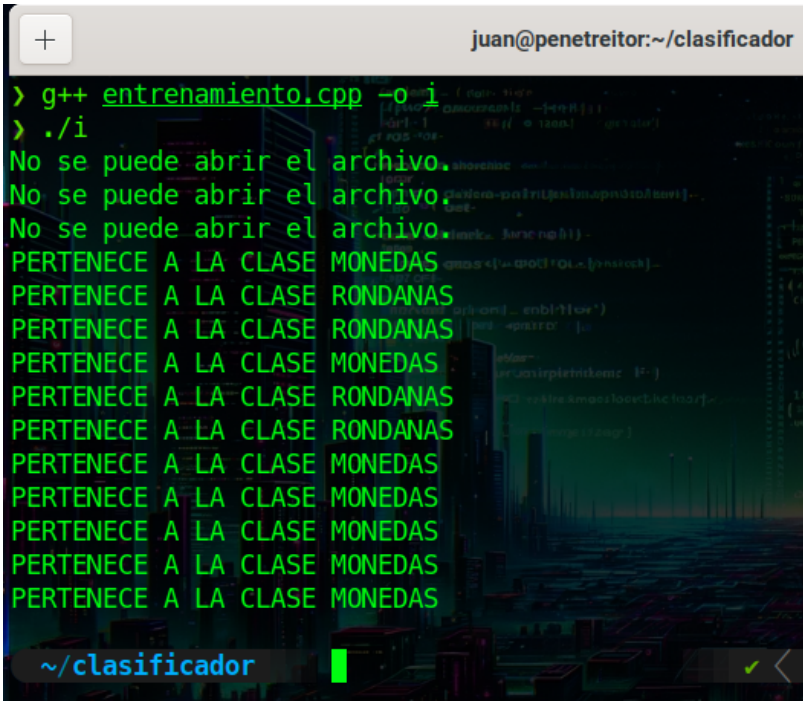

Fig. 4: Salida desde consola del programa.