Procesamiento de Imágenes
Trabajo Practico 5

Representación y Descripción de Características

1. Representación por contornos. Extraer los contornos de los objetos de una imagen binaria y representar la región solo por su borde. Sugerencia en Python: cv2.findContours(), cv2.drawContours().

2. (*) Representación por relleno de regiones. Identificar los objetos en una imagen binaria y colorear cada región detectada. Sugerencia: scikit-image: measure.label, regionprops, label2rgb.

3. Esqueleto de una región. Calcular el esqueleto de una figura en una imagen binaria. Sugerencia: skimage.morphology.skeletonize.

4. (*) Cálculo de propiedades geométricas. Extraer área, perímetro, excentricidad y compacidad de cada región segmentada. Sugerencia: regionprops de skimage.measure.

5. Descriptores de color. Extraer la media y desviación estándar de color (RGB o HSV) para diferentes objetos de una imagen segmentada. Sugerencia: cv2.split() o conversión a HSV + np.mean, np.std.

6. (*) Descriptores de textura con GLCM. Calcular contraste, correlación y homogeneidad de regiones usando matrices de co-ocurrencia. skimage.feature.greycomatrix, greycoprops.

7. Histogramas de color por región. Calcular el histograma de color para cada objeto identificado en una imagen segmentada. Sugerencia: cv2.calcHist() por máscara de cada región.

8. (*) Relación espacial entre regiones. Determinar si las regiones están adyacentes o si una está contenida en otra. skimage.measure.regionprops + análisis de coordenadas / bounding boxes.

Reconocimiento de Patrones (6 ejercicios)

1. (*) Template Matching. Buscar una figura conocida dentro de una imagen mediante una plantilla. Sugerencia: cv2.matchTemplate, cv2.minMaxLoc.

2. (*) Clasificación basada en características. Extraer características simples (como área o textura) de regiones segmentadas y clasificarlas usando KNN. Sugerencia: scikit-learn + descriptores de regionprops.

3. Clasificación con SVM. Usar características geométricas de objetos para entrenar un clasificador SVM y predecir la clase de nuevos objetos. Sugerencia: sklearn.svm.SVC.

4. Clasificación con histogramas de color. Entrenar un clasificador para reconocer frutas en imágenes usando histogramas de color. Sugerencia: cv2.calcHist, sklearn.ensemble.RandomForestClassifier.

5. (*) Reconocimiento estructural. Representar caracteres como grafos de líneas y nodos. Clasificarlos según su estructura. Sugerencia: Estructuras de grafos con networkx (librería de python).

6. (*) Clasificación con CNN (Deep Learning). Construir una red neuronal convolucional para clasificar dígitos (MNIST o similar). Sugerencia: TensorFlow o PyTorch + torchvision.datasets.MNIST.