**IMPLEMENTACIÓN DE KNN EN CLASIFICACIÓN DE IMAGENES**

1. **INTRODUCCIÓN**

El presente informe tiene como objetivo explicar cómo funcionan y se implementan el algoritmo K-Nearest Neighbors (KNN) en el contexto de la clasificación de imágenes, esto aplicado a la detección de señales de tránsito implementado con 4 descriptores distintos: Canny, SIFT, Local Binary Patterns e Histogramas.

1. **PROBLEMA**

El problema de investigación se refiere a la necesidad de desarrollar un sistema automatizado de detección y clasificación de señales de tráfico en la ciudad de Arequipa. Este problema se origina debido a las siguientes razones:

* **Incremento del Tráfico:** La ciudad de Arequipa ha experimentado un aumento significativo en la cantidad de vehículos en las últimas décadas. Este incremento en el tráfico ha llevado a un aumento en la complejidad de la gestión del tráfico.
* **Seguridad Vial:** La identificación y comprensión de las señales de tráfico son esenciales para garantizar la seguridad vial. La falta de reconocimiento de señales de tráfico por parte de los conductores puede dar lugar a infracciones de tráfico y accidentes.
* **Toma de Decisiones Informadas:** Las autoridades locales y los planificadores de tráfico requieren datos precisos sobre la ubicación y el estado de las señales de tráfico en la ciudad. Esta información es crucial para tomar decisiones informadas sobre la gestión del tráfico y la inversión en infraestructura vial.

1. **DEFINICIONES**
   1. **K-Nearest Neighbors (KNN)**

K-Nearest Neighbors es un algoritmo de clasificación supervisada que se basa en la idea de que objetos similares tienden a estar cerca en el espacio de características. Funciona de la siguiente manera:

1. **Entrenamiento:** En la fase de entrenamiento, KNN almacena un conjunto de datos de entrenamiento que consiste en ejemplos con etiquetas conocidas. Cada ejemplo está representado por un vector de características y una etiqueta.
2. **Clasificación:** Cuando se presenta un nuevo ejemplo sin etiquetar para su clasificación, KNN busca los K ejemplos más cercanos en el espacio de características en función de una métrica de distancia (por ejemplo, distancia euclidiana). Los K vecinos más cercanos se determinan calculando la distancia entre el nuevo ejemplo y todos los ejemplos en el conjunto de entrenamiento.
3. **Votación:** Una vez que se han encontrado los K vecinos más cercanos, se realiza una votación entre ellos para determinar la etiqueta de clase del nuevo ejemplo. La etiqueta más común entre los K vecinos se asigna al nuevo ejemplo.
4. **Resultado:** El nuevo ejemplo se clasifica con la etiqueta más frecuente entre sus vecinos más cercanos.
   1. **Algoritmo K-D Tree**

El algoritmo K-D Tree, o árbol k-dimensional, es una estructura de datos utilizada para organizar puntos en un espacio multidimensional. En el contexto de la clasificación de imágenes, los puntos representan descriptores de imágenes (por ejemplo, histogramas o descriptores SIFT) en un espacio de alta dimensión.

El objetivo principal del K-D Tree es acelerar las búsquedas de los vecinos más cercanos en este espacio multidimensional. Para ello, divide recursivamente el espacio en subespacios más pequeños a lo largo de ejes seleccionados, creando una estructura de árbol binario. Esto facilita la búsqueda eficiente de vecinos más cercanos a un punto objetivo.

El árbol se construye de la siguiente manera:

1. **Selección de Dimensión:** En cada nivel del árbol, se selecciona una dimensión diferente del espacio de características.
2. **Ordenamiento:** Los puntos del conjunto de datos se ordenan según el valor de su coordenada en la dimensión actual.
3. **División:** Se divide el conjunto de datos en dos subconjuntos aproximadamente iguales en función de la mediana en la dimensión actual.
4. **Recursión:** Se repiten los pasos anteriores para cada subconjunto, creando así un árbol binario de búsqueda equilibrado.

La ventaja principal de un K-D Tree es que reduce significativamente el número de comparaciones necesarias para encontrar los vecinos más cercanos, ya que descarta automáticamente regiones del espacio que no pueden contener vecinos más cercanos que los ya encontrados.

* 1. **Predictores**

Los predictores son variables o características utilizadas en modelos de aprendizaje automático y estadísticas para hacer predicciones sobre un resultado o variable objetivo. Estos predictores pueden ser variables numéricas o categóricas que se utilizan para estimar o predecir una variable de interés.

* 1. **Predictor Canny (Detector de Bordes Canny)**

El detector de bordes Canny es un algoritmo utilizado en procesamiento de imágenes para detectar bordes en una imagen. Fue desarrollado por John F. Canny en 1986. El algoritmo Canny opera detectando cambios abruptos en la intensidad de los píxeles de una imagen y produce un mapa de bordes resaltando las regiones de la imagen donde se encuentran esos cambios abruptos.

* 1. **SIFT (Scale-Invariant Feature Transform)**

SIFT es un algoritmo utilizado en visión por computadora y reconocimiento de objetos para extraer características distintivas de una imagen que son invariantes a cambios en escala, rotación, traslación y cambios en la iluminación. Estas características se utilizan comúnmente en aplicaciones como la correspondencia de imágenes y la detección de objetos.

* 1. **BLP (Bilateral Filtering)**

El filtrado bilateral es una técnica de procesamiento de imágenes que se utiliza para suavizar una imagen mientras se preservan los bordes y los detalles importantes. A diferencia del filtrado gaussiano, el filtrado bilateral considera tanto la distancia espacial como las diferencias de intensidad para lograr un suavizado controlado.

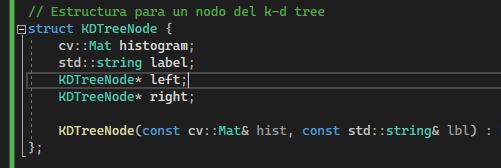
* 1. **Histogramas**

En el contexto del procesamiento de imágenes, un histograma es una representación gráfica de la distribución de intensidades de píxeles en una imagen. Muestra cuántos píxeles de la imagen tienen una intensidad particular. Los histogramas son útiles para comprender la distribución de los niveles de gris en una imagen y se utilizan en diversas aplicaciones, como la ecualización de histogramas para mejorar el contraste de las imágenes.

1. **IMPLEMENTACIÓN EN CÓDIGO DE KNN**

Se ha implementado una versión simplificada de KNN en código (C++). En este caso, se utiliza un conjunto de datos de imágenes de señales de tráfico, las cuales se analizan a través de descriptores, las cuales se utilizan para clasificar nuevas por categorías predefinidas basándose en su similitud con las del conjunto de entrenamiento.

En el código trabajado, el algoritmo K-D Tree se implementa utilizando la siguiente estructura de datos:



Donde:

* descriptor: Almacena el descriptor de la imagen en el nodo del árbol (por ejemplo, un descriptor SIFT o un histograma).
* label: Almacena la etiqueta de la imagen representada por el descriptor.
* left y right: Punteros a los subárboles izquierdo y derecho, respectivamente.

La construcción del árbol se realiza mediante la función *buildKDTree*, que toma un conjunto de datos de imágenes y construye un árbol k-d binario dividiendo los datos en cada nivel en función de una dimensión específica con alta varianza.

La búsqueda del vecino más cercano se realiza mediante la función *searchNearestNeighbor*, que explora recursivamente el árbol para encontrar el nodo más cercano al descriptor objetivo en función de la distancia euclidiana o alguna otra métrica.

La función *kdTreeClassify* realiza una búsqueda en el árbol k-d para encontrar el vecino más cercano al descriptor de la imagen de prueba y devuelve la etiqueta del nodo encontrado como la etiqueta predicha.

1. **FUNCIONAMIENTO DE CADA IMPLEMENTACIÓN DE CÓDIGO**
2. **Función Principal:**

El código se inicia en la función main. Su propósito es clasificar imágenes de señales de tráfico y evaluar el rendimiento del sistema.

1. **Construcción del Árbol K-D Tree:**

Se generan descriptores Canny, SIFT, Local Binary Patterns e Histogramas para cada implementación de código, los cuales se conectarán con un conjunto de imágenes de entrenamiento. El árbol K-D Tree organiza los descriptores en una estructura eficiente para búsquedas rápidas.

1. **Clasificación de Imágenes de Prueba:**

El programa carga imágenes de prueba y genera descriptores para cada una.

Utiliza el árbol K-D Tree para encontrar el descriptor más cercano en el conjunto de entrenamiento para cada imagen de prueba. Esto determina la etiqueta (tipo de señal de tráfico) de la imagen de prueba.

1. **Evaluación del Rendimiento:**

El sistema evalúa su rendimiento calculando métricas de clasificación como precisión, recuperación y puntaje F1.

Estas métricas ayudan a medir cuán precisamente el sistema identifica las señales de tráfico.

1. **Aplicación Potencial:**

Este código se puede utilizar para automatizar la clasificación de señales de tráfico en imágenes, lo que es útil para recopilar datos sobre señales en una ciudad o para tareas de seguridad vial.

También se puede personalizar y ampliar para otros sistemas de reconocimiento de objetos.

1. **CONCLUSIONES**

El código proporcionado demuestra la implementación y el uso de K-D Tree y KNN para la clasificación de imágenes. Estos algoritmos son especialmente útiles cuando se trabaja con datos de dimensiones altas, como descriptores de imágenes, y se desean búsquedas eficientes de vecinos más cercanos. La combinación de K-D Tree y KNN proporciona una forma eficiente de clasificar imágenes en un espacio de características multidimensional.

Sin embargo, es necesario señalar que la calidad de los descriptores utilizados también juega un papel crucial en la precisión de la clasificación.

En resumen, el uso de KD-Tree y KNN en la clasificación de imágenes es una estrategia efectiva que puede adaptarse a una variedad de problemas de visión por computadora y reconocimiento de patrones.