Segmentación de Video basada en Diferencia de Histograma

Algoritmo de Detección de Escenas basada en Diferencia de Histograma

Este algoritmo detecta cambios de escena en un video al calcular la diferencia entre los histogramas de cuadros consecutivos. Un cambio significativo en la diferencia indica un cambio de escena.

1. **Calcular Histogramas de Cuadros:** Sean I_t la imagen (cuadro) en el tiempo t. Calculamos el histograma de la imagen, H_t, que es un vector donde cada componente $H_t(i)$ representa la frecuencia de píxeles en el bin i.

$$H_t = \text{histograma}(I_t)$$

2. **Calcular la Diferencia entre Histogramas:** Calculamos la diferencia D_t entre los histogramas de cuadros consecutivos:

$$D_t = \sum_{i=1}^{N} |H_t(i) - H_{t-1}(i)|$$

   donde N es el número de bins en el histograma.

3. **Detectar Cambios de Escena:** Definimos un umbral T y detectamos un cambio de escena si D_t supera T:

$$\text{si } D_t > T \text{ entonces hay un cambio de escena en } t$$

Algoritmo

Algorithm 1 Detectar Escenas

```
function DETECT_SCENES(video_path, threshold)
    abrir archivo de video en video_path
    prev_hist = histograma(primer cuadro)
    scene_changes = []
    for cada cuadro en el video
        curr_hist = histograma(cuadro actual)
        diff = sum_{i=1}^{N} |curr_hist[i] - prev_hist[i]|      if diff > threshold then
            scene_changes.append(tiempo actual)
        prev_hist = curr_hist
    return scene_changes
```

Reconstrucción de Video basada en Ordenación y Concatenación

Algoritmo de Ordenación y Concatenación

Este algoritmo reensambla un video a partir de segmentos, ordenándolos según su posición original y concatenándolos para formar el video completo.

1. **Ordenar Segmentos:** Sea $S = \{S_1, S_2, \ldots, S_n\}$ el conjunto de segmentos de video, donde cada segmento S_i tiene una posición original p_i en el video original.

   Ordenamos los segmentos según p_i:

   $$S_{\text{ordenado}} = \text{sort}(S, \text{by } p_i)$$

2. **Concatenar Segmentos:** Concatenamos los segmentos ordenados para formar el video reconstruido V:

   $$V = S_{\text{ordenado}}[1] + S_{\text{ordenado}}[2] + \ldots + S_{\text{ordenado}}[n]$$

   donde + denota la operación de concatenación de video.

Algoritmo

Algorithm 2 Reconstruir Video

```
function REASSEMBLE_VIDEO(segments)
    ordenar segments por posición original
    video = concatenar segments ordenados
    return video
```

Matemáticas del Algoritmo de Interpolación de Transiciones

Algoritmo de Interpolación de Transiciones

Este algoritmo suaviza las transiciones entre segmentos de video interpolando los cuadros finales de un segmento y los cuadros iniciales del siguiente.

1. **Ordenar Segmentos:** Similar al algoritmo de ordenación y concatenación:

   $$S_{\text{ordenado}} = \text{sort}(S, \text{by } p_i)$$

2. **Calcular Interpolación entre Segmentos:** Para suavizar la transición entre dos segmentos S_i y S_{i+1}, interpolamos los cuadros finales de S_i y los cuadros iniciales de S_{i+1}.

   Sea F_i el último cuadro de S_i y I_{i+1} el primer cuadro de S_{i+1}. La interpolación lineal entre estos cuadros para un conjunto de cuadros interpolados $\{C_j\}$ se define como:

   $$C_j = \alpha_j F_i + (1 - \alpha_j) I_{i+1}$$

   donde α_j es un coeficiente de interpolación que varía linealmente de 0 a 1 a lo largo del número de cuadros interpolados.

3. **Concatenar Segmentos con Interpolación:** Concatenamos los segmentos junto con los cuadros interpolados $\{C_j\}$:

   $$V = S_{\text{ordenado}}[1] + \{C_1, C_2, \ldots, C_k\} + S_{\text{ordenado}}[2] + \ldots + S_{\text{ordenado}}[n]$$

   donde k es el número de cuadros interpolados entre cada par de segmentos.