



Hoja de trabajo #1

Nancy Mazariegos - 22513
Santiago Pereira - 22318
Brandon Morales - 22992

Task 1 - Análisis Teórico

1. Como director de un proyecto de conducción autónoma, debe dimensionar el hardware para un nuevo vehículo. El sistema utiliza 8 cámaras que capturan video a resolución 4K UHD (3840 x 2160). Debido a la necesidad de alto rango dinámico (HDR), los sensores operan a 12 bits por píxel (Raw Bayer Pattern) a 60 FPS. Métrica A: Enfocada puramente en el flujo vehicular.

Datos:

$$4K \text{ UHD} = 3840 \times 2160 = 8,294,400 \text{ pixels}$$
$$12 \text{ bits/px} = 1.5 \text{ bytes/px}$$

- a. Calcule el tamaño exacto de una sola imagen (frame) cruda en Megabytes (MB).

a)

$$\text{byte/frame} = 8,294,400 \times 1.5 = 12,441,600$$

$$\text{MB (decimal)} = \frac{12,441,600}{10^6} = 12.4416 \text{ MB} \approx 11.87 \text{ MiB}$$

- b. Calcule el ancho de banda necesario (en Gbps) para transmitir el flujo de las 8 cámaras al procesador central sin compresión.

b)

Ancho de banda para 8 cámaras

$$\text{Por Cámara: } 12,441,600 \times 60 = 746,496,000 \text{ B/s}$$

$$8 \text{ Cámaras: } 746,496,000 \times 8 = 5,971,968,000 \text{ B/s}$$

$$\text{bits} \cdot 8 = 47.775744 \text{ Gbps}$$

c. Si su procesador tiene una memoria RAM reservada de 16 GB exclusivamente para el buffer de video, ¿cuántos segundos de historia puede almacenar antes de empezar a sobrescribir datos?

c)

Historial con 16Gb de RAM

$$\text{Gb Decimal: } \frac{16000000000}{5977968000} = 2.68_s$$

$$\text{Gb Binario: } \frac{17179869184}{5977968000} = 2.87_s$$

d. Basado en su resultado, ¿es viable enviar estos datos "crudos" a la nube en tiempo real usando 5G? Justifique

R// En este caso no es sostenible Gbps de uplink, se necesita procesamiento en el borde de CPU/GPU y compresión.

2. Considere un píxel con valor de intensidad $I_{in}=50$ en una imagen estándar de 8 bits (0–255). Se aplican dos procesos de mejora secuenciales en el siguiente orden:

I. Corrección Gamma con $\gamma=0.5$ (para expandir sombras).

II. Ajuste Lineal con ganancia $\alpha=1.2$ y brillo $\beta=-10$ (para contrastar).

Realice los cálculos en el dominio de flotantes normalizados [0,1] como dicta la buena práctica y convierta a entero de 8 bits solo al final.

$$I_{in} = 50 \quad [0 - 255]$$

1. Normalizar a $[0, 1]$

$$r = \frac{I_{in}}{255} = \frac{50}{255} = 0.1960784314$$

2. Corrección gamma ($\gamma = 0.5$)

$$r' = r^\gamma = \sqrt{0.1960784314}$$

$$r' = 0.4428074428$$

$$r' \approx 0.4428$$

3. Ajuste lineal ($\alpha = 1.2$; $\beta = -10$ niveles)

$$v \approx 0.4428$$

3. Ajuste lineal ($\alpha = 1.2$; $\beta = -10$ niveles)
 * Normalizar β

$$\beta_{\text{normalizado}} = \frac{-10}{255} = -0.0392156$$

$$v^n = \alpha v' + \beta_{\text{normalizado}}$$

$$v^n = (1.2) \times (0.4428074428) + (-0.0392156)$$

$$v^n = 0.49215321532450$$

Clip de $[0,1]$

$$v^n \in [0,1] \Rightarrow \text{no hay clipping}$$

$$I_{\text{out}} = v^n \times 255 =$$

$$= 0.49215321532450 \times 255$$

$$= 125.506 \approx 125$$

uint8

a. Calcule el valor final del píxel I_{out} .

R// I_{out} es 125 (ver imagen anterior)

b. ¿Hubo saturación (clipping) en el proceso?

R// No hubo clipping porque $0.4921532450 \in [0,1]$.

c. Si hubiéramos realizado las operaciones usando uint8 directamente sin convertir a float (truncando decimales en cada paso intermedio), ¿cuál habría sido el error numérico resultante?

1. Operando en enteros

$$\bullet \frac{50}{255} \rightarrow 0$$

$$\bullet \sqrt{0} = 0$$

$$\bullet 1.2 \times 0 - 0 \rightarrow -0 \rightarrow [0, 255]$$

\downarrow
0

Comparar con el resultado Correcto

$$\text{Error numerico} = |125 - 0| = 125 \text{ niveles}$$

R//

3. Usted está programando un robot clasificador de pelotas. Tiene dos objetos: una pelota roja brillante bajo el sol $R_{\text{rgb}}=(255,0,0)$ y la misma pelota roja en una sombra profunda $S_{\text{rgb}}=(50,0,0)$

a. Calcule la distancia entre estos dos colores en el espacio RGB.

$$R_{\text{rgb}} = (255, 0, 0)$$

$$S_{\text{rgb}} = (50, 0, 0)$$

a) Distancia

$$d = \sqrt{(255-50)^2 + (0-0)^2 + (0-0)^2} = \sqrt{205^2}$$

$$d = 205$$

b. Convierta ambos colores al espacio HSV (asuma rangos normalizados $H \in [0,1], S \in [0,1], V \in [0,1]$ para simplificar, sabiendo que el Hue del rojo es 0).

b) Conversión a HSV

$$1. R_n = \frac{R}{255}, G_n = \frac{G}{255}, B_n = \frac{B}{255}$$

$$2. \begin{aligned} \text{Max} &= \max(R_n, G_n, B_n) \\ \text{min} &= \min(R_n, G_n, B_n) \end{aligned}$$

$$3. \Delta = \text{max} - \text{min}$$

$$\min = \min(R_n, G_n, B_n)$$

$$3. \Delta = \max - \min$$

$$V = \max$$

$$\text{Saturación}(s) = \begin{cases} 0, & \text{si } \max = 0 \\ \frac{\Delta}{\max}, & \text{en otro caso} \end{cases}$$

$$\text{Si } \Delta = 0 \Rightarrow H = 0$$

$$\text{Si } \max = R_n: H^\circ = 60 \left[\left(\frac{G_n - B_n}{\Delta} \right) \bmod 6 \right]$$

$$\text{Si } \max = G_n: H^\circ = 60 \left[\left(\frac{B_n - R_n}{\Delta} + 2 \right) \right]$$

$$\text{Si } \max = B_n: H^\circ = 60 \left[\left(\frac{R_n - G_n}{\Delta} + 4 \right) \right]$$

$$\text{Normaliza} \rightarrow [0, 1]: H = \frac{H^\circ}{360}$$

$$\text{Caso 1: } R = (255, 0, 0)$$

$$\cdot R_n = 1.000, G_n = 0, B_n = 0$$

$$\cdot \max = 1, \min = 0, \Delta = 1$$

$$\cdot V = 1$$

$$\cdot s = \frac{\Delta}{\max} = \frac{1}{1} = 1$$

$$R_{HSV} = (H, S, V)$$

$$= (0, 1, 1)$$

$$\text{Caso 2: } S = (50, 0, 0)$$

$$\Rightarrow R_n = \frac{50}{255} = 0.196078, G_n = 0, B_n = 0$$

$$\rightarrow \max = 0.196078, \min = 0, \Delta = 0.196078$$

$$\rightarrow V = \max = 0.196078$$

$$\Rightarrow s = \frac{\Delta}{\max} = 1$$

$$H^\circ = 0 \Rightarrow H = 0$$

$$S_{HSV} = (H, S, V)$$

$$= (0, 1, 0.196078)$$

c. Calcule la diferencia absoluta canal por canal en HSV

$$\begin{aligned}
 c) \quad |\Delta H| &= |0 - 0| = 0 \\
 |\Delta S| &= |1 - 1| = 0 \\
 |\Delta V| &= |1 - 0.196078| = 0.803922 \approx 0.8039 \\
 (|\Delta H|, |\Delta S|, |\Delta V|) &= (0, 0, 0.8039)
 \end{aligned}$$

d. Argumente matemáticamente por qué un algoritmo de agrupación (clustering) simple fallaría en RGB pero funcionaría en HSV para determinar que ambos píxeles pertenecen al mismo objeto "pelota roja".

R// Modelo de luz: mismo objeto con distinta iluminación donde

$$x' = Kx \quad (K \in [0, 1])$$

En RGB, la distancia usada por k-means es

$$d = \|x - Kx\| = |1 - K| \cdot \|x\|$$

para (255,0,0) y $K = 50/255$, $d = (0.804)(255)$ donde el algoritmo los separa. En HSV el pigmento conserva $H(0)$ y $S(1)$ en este caso solo varía $V(1 \text{ vs } 0.196)$. Con una métrica ponderada

$$d_{HSV}^2 = w_H \Delta H^2 + w_S \Delta S^2 + w_V V^2 \text{ con } w_H, w_S \gg w_V$$

la distancia entre ambos es pequeña y el clustering los agrupa como el mismo objeto.

Task 3 - Preguntas Post-Práctica

- 1. En la diapositiva 15 se mencionó que "Iterar píxel a píxel en Python es un Pecado Capital". Explique en términos de gestión de memoria y CPU por qué una operación vectorizada en NumPy es órdenes de magnitud más rápida que un for loop.**

R// En este caso recordemos que NumPy es desarrollado o corre sobre 'C', lo cual de por si lo pone en una clara ventaja para operaciones a nivel de RAM y CPU en comparación con un for loop normal por ejemplo en python, entonces con esto sabemos que es capaz de procesar instrucciones SIMD que procesan muchos pixeles simultáneamente.

- 2. Al visualizar imágenes con matplotlib, ¿qué sucede si olvida que OpenCV carga las imágenes en formato BGR? ¿Cómo se ve visualmente el error?**

R// En pocas palabras que todos los tonos de color se ven invertidos, de tal manera que los colores calidos se perciben frios y los frios calidos, esto debido a que los 3 canales principales tambien son intercambiados entre si.

Link a repositorio: <https://github.com/BrandonReyes0609/VP-H1.git>