

Algoritmo JPEG $\rightarrow$ wiki in inglese

Compressione di immagini: ogni pixel ha un colore descritto da una terna di numeri (da 0 a 255)

Immagine: 3 matrici $i \times j$ (i e j sono le dim. orizz. e vert.)

Facciamo finta che sia una sola matrice (immagine per semplicità in B/N)

Immagine FULL HD con colore a 24 bit $\leadsto$ 6 MB

Prima scomposizione: dividiamo la matrice in matrici 8×8 .

Tutto si riduce a comprimere blocchi 8×8 .

Lo spazio delle matrici 8×8 ha dim. 64, quindi dovrai salvare 64 componenti

Strategia migliore: usare una base diversa nello spazio delle matrici e poi trasmettere solo alcune componenti rispetto a questa base, quelle "più importanti".

Sarebbe comodo avere una base ortogonale, perché calcolo facilmente le componenti.

Idea: trovare operatore A : matrici $\rightarrow$ matrici e che verifichi le ipotesi del teo. spettrale

L'applicazione funziona così: ad ogni matrice M ne associò una nuova sostituendo ogni elemento con la somma di quelli (4) che gli stanno accanto. Bisogna vedere al bordo.

Scendiamo di una dimensione: matrici $\rightsquigarrow$ vettori

A $v =$ sostituire ogni elemento con la somma dei 2 vicini

$$\begin{aligned} f(x) &\rightsquigarrow f(x+R) + f(x-R) \\ &= f(x) + R f'(x) + \frac{1}{2} R^2 f''(x) + f(x) - R f'(x) + \frac{1}{2} R^2 f''(x) \\ &= 2f(x) + R^2 f''(x) \end{aligned}$$

Se cerco gli autovalori, sto cercando di risolvere

$$2f(x) + R^2 f''(x) = \lambda f(x)$$

$$f''(x) = \frac{\lambda - 2}{R^2} f(x)$$

Sui vettori

$\rightarrow$ l'equivalente di $f'(x)$ è fare $v_{i+1} - v_i$

$\rightarrow$ l'equivalente di $f''(x)$ è fare $v_{i+1} + v_{i-1} - 2v_i$

Fare gli autovalori di $f''(x)$ equivale a fare gli autovalori di $v_{i+1} - v_{i-1} = \lambda v_i$.

A livello di matrici, fare

$$\{m_{i,j}\} \rightsquigarrow \{m_{i+1,j} + m_{i-1,j} + m_{i,j+1} + m_{i,j-1} - 4m_{i,j}\}$$

è l'equivalente del Laplaciano

Condizioni di Neumann equivale a riflettere sui lati

$$\begin{array}{c|cccc} & 2 & 4 & -1 & 7 \\ \hline 2 & 2 & 4 & -1 & 7 \\ 5 & 5 & & & \\ 3 & 3 & & & \end{array}$$

La base ortonormale è fatta da 64 matrici i cui elementi sono del tipo

$$\cos\left((2k+1)\frac{\pi}{16}i\right) \cdot \cos\left((2l+1)\frac{\pi}{16}j\right) = \text{elemento } (i,j) \text{ della base}$$

k e l sono gli indici della casella (k,l) della matrice

sono gli indici dell'elemento (i,j) della base

→ corrisponde a $\cos\left(\frac{\pi}{16}i x\right) \cdot \cos\left(\frac{\pi}{16}j y\right) = e_{i,j}(x,y)$

Limitiamoci a $\cos(\alpha(2k+1)) = c_k$

$$c_{k+1} + c_{k-1} = \cos(\alpha(2k+3)) + \cos(\alpha(2k-1))$$

$$= \text{coeff. } \underbrace{\cos(\alpha(2k+1))}_{\text{semisomma}} \cdot \cos(2\alpha)$$

$$= \text{coeff. } c_k$$

quindi il vettore c_k è un autovettore della derivata seconda discreta

Conclusione dell'algoritmo: ho una nuova base costituita da 64 matrici, alcune + importanti di altre (quelle con i e j piccolo sono le più importanti).

A quel p.to, se voglio risparmiare, considero solo le comp. rispetto ai primi elementi.