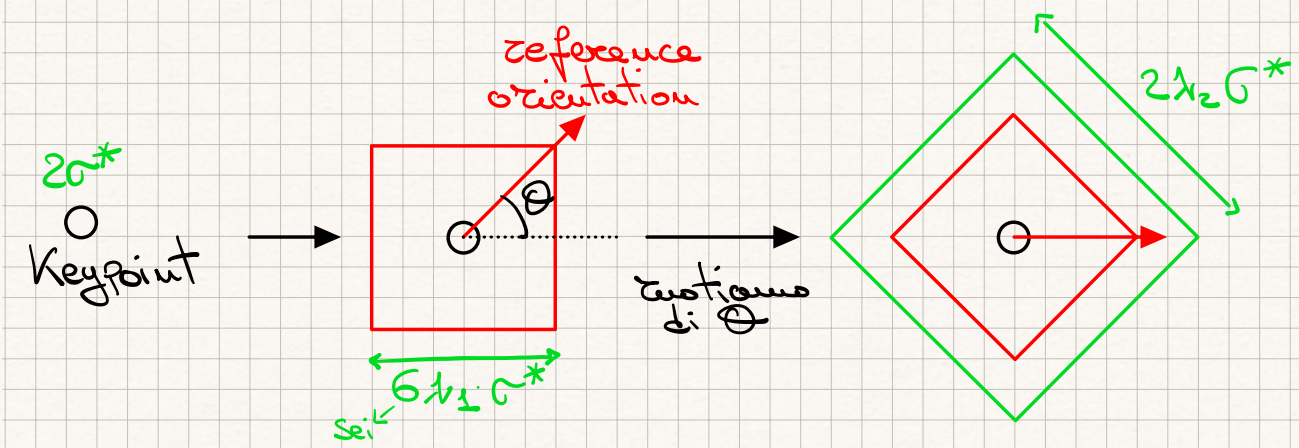
FEATURE DESCRIPTIONS

Ora che abbiamo trovato i keypoint bisogna descriverli in qualche modo.
Qualsiasi sia la descrizione della feature deve essere translation invariant dato che è basata sulle derivate 1° e 2°.
È necessario disegnarla in modo che sia anche scale invariant e rotation invariant.

Trovato σ^* lo usiamo per cercare blobs scale-invariant dato che ispezioniamo versioni ingrandite e ridotte della nostra immagine per cercare σ_1^* nella prima immagine e σ_2^* nella seconda, minimizzando NLOG.
E per quanto riguarda la rotation invariant? SIFT ottiene questa proprietà ruotando un keypoint rispetto a una reference orientation in ogni immagine:



Così ruotati, tutti i patches sono "uguali" nel senso che vengono riportati alla stessa direzione

Quindi, abbiamo un keypoint nella scala Gaussiana (x^*, y^*, σ^*). Estraiamo un "normalized patch" Por dall'immagine $f[x, y, \sigma^*] = g[x, y, \sigma^*] * f[x, y]$:

$$Por = \{(x,y) \in f[x, y, \sigma^*] : \max(|x^* - x|, |y^* - y|) \leq 3\lambda_1\sigma^*\}$$

tipicamente $\lambda_1 = 1.5$. Il Por è un set di pixel che differiscono da (x^*, y^*) meno della soglia $3\lambda_1\sigma^*$.

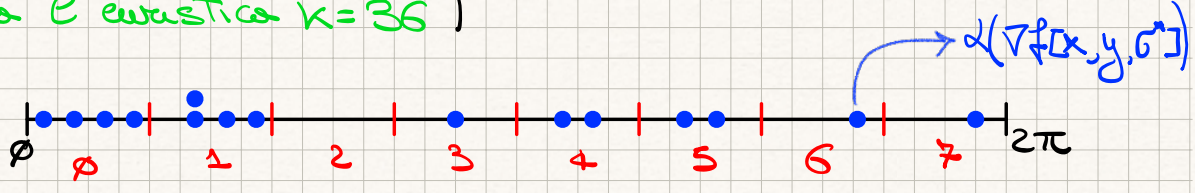
Dato Por, calcoliamo la reference orientation: Usiamo il gradiente che ci dice dove cercare gli edge, e, dato che ci saranno diversi gradienti e diverse direzioni, prendiamo la più comune. Calcoliamo quindi $\nabla f[x, y, \sigma^*]$ e costruiamo l'orientation histogram H

$$H = \{(\emptyset, w_\emptyset), (1, w_1), \ldots, (K-1, w_{K-1})\}$$

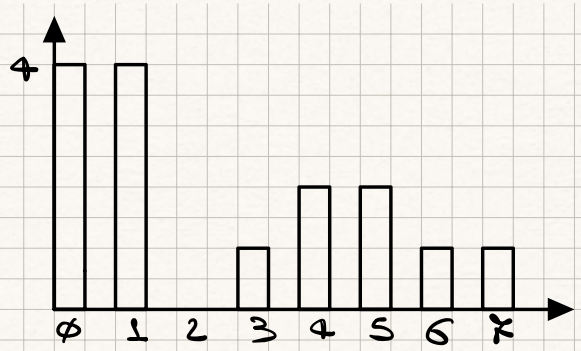
Dove K è il numero di bins usati per discretizzare 2π. Per ogni vettore gradiente calcoliamo $\alpha(\nabla f[x, y, \sigma^*])$, ovvero uno per ogni pixel del Por.

Fase o angolo del gradiente

Ora ordiniamo dal più piccolo al più grande e prendiamo K bins: (Nell'esempio K=8 bins. Solitamente si usa l'euristica K=36)



$$H = \{(\emptyset, 4)\ (1, 4)\ (2, \emptyset)\ (3, 1) \ldots (7, 1)\}$$



A ogni pixel $(x, y) \in P_{or}$, viene assegnato il bin k-esimo con k pari a:

Numero di bin

Bin k-esimo

$$k = \left\lfloor \frac{k}{2\pi} \alpha \left(\nabla f[x, y, \sigma^*] \right) \right\rfloor$$

Ogni pixel contribuisce a formare il peso w_k del k-esimo bin, tramite la magnitudine del gradiente in quel pixel, sumssata utilizzando il filtro gaussiano centrato in (x^*, y^*) con spread $\lambda_1 \sigma^*$.
Iniziando con $w_k = 0$, si aggiornano i pesi usando

$$w_k = w_k + \underbrace{g[x - x^*, y - y^*, \lambda_1 \sigma^*]}_{\text{LowPass filter}} \cdot \underbrace{\|\nabla f[x, y, \sigma^*]\|}_{\text{magnitudine di } \nabla f}$$

più è simile al centro del gradiente e meglio è

Usiamo un filtro gaussiano perché è in grado di dare maggior peso ai punti che presentano $\|\nabla f\|$ maggiore e più vicini al KeyPoint (x^*, y^*). La gaussiana esalta i punti più vicini al suo centro.

Nota: Più la magnitudine si trova vicina al centro del patch di riferimento più contribuisce a far aumentare il peso di quel bin.

RIASSUMENDO

1) Dato il POI, trovare l'istogramma H:

$$H = \{(0, w_0) \dots (k-1, w_{k-1})\}$$

$$w_k = w_k + g[x - x^*, y - y^*, \lambda_1 \sigma^*] \cdot \|\nabla f[x, y, \sigma^*]\|$$

2) Trovato H, calcolare la **reference orientations**, cercando il massimo locale di H:

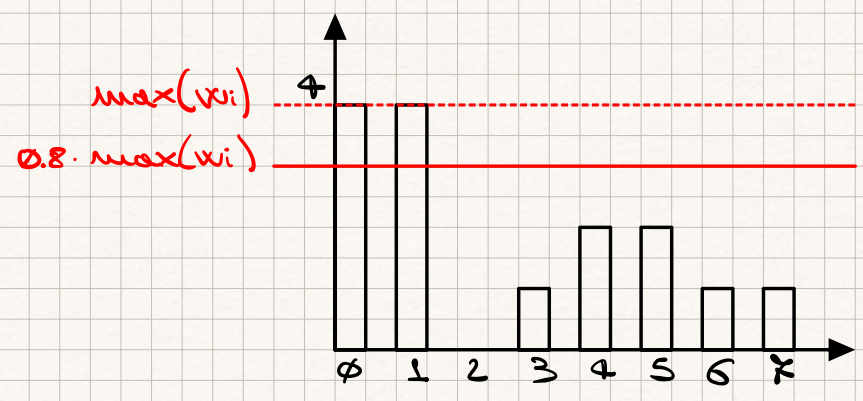
$$k^* \in \{k \in \{0, \dots, k-1\} : w_k \geq \max(w_0, \dots, w_{k-1})\}$$

(bins; max dei pesi)

Troviamo il bin con peso massimo. Prendiamo poi θ^* corrispondente ai pesi che sono almeno t volte più grandi del massimo, con $t = 0.8$.

$$k^{**} \in \{k \in \{0, \dots, k-1\} : w_k \geq t \cdot \max(w_0, \dots, w_{k-1})\}$$

(bins)



Convertiamo ora K^{**} in un angolo (radianti), e prendiamo ogni angolo sopra $t \cdot \max(w_i)$ come una reference orientation.
Procediamo così perché, ognuna di queste reference orientation può essere usata poi per fare il confronto con immagini diverse, e quindi con caratteristiche diverse.

KEYPOINT: n-upla $(x^*, y^*, \sigma^*, \theta^*)$

Prendiamo ora l'immagine $f[x, y, \sigma^*]$, la trasliamo nel punto (x^*, y^*) e la ruotiamo di θ^*. Ogni pixel originale si trasforma: $(x, y) \to (\hat{x}, \hat{y})$:

$$\hat{x} = (x - x^*)\cos\theta^* + (y - y^*)\sin\theta^*$$
$$\hat{y} = -(x - x^*)\sin\theta^* + (y - y^*)\cos\theta^*$$

Difatti è una roto-traslazione con matrice di rotazione:

$$\begin{bmatrix} \cos\theta^* & \sin\theta^* \\ -\sin\theta^* & \cos\theta^* \end{bmatrix}$$

3) Dall'immagine ruotata estraiamo una nuova patch detta P_{desc}:

$$P_{desc} = \{(x, y) \in f[x, y, \theta^*] : \max(|\hat{x}|, |\hat{y}|) < \lambda_2 \sigma^*\}$$

Con $\lambda_2 = 6$, testato empiricamente. Ovviamente non bisogna considerare quei pixel che cadono fuori dal contorno dell'immagine originale.

Ogni pixel $(x, y) \in P_{desc}$ ha:

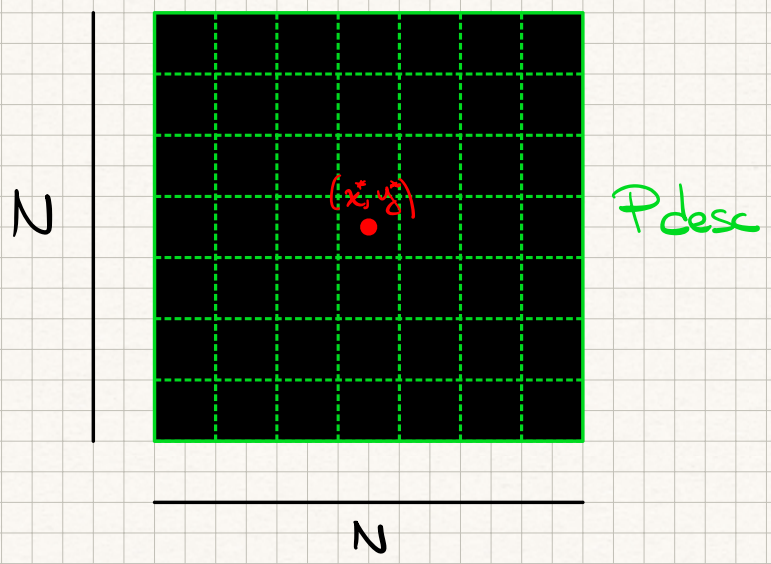
alfa == fase == angolo

$$\hat{\alpha}[x, y] = \alpha[x, y] - \theta^*$$

Definiamo ora $\hat{\gamma}[x, y]$

$$\hat{\gamma}[x, y] = g[x - x^*, y - y^*, \lambda_2 \sigma^*] \cdot \|\nabla f[x, y, \sigma^*]\|$$

Non abbiamo in questa definizione θ^* perché la gaussiana è rotation invariant.



Per ogni pixel $(x, y) \in P_{desc}$ abbiamo calcolato il corrispondente $\hat{\alpha}$ e $\hat{\gamma}$.

- Dividiamo P_{desc} in $N \times N$ subpatches uguali.
- Per ogni subpatches calcoliamo nuovi K_2-bin orientation histogram ($K_2 = 8$).
  Ora, per ogni subpatch abbiamo K_2 nuovi numeri:

  $$(w_0, w_1, \dots, w_7)$$

  Li mettiamo in un vettore.

- Costruiamo il descrittore, impilando nello stesso ordine in cui li troviamo, attraverso Pdesc patches diversi, i vettori dei pesi grandi N^2, in un unico vettore delle caratteristiche $\psi \in \mathbb{R}^e$ con $e = K_2 N^2 = 8 \cdot 4 \cdot 4 = 128$

4) Normalizziamo

a) Per ridurre l'impatto dei cambiamenti di intensità non lineari "saturiamo" ψ come segue

$$\forall i = 1 \dots e \quad \psi_i \leftarrow \max(\psi_i, \underbrace{0.2 \cdot \|\psi\|_2}_{\text{20\% della norma euclidea}})$$

b) Normalizziamo ψ così che $\|\psi\| = 512$ e che ogni componente sia quantizzato su 8 bits

$$\forall i = 1 \dots e \quad \psi_i \leftarrow \min(\lfloor \psi_i \rfloor, 2^8 - 1)$$

Alla fine otteniamo la SIFT feature descriptor per il feature Key Point (x^*, y^*).