Fiche BIMA

Charles Vin

Décembre 2022

Table des matières

Gausienne 2D : $\sigma_f = \frac{1}{\sigma_s \pi}$

$$\frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}}.$$

Changement d'échelle :?

Rotation :

$$P\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}.$$

Changement de variable 2D :

$$f(x,y) = (f_1(u), f_2(t))$$
$$\begin{cases} x = f_1(u) \\ y = f_2(t) \end{cases}$$
$$dxdy = \det(Jacobienne(f))$$

Pour inverser les variables : penser au matrice !

Convolution et dérivation :

$$\frac{\partial}{\partial x}(f \star g) = f \star \frac{\partial g}{\partial x}.$$

Variance K-D :

$$\frac{1}{n}\sum (X_i - g)^T (X_i - g).$$

Généralité :

— Dynamique d'une image : $L = (k_{max} - k_{min}) + 1$
— Changer la dynamique $[k_{min}; k_{max}] \to [I_1, I_2]$: $f(k) = \frac{k-k_{min}}{k_{max}-k_{min}} * (I_2 - I_1) + I_1$
— Inverser une image : $L - p_i$
— Histogramme : compter les pixels de chaque couleurs
— Flat hist : $k' = Int(\frac{k_{max}-k_{min}}{N*M} H_c(k))$
— Gray value profile : line plot de la ligne de l'image

1 Digitalisation and subsampling

SIGNAL pour moi sorry

2 Filtrage

Pense à retourner h pour la convolution !!! Diamond formula

$$f(t)\delta(t - t_0) = f(t_0)\delta(t - t_0).$$
$$f(t) \star \delta(t - \tau) = f(t - \tau).$$

3 Edge Detection with filtering

— Un bord dans une image peut ressembler à une marche d'escalier ou à une rampe : il est plus ou moins nette
— On regarde la direction du gradient : $\|\nabla f\| = \sqrt{(\frac{\delta f}{\delta x})^2 + (\frac{\delta f}{\delta y})^2}$ que l'ont normalise $\frac{\nabla f}{\|\nabla f\|}$ pour obtenir un vecteur unitaire
— Par une méthode mathématique obscure nommée différence finis, on peut approximer les dérivés des images pas une convolution

3.1 Sobel Edge filter

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}.$$
$$G_y = G_x^T.$$

— la réponse impulsionnel de Sobel est en faite composé d'une matrice qui approxime la gaussienne et la matrice de dérivation horizontale $\begin{pmatrix} 1 & 0 & -1 \end{pmatrix}$
— $\|G\| = \sqrt{G_x^2 + G_y^2}$ cette norme est plus forte au niveau des contours (car dérivé d'un escalier $= +\infty$)

3.2 Second order

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \text{ ou } \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

- Ici on regarde quand la dérivée seconde s'annule pour trouver le max de la dérivé
- On utilise un laplacien pour approximer la matrice hessienne $\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$
- Detecter les passages par zéros :
  - Fenetre 3x3 $\rightarrow$ max et min
  - zéro crossing = $max > 0, min < 0, max - min > S$
- Plus précis et moins sensible à la threshold que gradient
- **Pas** invariant par rotation !
- Thick edge
- bruit ++ $\rightarrow$ filtrage nécessaire $\rightarrow$ **On peut combiner les deux en une convolution** avec le laplacien de la gaussienne 2D

3.3 Approche pyramidale

Filtre gaussien $\rightarrow$ subsample 2 $\rightarrow$ filtre $\rightarrow$...

$$\begin{aligned} & fe > 2fmax \text{ (shannon)} \\ \Leftrightarrow & fe > 2 * \frac{3}{\sigma\pi} \\ \Leftrightarrow & \frac{\pi}{6} fe > \frac{1}{\sigma} \\ \Leftrightarrow & \frac{\pi}{6} * \frac{1}{2} > \frac{1}{\sigma} \text{ (1/2 car subsample 2)} \\ \Leftrightarrow & \frac{12}{\pi} < \sigma \end{aligned}$$

3.4 Canny-Deriche

Filtre gaussien plus optimisé + implémentation récursive possible pour éviter de faire deux fois la convolution(x et y)

3.5 Post processing

- Binarization Threshold : thick edge + bruit ou missed detection $\Rightarrow$ Gaussian smoothing
- Gaussian smoothing + Threshold :
  - flou ++ = moins de bruit // thick edge (imprecise localization)
  - Flou – = bruit // bonne localisation
- Non maxima suppression
  - Arrondie sur une des 8 directions
  - Interpolation à partir des deux voisins
  - $\rightarrow$ Bord fin

4 Corner Detection

- Point critique de l'image (local extrema, saddle points) = variation dans une ou plusieurs direction $\Leftrightarrow det(Hess) = 0$. Ca c'est la detection basique mais elle est vraiment pas ouf
- $\rightarrow$ On vas donc jouer avec la matrice Hessienne

4.1 Moravec keypoint detection

$$E_{u,v}(x_1, y_1) = \sum_{k=-W_x/2}^{W_x/2} \sum_{k=-W_y/2}^{W_y/2} [I(x_1 + k + u, y_1 + l + v) - I(x_1 + k, y_1 + l)]^2.$$

Par un DL et développement on tombe sur la suite

4.2 Harris detector

Pour chaque pixel

$$M(x, y) = g_\sigma \star \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}.$$

$$E^1_{u,v}(x, y) = (u, v) M(u, v)^T.$$

Puis on regarde la taille des valeurs propres par la valeur suivante

$$R(x, y) = \det M(x, y) - k(TraceM(x, y))^2.$$

- Région homogène : $R = 0$
- Edge : $R < 0$
- Corner : $R > 0$
- Instinctivement : Dans le voisinage du filtre gaussien, on regarde pour des variations dans deux directions
- Les valeurs propre de la matrice Hessien indique les changement dans leurs direction respective x et y
- Bien pour les scene texturé
- Mauvais sur les changements d'échelles
  - Multiscale Harris detector :
    - **Normalized derivative** Si l'image est scale d'un facteur s on peux aussi scale la gaussienne d'un facteur s. $s * I \star \frac{\partial g_{s\sigma}}{\partial x}(sx, sy)$
    - **Multiscale Harris** $R_{\sigma_I, \sigma_D}(x, y) = \sigma_D^2 g_{\sigma_i} \star \begin{pmatrix} I_x^2(\sigma_D) & I_x(\sigma_D) I_y(\sigma_D) \\ I_x(\sigma_D) I_y(\sigma_D) & I_y^2(\sigma_D) \end{pmatrix}$
    - On peut le stabiliser en changeant l'échelle σ du filtre gaussien en fonction de la scale de ??? Puis en choisissant la meilleure valeur de R comme coin **pas compris** Bref on change la scale et on choisis le meilleur coin sur toutes les scales testées
  - Pareil pour Harris-Laplace : harris laplace = flou gaussien + laplace = on peut combiner comme vu avant, je dirais même plus on peut approximer la combinaison pas une différence de gaussienne

Preuve : On part du détecteur de Moravec : il cherche des changements de variation dans les 8 directions possibles : coin = changement dans deux directions = $E_{u,v}(x, y)$ grands.
Forcément une image full bruit aura beaucoup de variation, donc pour éviter ça on vas lisser l'image avec un filtre gaussien.

Donc on cherche les endroits où $E_{u,v}(x, y)$ est grand aka on veux maximiser la fonctions et trouver les points critique max.

Point mathématique : Pour savoir la nature d'un point (critique normalement mais bon je sais pas où on annule le gradient) on regarde si la matrice Hessienne est définie positive/négative. Pour ça on regarde les valeurs propre, si elle sont strictement négative, c'est gagné on a trouvé un maximum local ! Par chance, on a pas besoin de diagonaliser à chaque fois car le determinant ($= \lambda_1 \lambda_2$) et la trace ($= \lambda_1 + \lambda_2$) sont invariant par changement de base et suffise pour déterminer le signe des valeurs propres.

- $\det Hess > 0$ Valeur propre de même signe
  - $Trace > 0 \rightarrow$ valeur propre positive $\rightarrow$ minimum local
  - $Trace < 0 \rightarrow$ valeur propre negative $\rightarrow$ maximum local

Finalement en posant $R = det - k * trace^2$ on reproduit le même principe. Ce critère permet de comparer λ_1 et λ_2.

- If λ_1 and λ_2 are approximately equal, we note $\lambda_1 \approx \lambda_2 = \lambda$ and get

$$R(x, y) \approx \lambda^2 - k(4\lambda^2) = \lambda^2(1 - 4k).$$

Since in practice k is chosen as a small value, i.e. $k << 1$, we get $R(x, y) \approx \lambda^2$. Then :

  - If $\lambda \to 0$, this means that the derivative of I are close to 0, i.e. the region is flat (homogenous gray levels), and $R \to 0$
  - If $\lambda > 0$ then $R > 0$, and we have locally a corner.
- If $\lambda_1 >> \lambda_2$ (or the reverse) then $R(x, y) \approx \lambda_1\lambda_2 - k\lambda_1^2 = \lambda_1^2(\frac{\lambda_2}{\lambda_1} - k) \approx -k\lambda_1^2$ If the pixel is an edge, it means that the variations of I are in one direction only, i.e. $\lambda_1 >> \lambda_2$, and we get $R < 0$.

□

5 Segmentation

5.1 Optimization based

Osef je pense

- Principe : on vas minimiser une fonction mesurant la distance entre l'image segmenté et l'originale
- On définis l'énergie de l'image segmenté comme somme des distance euclidienne entre chaque pixel de l'image segmentée I et les valeurs moyenne des pixels de l'image originale R
- Par magie mathématique ça donne finalement la somme des valeurs propres de la matrice de variance covariance des couleurs pixel (3x3)
- Facile à optimiser comme un problème de dual + lagrange

5.2 Clustering based

5.2.1 Threshold

- Il existe des méthodes automatique pour choisir une bonne threashold
- **Histograme** on split l'histogramme là ou ça semble logique. Pour chaque partie on trouve la meilleurs threshold qui maximise la taille de la région.
- **K-means** simple, converge, mais hyperparameter k, sensible à l'init, hp cluster sphérique et distinct
- **Mean shift** : Finalement proche d'un k-mean mais au lieux de prendre les tous les points les plus proche, on prend uniquement ceux dans un certain rayon. Puis centroide puis boucle. Une sorte de descente de gradient.

5.3 Slit n Merge

5.3.1 Split

TO DO

5.3.2 Merge

TO DO

6 Image Descriptor

On cherches des caractérisées

- Robustes au changement : rotation, scale, …
- Unique
- Efficace
- représentative : beaucoup de point, petite zone
- Locaux : car plus robuste au background ou bruit et plus modulaire

6.1 Image intégrale

Somme des pixels dans le carré t'as capté. refaire les dessin du TD8 quoi. Si on une rotation par θ fait des dessins avec les centres du pixel comme coordonnée x, y.

6.2 SIFT

— Analyse des keypoints de l'image, méthode proche de Harris, avec prise en compte de la scale
— Pour chaque keypoint, fenêtre autour de lui, direction du gradient pour chaque pixel, histograms pour chaque direction et on garde la direction max
— Feature vector : On prend une fenêtre (16x16) qu'on divise en 4x4 blocs, pour chaque bloc on fait un histogramme de 8 direction. $\rightarrow$ Un vecteur de 128
— **Invariance par rotation** : on normalise le vecteur final par la direction du keypoint ! Pas con pas con

6.3 SURF

Approximation plus rapide de SIFT. En faite c'est le même principe général mais pas vraiment la même méthode mathématique pour trouver l'orientation. De plus le vecteur final n'est pas vraiment du même type. (Ca parle d'ondelette de Hars partout)

6.4 Gabor filter

Détecte l'orientation par ? Pas trop compris
— On peut compiler toutes les convolutions de dérivé gaussienne dans un seule vecteur, qui est invariant par rotation. En faite c'est des arrondies des matrices gaussienne habituelle (SURF utilise pour être plus rapide)

7 PCA

But : réduire les dimensions en gardant uniquement celle qui maximise la variance. C'est à dire les plus explicatives

Variance corrigé projeté :

$$\begin{aligned}\sigma_v^2 &= \frac{1}{n-1}\sum_{i=1}^{n}(\pi(x_i-g))^T(\pi(x_i-g)) \\ &= v^T\Sigma v\end{aligned}$$

$\Sigma = (n-1)^{-1}XX^T, X = (x_1-g, x_2-g, \ldots, x_n-g) = X-g$ matrice de covariance variance des données

$$\begin{cases}\max v^T\Sigma v \\ \text{s.c } v^Tv = 1\end{cases} \Leftrightarrow \Sigma v = \lambda v.$$

Par définition de la diagonalisation : λ valeurs propre et v vecteur propre de Σ
$\Rightarrow$ Composant principal = les vecteurs propres ayant les plus grandes valeurs propre. On peut choisir de garder 95% de la variance par exemple.

8 LDA

On vas projeter sur un axe qui :
— Minimise la variance intra classe (within class)
— Maximise la variance inter classe (between class)

$$
\begin{aligned}
\sigma^2 &= \sigma_w^2 + \sigma_b^2 \\
\sigma_v^2 &= v^T \Sigma v \\
\sigma_b^2 &= \sum_{k=1}^{K} \frac{N_k}{N} \sigma_{k,b}^2 \\
&= \sum_{k=1}^{K} \frac{N_k}{N} (g_k - g)^T (g_k - g) \\
&= \text{Variance pondérée de la moyenne des classe} \\
&= \text{inter class variance} \\
\sigma_{v(b)}^2 &= V^T [\sum_{k=1}^{K} \frac{N_k}{N} (g_k - g)(g_k - g)^T] v = v^T B v \\
\sigma_w^2 &= \sum_{k=1}^{K} \frac{N_k}{N} \sigma_{k,w} \\
&= \sum_{k=1}^{K} \frac{N_k}{N} \frac{1}{N_k} \sum_{X_i \in C_k} (X_i - g_k)^T (X_i - g_k) \\
&= \text{ Moyenne pondéré par } N_k \text{ des variances intra classe} \\
\sigma_{v(b)}^2 &= v^T [\frac{1}{n} \sum_{k=1}^{K} n_k (\frac{1}{n_k} \sum_{x_i \in C_k} (x_i - g_k)(x_k - g_k)^T)] v = v^T W v \\
&\begin{cases} \min \sigma_{v(w)}^2 \\ \max \sigma_{v(b)}^2 \end{cases} \Leftrightarrow \Sigma^{-1} B V = \lambda V
\end{aligned}
$$

Puis on sort $\Sigma^{-1} B$ en fonction des valeurs propres again.
Pour assiger une classe à un nouveau point, on le projette et on regarde le centre de classe le plus proche.

9 Autre truc moins important

9.1 Rappel très très rapide diagonalisation

$$
\begin{aligned}
& f(v) = \lambda v \\
& \Leftrightarrow f(v) - \lambda v = 0 \\
& \Leftrightarrow \det(A - \lambda Id) = 0 \\
& \Leftrightarrow \lambda_1, \ldots, \lambda_k = \lambda \text{ valeurs propres} \\
& AX = \lambda X \Rightarrow X \text{ vecteur propre}
\end{aligned}
$$

9.2 Dérivé de matrice

$$
\begin{aligned}
\frac{\partial(vMv)}{\partial v} &= (M + M^T)V = 2MV \text{ si } M \text{ symétrique} \\
\frac{\partial(v^T a)}{\partial v} &= \frac{\partial(a^T v)}{\partial v} = a \\
\frac{\partial(\log \det M)}{\partial v} &= M^{-1} \\
\frac{\partial(Tr(AM))}{\partial v} &= A
\end{aligned}
$$

9.3 Blob detector

On parcourt l'image (convolution) à différente échelle (variation du σ), si on tombe sur un truc qui resemble à une gaussienne y'a un match sur $\arg\min_{x,y}\max_{\sigma}\Delta[\sigma L(x,y,\sigma)] = \{x_0, y_0, \sigma_0\}$. remember la forme des dérivé d'une gaussienne