Pour Intech

Clément Besnier

27 janvier 2017

Contents

1 Les descripteurs **2**
1.1 Les invariances qu'on peut rechercher . . . 2
1.2 Cadre de travail . . . 2
1.3 Les textures . . . 2
1.4 GIST . . . 3
1.5 Détecteur de coins de Harris . . . 3
1.6 De la transformation Census à mCentrist . . . 3
1.7 Filtre de Gabor . . . 4
1.8 Les histogrammes de gradients . . . 4
1.9 SIFT . . . 4

2 Codage et regroupement **5**
2.1 Modèle de sac de mots . . . 5
2.1.1 La construction . . . 6
2.1.2 Ses limites et solutions . . . 6
2.1.3 Ambiguïté et plausibilité . . . 7
2.2 Détections de points importants . . . 7
2.3 Vecteur spatial de Fisher et VLAD . . . 7
2.3.1 Vecteur spatial de Fisher . . . 7
2.3.2 VLAD . . . 8
2.4 Le codage linéaire sous contrainte de localité . . . 8
2.5 Le codage parcimonieux sous contrainte de localité . . . 8
2.6 Les vecteurs d'attribut moyen-niveau . . . 9

3 Classification **9**
3.1 Forêts aléatoires . . . 10
3.2 Réseaux de neurones . . . 10
3.3 Chaîne de Markov cachée . . . 11
3.4 Chaînes de Markov couple . . . 12
3.5 Chaînes de Markov triplet . . . 12
3.6 Champs de Markov cachée—conditionnels . . . 12
3.7 Segmentation statistique floue . . . 13
3.8 SVM . . . 13
3.9 Adaboost . . . 13
3.10 Réduction de dimension . . . 13
3.10.1 Analyse en composante principale . . . 13
3.10.2 ISOMAP . . . 13
3.11 Modèle à maximum d'entropie . . . 13

1 Les descripteurs

Un descripteur est un vecteur qui par ses valeurs permet de réduire la dimension de l'entrée et augmente la capacité de discriminer l'entrée. Ici on se concentre donc sur comment passer d'un image à un vecteur décrivant l'image. On étudiera dans le même temps les méthodes de rationalisation des descripteurs qui retournent aussi des descripteurs plus globaux. Ces méthodes sont donc entremêlées dans cet exposé.

1.1 Les invariances qu'on peut rechercher

Plus on a des descripteurs précis, moins ils sont généralisables. Invariances par :

- translation
- rotation
- luminosité
- bruit
- homothétie
- affine (homothétie anisotropique)

1.2 Cadre de travail

Dans le domaine de la classification d'images, un cadre de travail récurrent, et formalisé dans [WB07], est le suivant:

- prétraitement de l'image
- extraction du descripteur
- encodage
- regroupement spatial
- normalisation du descripteur

1.3 Les textures

Ces descripteurs décrivent la régularité de la distribution des pixels dans une image. Cela a d'abord été défini dans [Har79] pour les pixels en noir et blanc. D'abord, on calcule d'abord la matrice P de co-occurrences des niveaux de gris avec $P_{i,j}$ égal au nombre de fois que les pixels avec la valeur i et la valeur j sont adjacents. Par exemple, voici trois grandeurs de textures : le contraste, la corrélation et l'énergie. Levels représente le nombre de niveau de gris.

- contraste : $\sum_{i,j=0}^{levels-1} P_{i,j}(i-j)^2$ (normalisé ou non)
- corrélation : $\sum_{i,j=0}^{levels-1} P_{i,j}(\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}}$
- énergie : $\sqrt{\sum_{i,j=0}^{levels-1} P_{i,j}}$
- entropie : $-\sum_{i,j=0}^{levels-1} P_{i,j} log_2(P_{i,j})$
- homogénéité locale : $\sum_{i,j=0}^{levels-1} \frac{P_{i,j}}{1+(i-j)^2}$

1.4 GIST

La première mention de ce descripteur sans le nommer explicitement se trouve dans [OT01]. Ce descripteur permet de capturer des propriétés de l'image qu'un observateur humain donnerait comme le naturel, l'ouverture ou encore la rugosité. L'image est d'abord divisée à l'aide d'une grille 4 x 4 et alors on concatène l'énergie moyenne pour 8 orientations sur 4 distances possibles. Ce descripteur est donc une extension de calculs de textures.

Les caractéristiques données par les humains sont :

- le naturel (naturalness), (est-ce que les constructions sont humaines ou non ?)
- l'ouverture (openness), (est-ce qu'on voit beaucoup de contours ? est ce que la ligne d'horizon est haute et loin ?)
- la perspective (perspective)
- la taille (size), (y a-t-il de grandes structures ? Cela dépend de l'échelle des éléments dans l'image;)
- la plaine diagonale (diagnoal plane)
- la profondeur (depth)
- la symétrie (symmetry)
- le contraste (contrast)

Une implémentation Matlab existe et est fournie par l'auteur. [1]

1.5 Détecteur de coins de Harris

À chaque point de l'image, on associe une matrice de dérive seconde convolué à un filtre gaussien.

$$M(\mathbf{x}, \sigma_I) = G(\mathbf{x}, \sigma_I) * \begin{bmatrix} (\frac{\partial I(\mathbf{x})}{\partial x})^2 & \frac{\partial I(\mathbf{x})}{\partial x} * \frac{\partial I(\mathbf{x})}{\partial y} \\ \frac{\partial I(\mathbf{x})}{\partial x} * \frac{\partial I(\mathbf{x})}{\partial y} & (\frac{\partial I(\mathbf{x})}{\partial y})^2 \end{bmatrix} \quad (1)$$

Avec $G(\mathbf{x}, \sigma_I) = \frac{1}{2\pi\sigma_I^2} * exp(-\frac{x^2+y^2}{2\sigma^2})$.

Un bon critère qui fait qu'un point est un coin ou non est qu'il y ait un fort changement d'intensité lumineuse dans 2 directions orthogonales.

$\Phi(\mathbf{x}) = det(M(\mathbf{x}, \sigma_I)) - \alpha * trace(M(\mathbf{x}, \sigma_I))^2$, avec $\alpha = 0.04$

En ne conservant que les maxima locaux de $\Phi(\mathbf{x})$, on trouve les coins dans l'image. [2] Si un point est le maximum dans une fenêtre autour de ce point (les 8 ou 24 pixels autour) alors il peut être considéré comme un maximum local.

1.6 De la transformation Census à mCentrist

On considère la matrice suivante centrée en un point p0. Pour chaque point d'une image (sauf les bords), on fait correspondre ce point avec p0.

p1	p2	p3
p4	p0	p5
p6	p7	p8

On définit $v_2 = (d_{01}, d_{02}, d_{03}, d_{04}, d_{05}, d_{06}, d_{07}, d_{08})_2$ avec $d_{ij} = \delta(p_i \geq p_j)$. On convertit ce nombre qui est en base 2 vers la base 10 avec $CT = v_{10}$. On obtient alors une image avec la hauteur et la largeur diminuée chacune de deux pixels. La transformée CENTRIST [WR11] calcule l'histogramme de la transformée census sur l'image en niveau de gris. Il existe une transformation plus riche qui utilise plusieurs deux canaux. Cette méthode s'appelle mCENTRIST (multi-channel Census Transform) [XWY14].On compare deux à deux, les canaux d'une image couleur (qui sont 3 plus un canal représentant les contours de l'image). Soient les deux matrices CT^1 et CT^2 dont le centre est p0.

[1] http://people.csail.mit.edu/torralba/code/spatialenvelope/
[2] http://devernay.free.fr/cours/vision/pdf/c4.pdf

p1		p2
	p0	
p3		p4

	p1	
p2	p0	p3
	p4	

$mCT^1 = concat(CT^1_{ch1}, CT^1_{ch2})$
$mCT^2 = concat(CT^2_{ch1}, CT^2_{ch2})$
et ce avec les couleurs hyper-opposées

$$\begin{pmatrix} \frac{R+G+B}{\sqrt{3}} \\ \frac{R+G-2B}{\sqrt{6}} \\ \frac{R-G}{\sqrt{2}} \end{pmatrix}$$

Deux histogrammes mCT^1 et mCT^2 pour deux chaînes. Pour 4 chaînes (R,G,B,SOBEL), il faut 12 histogrammes (2 pour chaque couple de chaînes). Les histogrammes sont calculés sur des parties de l'images : L'image : divisée en 16 rectangles, puis 9 rectangles intérieurs. L'image est divisée en 4 rectangles, puis 1 à l'intérieur, et finalement l'image en entier, ce qui fait 16+9+4+1+1 = 31 rectangles différents où on calcule à l'intérieur de chaque l'histogramme des mCT et on concatène les histogrammes, puis par une ACP (analyse en composantes principales). Cette méthode de division de l'image se nomme "Spatial pyramid" ou la pyramide spatiale.

1.7 Filtre de Gabor

Soit une image I. On définit le filtre de Gabor G 3 x 3 [MBM+13] de la manière suivante. D'abord on définit cette fonction, puis on la convolue avec l'image. pour obtenir J.

$$G(x, y, \theta, f) = exp(-\frac{1}{2}(\frac{x^2_{theta}}{\sigma^2_x} + \frac{y^2_\theta}{\sigma^2_y})) * cos(2\pi f x_\theta) \tag{2}$$

avec $x_\theta = x * cos(\theta) + y * sin(\theta)$ et $y_\theta = y * cos(\theta) - x * sin(\theta)$. Soit $J = G * I$ avec $J(x, y) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} G(k, l, \theta, f) * I(x - k, y - l)$.

1.8 Les histogrammes de gradients

Dans [DT05], il est rappelé comment on construit l'histogramme des gradients [3] :

1. on normalise l'image (le contraste, les couleurs, etc, afin que les résultats soient invariants face à ces grandeurs);
2. on calcule le gradient en chaque point à l'aide du filtre de Sobel dans les directions (Ox) et (Oy) et on combine les résultats dans le vecteurs gradient au point concerné;
3. on partitionne l'image en différentes case, nommons χ l'ensemble des cases;
4. on peut faire un histogramme des angles du gradient avec $arctan(\frac{\frac{\partial I(x,y)}{\partial x}}{\frac{\partial I(x,y)}{\partial y}})$ ou des intensités avec $\sqrt{(\frac{\partial I(x,y)}{\partial x})^2 + (\frac{\partial I(x,y)}{\partial y})^2}$;
4. on concatène les vecteurs et ça fait donc un vecteur caractéristique.

1.9 SIFT

Dans [Low99], un algorithme de création d'un descripteur invariable aux variations de contraste, de luminosité, d'échelle, aux rotations, de translation. Ces propriétés en font un outil très utilisé, même si les conséquences sont les besoins en calculs. Voici les principales étapes de l'application de la méthode SIFT à une image.

[3] https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients

- Construction de la pyramide : on applique à une même image des filtres gaussiens de différentes variances qui varient selon un facteur k^i avec comme valeur initiale σ et i dans {0,...,4} et on répète l'opération sur différentes tailles de l'image (des tailles qui décroissent par 2). On applique ensuite une différence entre ces images lissées en fonction des images qui sont proches en terme de lissage et de taille ($DoG((x,y),\sigma,k,i) = I * (G((x,y)),\sigma k^i) - G((x,y),\sigma k^{i-1}))$). Pour être clair, il y a à chaque taille d'image, une liste d'images filtrées rangées dans l'ordre croissant de i dans $DoG((x,y),\sigma,k,i)$. On peut aussi dire que pour un i donné dans $DoG((x,y),\sigma,k,i)$, il y a plusieurs tailles de la même image filtrée avec le même filtre gaussien.

- détection des extrema : un extremum est un point qui est un extremum pour les images ayant la même taille mais pas filtré avec la même variance, d'où c'est un extremum pour les 8 pixels autour de lui, mais aussi pour les 9 pixels de l'image qui ont été filtrés avec la variance juste au-dessus et ceux avec la variance juste en dessous (8 + 9 + 9 =26).

- localisation des points caractéristiques : pour chaque point-clé candidat, on fait un développement de Taylor à l'ordre 2 autour de ce point de la fonction différence de gaussienne. Cela donne $DoG((x,y),\sigma) = D((x,y)) + \frac{\partial D^T}{\partial x}x + \frac{1}{2}x^T\frac{\partial^2 D}{\partial x^2}x$. En annulant la dérivée de cette équation, on obtient $\hat{x} = \frac{\partial^2 D}{\partial x^2}^{-1}\frac{\partial D}{\partial x}$. Si la différence entre ce résultat et le point évalué est supérieur à 0.5, alors...

- Élimination des points de faible contraste,

- Élimination des points localisés sur des contours de faible courbure : la méthode utilisée ici est semblable à celle utilisée pour le détecteur de Harris.

- Affectation d'une orientation : pour chaque point-clé candidat, on calcule la norme et l'angle en ce point (à l'aide des différences finies en ce point). Autour de chaque point-clé, on calcule un histogramme des orientations avec 36 intervalles sur un voisinage autour de ce point. Chaque point est pondéré par son amplitude et par son facteur d'échelle. D'autres points-clé apparaissent s'ils représentent au moins 80 % d'un intervalle de l'histogramme. Chaque point-clé a donc 4 paramètres : sa position, son facteur d'échelle et son angle

- Calcul des descripteurs locaux : autour de chaque point-clé, on considère un voisinage 16 x 16 pixels, subdivisées en 4 x 4 zones. On calcule dans chaque zone un historgramme des orientations avec 8 intervalles et chaque point est pondéré avec son amplitude. On concatène tous ces histogrammes et on les normalise. On plafonne les valeurs à 0.2 et on normalise à nouveau. On obtient un vecteur de dimension 12829.

Une utilisation dérivée de l'algorithme SIFT, qui vient d'être présentée, est souvent faite dans le contexte de la classification d'images. Cette utilisation est basée sur un calcul du descripteur SIFT, non uniquement sur les points caractéristiques, mais plutôt sur chaque case d'un quadrillage. Cette technique est nommée "dense SIFT" [**?**]. Cela permet d'avoir un nombre fixe de descripteurs SIFT pour une image entière. Cela rend l'apprentissage possible.

Il existe ASIFT, qui est une variante de SIFT, dont la propriété est d'être aussi invariant aux transformations affines. [**?**]

2 Codage et regroupement

2.1 Modèle de sac de mots

La représentation par sac de mots d'images date de plus d'une dizaine d'années [SZ03], [CDF+04]. Cette approche, qui provient du domaine du traitement automatique du langage (TAL), n'est pas aisée en traitement d'images (TIm). En effet, alors qu'en TAL les mots sont très souvent correctement délimités (word segmentation or tokenization), en TIm, les mots ne sont pas une évidence. Les mots sont très souvent des points d'intérêt semblables à d'autres dans une même image ou dans une collection d'images. Par une prise en compte de l'information spatiale [YWY07], [PSV+14], [KBMD15], [HGBRM10], [BBLP10]

Dans Visual word spatial arrangement for image retrieval and classification [PSV+14]

Le but est d'améliorer les techniques de récupération d'image et de classification d'images.

Création d'un dictionnaire visuel On extrait des blocs autour de points d'intérêt des vecteurs d'attribut. On quantifie ces vecteurs d'attributs par un algorithme de regroupement. Chaque regroupement code un mot visuel et on choisit la moyenne de chaque groupe pour représenter ce mot.

Les idées : - le problème de l'approche par sac-de-mot est qu'il manque de l'information spatiale. - Une des solutions est l'usage des pyramides spatiales.

L'idée présentée : l'arrangement spatial des mots avec assignation douce.

Autour de notre point d'intérêt, on compte le nombre de mots présents et lesquels sont dans chaque quadrant de l'image.

Data: $P = \{p_k\}_{k \in 1:N}$ ensemble des points détéctés caractérisé par sa position et le mot w'_k, $k' \in 1 : K$ assigné à ce point

Result: R un vecteur à 4*K composantes, telles que R_i représente la position relative moyenne du mot w_i dans l'image

foreach $p_i \in P$ **do**
  On place un quadrant en p_i
  foreach $p_j \in P / p_j \neq p_i$ **do**
    On incrémente le comptage de w_j à la position où p_j est situé en fonction de p_i.
  end
end

Pour chaque mot, il y a donc 4 valeurs. Si l'une des valeurs est en dessous d'un seuil, alors elle est ramenée à 0. (utile seulement pour les assignations douces). Une idée est avancée est qu'au lieu de prendre tous les points dans une des directions, on pourrait calculer un poids proportionnel à la distance par un noyau gaussien. Le résultat à retenir est que l'assignation spatiale de mots avec la pyramide spatiale permet une meilleure classification. Cependant, l'approche par fenêtre pondérée de l'assignation spatiale de mot donne de moins bons résultats.

Dans le regroupement nécessaire à la création d'un dictionnaire visuel, on peut soit assigner un nouveau point au mot le plus proche, soit assigner plusieurs mots selon une distance pondérée à un certain nombre de mots $a_{i,j} = \frac{K_\sigma(D(v_i,w_j))}{\sum_{l=1}^{k} K_\sigma(D(v_i,w_j))}$ avec D un noyau gaussienne. On garde l'assignation si $a_{i,j}$ est supérieur à un seuil et on rééchelonne suivant les affectations conservées. Cela s'appelle l'assignation douce. (à revoir).

2.1.1 La construction

La construction d'un sac de mots dans des images est faite de la manière suivante [BMM07] :

1. détecter automatiquement les régions et points d'intérêts, (détecteur de Harris, SIFT, SURF ou alors on les calcule selon une pyramide spatiale),
2. calculer des descripteurs spécifiques à ces régions et points d'intérêt (mCentrist, HOG, Gabor, textures, couleurs, formes),
3. quantifier les descripteurs en mots pour créer un vocabulaire (on peut utiliser des algorithmes de classification non supervisés comme k-means, k-medoids, LGB ou ISODATA)
4. trouver les occurrences de chaque mot dans les images pour créer le sac de mot ou l'histogramme des mots.

Par ailleurs [BMM07] fait le point sur la manière de classer des images, mais cela date de 2006. Une autre étude est sortie en 2015 [FS15].

Dans le travail préliminaire à la thèse [FMC15], le travail a été fait de la manière suivante pour la partie haut-niveau

1. et 2. on ne détecte pas les régions d'intérêt mais on calcule dans un quadrillage particulier, la pyramide spatiale, un histogramme des gradients (HOG) ainsi que l'histogramme des couleurs normalisées;
3. le vocabulaire a été créé par un algorithme non-supervisée qui est un k-means dont l'initialisation n'est plus vraiment aléatoire; on a comparé cette méthode a la méthode qui consiste à utilisé un vocabulaire déjà construit et à montrer quelle partie de l'image correspond à tel mot.
4. cette partie est classique.

2.1.2 Ses limites et solutions

Dans [Que07], des limites de ce modèle sont relevées. Les limites de ce type de représentation des mots proviennent de la perte d'information spatiale (en tout cas pour ce qui concerne le modèle le plus simple), et de la dispersion ou concentration de différents sens : un sens porté par plusieurs mots (synonymie) ou plusieurs sens portés par un seul mot (polysémie). Dans le cas où le nombre de classes est très élevé, la tendance sera d'avoir beaucoup de synonymes alors que quand le nombre de classe est faible, le nombre de mots polysémiques sera élevé.

Pour pallier ce dernier problème, on a proposé dans le monde du TAL une méthode probabiliste nommée "latent aspect representation" ou encore "représentation par aspect latent". Ceci permet de prendre en compte les co-occurrences des synonymes et des mots polysémiques. Soit I_i une image et m_j un mot du vocabulaire établi.

$$P(I_i, m_j) = P(I_i) * P(m_j|I_i) = P(I_i) * \sum_{l=1}^{N_A} P(z_l|I_i)P(m_j|z_l) \tag{3}$$

On a introduit dans l'équation précédente une variable latente z_l qui peut représenter un aspect.

2.1.3 Ambiguïté et plausibilité

Attention : ils disent [VGVSG10] qu'un mot représente une image, ce qui n'est pas le cas ailleurs.

Lorsqu'il faut choisir un mot décrivant une image et que plusieurs sont de bon candidats, alors on fait apparaître le fait que les mots ainsi présentés sont ambigus, de même que si pour une image donnée, si on cherche un mot lui correspondant, on cherche à connaître la plausibilité d'un mot représentant l'image. Une propriété intéressante à utiliser est le fait que les mots ayant un sens similaire apparaissent souvent ensemble.

On définit un noyau gaussien $K_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{x^2}{2\sigma^2})$

L'incertitude d'un mot est défini comme :

$$\text{UNC(w)} = \frac{1}{n}\sum_{i=1}^{n} \frac{K_\sigma(D(w, r_i))}{\sum_{j=1}^{|V|} K_\sigma(D(v_j, r_i))} \tag{4}$$

avec n le nombre de régions dans l'image, r_i la région i de l'image et $D(w, r_i)$ la distance entre w et r_i. Et la plausibilité d'un mot :

$$\text{PLA(w)} = \frac{1}{n}\sum_{i=1}^{n} K_\sigma(D(w, r_i))\delta(w = \underset{v \in V}{\text{argmin}} D(v, r_i)) \tag{5}$$

2.2 Détections de points importants

Dans [KB01], l'idée est de calculer l'entropie dans une aire autour du point $\mathbf{x}$ à plusieurs échelles.

$H_D(\sigma, \mathbf{x}) = -\int_{R_I} p(R_I, \sigma, \mathbf{x}) * log_2(p(R_I, \sigma, \mathbf{x}))dR_I$ $\sigma_{pic} = \underset{\sigma}{\text{argmax}} H_D(\sigma, \mathbf{x})$. On calculer l'amplitude des variations de la probabilité $W_D(\sigma, \mathbf{x}) = \frac{\sigma^2}{2\sigma - 1}\int_{R_I} |\frac{\partial p(R_I, \sigma, \mathbf{x})}{\partial \sigma}| dR_I$. L'importance est définie par $H_D(\sigma, \mathbf{x}) * W_D(\sigma, \mathbf{x})$.

2.3 Vecteur spatial de Fisher et VLAD

2.3.1 Vecteur spatial de Fisher

Je reprends ici le raisonnement donnée dans [SPMV13].

Soit $X = (X_t)_{t \in 1:T}$ un échantillon de vecteurs de dimension D dans H. Soit u_λ une densité de probabilité qui modélise la génération des vecteurs de H avec $\lambda = [\lambda_1, ..., \lambda_M]^T \in \Re^M$ un paramètre. Soit $G_\lambda^X = \nabla_\lambda log(u_\lambda(X))$ qui est la fonction score. Cette fonction sert à savoir comment le paramètre devraient être modifiées pour que u_λ modélise au mieux les données X. Soit la matrice d'information de Fisher (qui est une métrique locale de la variété riemannienne $U = \{u_\lambda\}$) $F_\lambda = E[G_\lambda^x (G_\lambda^x)^T]$ avec x qui suit la loi u_λ. Étant une matrice de covariance, F_λ est semi-définie positive, comme l'est son inverse, alors, par la décomposition de Cholesky, donnée par $F_\lambda^{-1} = L_\lambda^T L_\lambda$, on peut définir le noyau de Fisher $K_{FK}(X, Y) = (\Gamma_\lambda^X)^T F_\lambda^{-1} \Gamma_\lambda^Y$ avec $\Gamma_\lambda^X = L_\lambda G_\lambda^X$ qu'on appelle le vecteur de Fisher.

Si les échantillons sont indépendants, alors on peut réécrire le vecteur de Fisher comme étant $\Gamma_\lambda^X = \sum_{t=1}^{T} L_\lambda \nabla_\lambda log(u_\lambda(x_t))$.

On fera l'hypothèse u_λ est mélange de gaussienne notée $\sum_{k=1}^{K} w_k u_k(x)$.

à suivre...

2.3.2 VLAD

Une approche moins complexe mais qui est dans la même lignée du vecteur de Fisher est l'usage du descripteur VLAD (Vector of Locally Aggregated Descriptors) qui est, selon les dires de l'auteur [JPD+12], pour le vecteur de Fisher, la même chose que l'algorithme des k-moyennes est pour le regroupement par mélange de gaussiennes. En effet, l'algorithme des k-moyennes suppose que les gaussiennes sont équiprobables et leur matrice de covariance est diagonale avec des valeurs très petites.

La création du descripteur VLAD se fait de la manière suivante :

1. Données : ensemble de descripteur dans l'image $\{x_i\}_{i\in 1:N}$ avec x_i u vecteur à d composantes.
2. Usage des k-moyennes afin de créer un vocabulaire visuel : $\{\mu_i\}_{i\in 1:K}$
3. Tout nouveau descripteur x_t est associé le plus proche voisin $NN(x_t)$
4. $v_i = \sum_{x_i:NN(x_t)} x_t - \mu_i$
5. Tous les v_i sont concaténés pour former un vecteur avec K*d composantes.
6. On normalise chaque composante par un exposant α ($\alpha = 0.5$ là où c'est présenté) et on divise par la norme L_2 du vecteur.

Il faut noter que ce descripteur est introduit dans le cadre de la recherche d'image et non dans la classification d'images. L'auteur [JPD+12] des cases de taille 4 x 4 pour dense SIFT.

2.4 Le codage linéaire sous contrainte de localité

(Locality-constrained Linear Coding) [YYGH09]

Cette méthode est similaire à la création des mots pour le modèle de sac-de-mots sauf que les descripteurs des sous-images sont encodés par les coefficients de la combinaison linéaire des mots-visuels qui minimisent l'écart quadratique moyen entre ces mots-visuels.

Dans [WYY+10], on présente la méthode de codage linéaire et locale. Soit $\mathbf{X}$ l'ensemble de cardinal N des descripteurs et de dimensions D extrait de l'image I. Soit un dictionnaire B avec M entrées. Chaque élément de l'espace des descripteurs possibles a une entrée correspondante dans V.

Pour trouver les coefficients c_i associés aux entrées de B, on utilise la fonction objectif suivante :

$$c^* = \underset{\mathbf{C}}{\operatorname{argmin}} \sum_{i=1}^{N} || \mathbf{x_i} - \mathbf{B c_i} ||^2 + \lambda || \mathbf{d_i} \odot \mathbf{c_i} ||^2 \tag{6}$$

avec $\mathbf{1^T c_i} = 1$, $\forall i$ et $\mathbf{d_i} = \exp(\frac{dist(\mathbf{x_i},\mathbf{B}) - \max_{\mathbf{j}}(\mathbf{dist}(\mathbf{x_j},\mathbf{B}))}{\sigma})$. On définit $\text{dist}(\mathbf{x_i}, \mathbf{B}) = [\text{dist}(\mathbf{x_i}, \mathbf{b_1}), ..., \text{dist}(\mathbf{x_i}, \mathbf{b_M})]$ et *dist* comme étant la distance euclidienne entre les vecteurs x_i et b_i. Une version optimisée est présentée dans le même papier pour accélérer l'encodage.

2.5 Le codage parcimonieux sous contrainte de localité

(Locality-constrained sparse coding)

Il faut signaler qu'il y a une incohérence dans les notations.

Ce papier [THC+16] utilise un type de codage similaire au précédent. Le but est de trouver une base D de vecteurs $[\phi_1, ..., \phi_C]$ sur laquelle on peut représenter tous les descripteurs fournis par dense SIFT avec $\alpha_{i i\in 1:C}$ tels que $\text{x} = \sum_{i=1}^{C} \phi_i \alpha_i$, $x \in \Re^d$, $D \in \Re^{dxC}$ et $\alpha_i \in \Re^C$

$$\hat{D}, \hat{A} = \underset{D,A}{\operatorname{argmin}} \|x_i - DA\|_2^2 + \lambda \sum_{i=1}^{C} \|l_i \odot DA\|_2^2 \tag{7}$$

avec $l_i = \|x - D\|_2$ et $\mathbf{1}^T \alpha_\mathbf{i} = 1$. Un algorithme itératif est proposé afin de trouver le meilleur dictionnaire possible. Le regroupement spatial est légèrement différent de celui proposé en général. Il est fait en trois couches. Sur chaque couche l, on calcule l'histogramme du descripteur de l'image à un intervalle de 2^l. il est défini le noyau

de la pyramide spatiale $K(\sigma(u), \sigma(v)) = \sum_{l=1}^{L} \omega_l N_l$ avec ω_l qui est le poids au niveau l et N_l qui est le nombre de correspondances dans les vecteurs d'attributs à la couche l : $F(H_u^l, H_v^l) - F(H_u^{l-1}, H_v^{l-1})$. La fonction d'intersection des histogrammes est $F(H_u^l, H_v^l) = \sum_{b=1}^{B} min(H_u^l(b), H_v^l(b))$.

2.6 Les vecteurs d'attribut moyen-niveau

[BBLP10] s'intéresse à deux étapes essentielles dans l'apprentissage de vecteurs d'attribut moyen-niveau : l'encodage et le regroupement spatial. Le premier est l'étape qui permet d'obtenir un dictionnaire de mots visuels. Les deux méthodes principales sont l'assignation (douce ou forte) par quantification des vecteurs d'attributs bas-niveau et par le codage parcimonieux (sparse). Le second est l'étape où l'on regroupe les mots visuels selon un schéma guidé par le voisinage ou par une hiérarchie. Les méthodes retenues sont le regroupement en faisant la moyenne ou en prenant le maximum.

- I : image, $\{x_i\}_{i \in 1:N}$ l'ensemble des descripteurs de bas-niveau, M le nombre de régions d'intérêt, N_m l'ensemble des lieux endroits à l'intérieur de la région m
- f : la fonction d'encodage, $\alpha_i = f(x_i), i \in 1 : N$
- g : la fonction de regroupement, $h_m = g(\{\alpha_i\}), m \in 1 : N$
- z : le vecteur d'attribut de moyen-niveau, $[h_1^T, ..., h_n^T]$

Dans ce cadre-ci, le modèle le plus simple [SZ03] peut se réécrire de la façon suivante :

- f : $\alpha_i \in \{0, 1\}^K, \alpha_{i,j} = \delta(j = \underset{k \in 1:K}{\text{argmin}} \|x_i - d_k\|_2^2)$ avec d_k le d_k^{eme} mot visuel;
- g : $h_m = \frac{1}{|N_m|} \sum_{i \in N_m} \alpha_i$.

Dans [VGVSG10], $\alpha_i \in \{0, 1\}^K, \alpha_{i,j} = \frac{exp(-\beta \|x_i - d_j\|_2^2)}{\sum_{k=1}^{K} exp(-\beta \|x_i - d_j\|_2^2)}$

L'encodage parcimonieux, c'est le fait de faire la combinaison linéaire d'un petit nombre de mots visuels sur dictionnaire pour décrire un mot visuel observé.

- f : $\alpha_i \in \{0, 1\}^K, \alpha_{i,j} = \underset{\alpha}{\text{argmin}} L(\alpha, D) \|x_i - D\alpha\|_2^2 + \lambda \|\alpha\|_1$;
- g : $h_m = \frac{1}{|N_m|} \sum_{i \in N_m} \alpha_i$.

Afin de trouver la meilleure des combinaisons qui aident à mieux classer, les auteurs utilisent un réseau de neurones convolutionnel.

- $\hat{x_i} = [x_{i_1}, ..., x_{i_L}], pour i_1, ..., i_L \in L_i$ pour L_i
- f : $\underset{\alpha}{\text{argmin}} L(\alpha, D) \|\hat{x_i} - D\alpha\|_2^2 + \lambda \|\alpha\|_1$;
- g : $h_m = \frac{1}{|I|} \sum_{i \in N_m} \alpha_i$.

Les détails de l'optimisation du dictionnaire qui sert à coder parcimonieusement les mots visuels sont omis ici. Il faut retenir le codage parcimonieux et le regroupement en ne conservant que le maximum d'un groupe sont le duo gagnant des comparaisons.

3 Classification

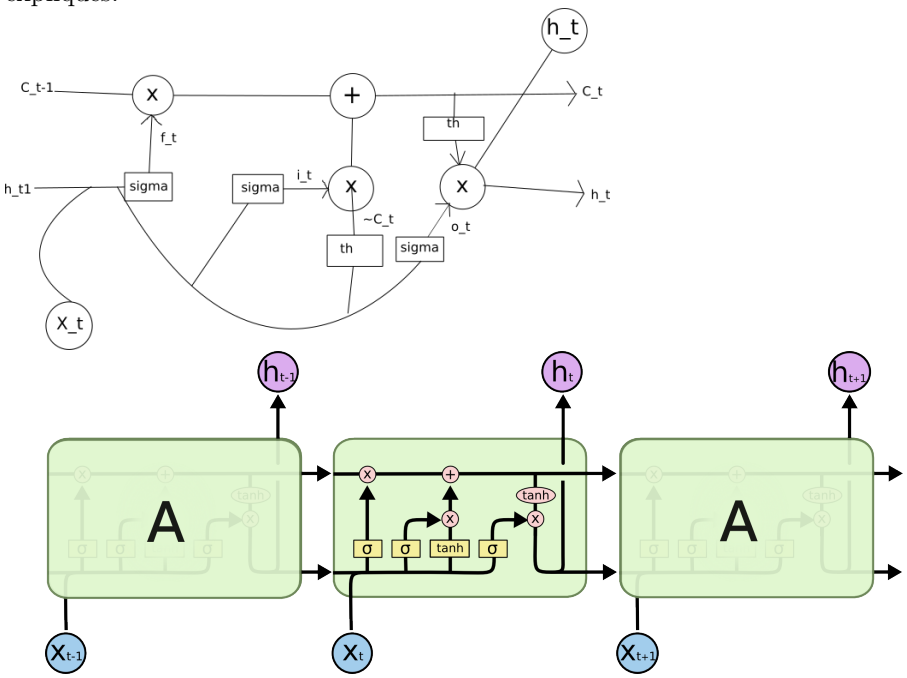
Premier scénario : classer directement par les vecteurs d'attributs observés. La difficulté vient du codage des vecteurs d'attributs. Il est parfois difficile d'avoir une loi de probabilité générative. Deuxième scénario : d'abord segmenter les images en différentes régions (qui correspondent à différentes catégories d'éléments de la route), puis apprendre à classer une image à partir du contenu segmenté. Troisième scénario : lisser les résultats des classements déjà obtenus par un autre algorithme de classement.

3.1 Forêts aléatoires

[Bre01] Les paramètres sont le nombre d'arbres de décisions créés aléatoirement, la profondeur maximale autorisée pour chaque arbre.

3.2 Réseaux de neurones

Dans beaucoup de domaines, l'usage de réseaux de neurones a permis l'amélioration significative des performances, comme dans la synthèse de voix à partir de textes [WHM+16]. D'après cette source principale http://www.deeplearningbook.org/, Les réseaux de neurones convolutionnels [OBLS14] et récurrents, LSTM (Long Short-Term Memory) [HS97] sont expliqués.



$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{8}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{9}$$

$$\widetilde{c_t} = th(W_c[h_{t-1}, x_t] + b_c) \tag{10}$$

$$c_t = f_t * c_{t-1} + i_t * \widetilde{c_t} \tag{11}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{12}$$

$$h_t = o_t * th(c_t) \tag{13}$$

Les réseaux de neurones convolutifs sont un type de réseaux de neurones qui comportent au moins une couche qui effectue des convolutions avec des données provenant de l'entrée ou d'une autre couche.

L'intérêt de ce type de réseaux de neurones, c'est qu'on prend moins en compte les interactions entre les données en entrées, en effet, suivant la taille du filtre, pour le filtrage d'un point , on ne prendra en compte qu'un voisinage réduit pour le calcul alors que pour les réseaux de neurones classiques, pour la première couche, on prend très souvent toutes les données en entrées pour calculer toutes les données en sortie.

Un autre fait intéressant pour les calculs, c'est que les poids sont partagés entre les calculs, ou au moins sont liés, alors que pour les réseaux de neurones traditionnels, chaque composante des données en entrée d'une couche a un vecteur poids associé à lui-même. Cette propriété rend toutes translations de données invariant aux translations.

D'abord, des convolutions sont faites en parallèle sur les données en entrée, ensuite, toutes ces données passent à travers des fonctions d'activation, enfin toutes les données sont regroupées (*pooling*) et résumées à travers une opération telle le *max pooling* qui consiste à garder la valeur maximale à l'intérieur d'une fenêtre des données et de la renvoyer. Il y a aussi la possibilité de prendre la moyenne dans le voisinage, ou alors de prendre la norme L2, ou une moyenne pondérée par a distance d'éloignement du point central. Ce dernier point aide à être invariant par translation, mais aussi à accepter des données en entrée de tailles différentes.

3.3 Chaîne de Markov cachée

Y = vecteurs d'attributs X = classe de vitesse Algorithme forward-backward

Une fois le modèle appris, il nous reste l'algorithme pour segmenter les images en différentes catégories. L'algorithme forwards-backwards sert à calculer la probabilité a posteriori, i.e. $p(x_i|y)$. Pour une image donnée, on calcule cette probabilité pour toutes les classes possibles et on choisira la classe qui maximise cette probabilité a posteriori.

L'algorithme qui permet de calculer la probabilité a posteriori de chaque image d'une séquence est l'algorithme forwards-backwards. Redémontrons l'algorithme sur une séquence annotée de taille N. Passons d'abord par la loi jointe de la chaîne de Markov cachée. Soit

$$p(X_{1:N}, Y_{1:N}) = p(X_{1:N}) * p(Y_{1:N}|X_{1:N}) = p(X_1) * \prod_{i=1}^{N-1} p(X_{i+1}|X_i) \prod_{i=1}^{N} p(Y_i|X_i). \tag{14}$$

Pour la formule précédente et pour les prochaines, on a ces propriétés : la formule de Bayes, $(X_i)_{i\in 1:N}$ qui est une chaîne de Markov, les X_i sont indépendantes conditionnellement à $X_{1:N}$ et $Y_i|X_{1:N} = Y_i|X_i$ Ici, on ne souhaite calculer que les $p(X_i|Y_{i:N})$. En posant

$$\begin{aligned} p(x_i|y_{1:N}) &= \frac{p(x_i, y_{1:N})}{p(y_{1:N})} \\ &= \frac{p(x_i, y_{1:i}) * p(y_{i+1:N}|x_i, y_{1:N})}{p(y_{1:N})} \\ &= \frac{p(x_i|y_{1:i}) * p(y_{1:i}) * p(y_{i+1:N}|x_i)}{p(y_{1:N})} \\ &= p(x_i|y_{1:i}) * p(y_{i+1:N}|x_i) * \frac{p(y_{1:i})}{p(y_{1:N})} \end{aligned} \tag{15}$$

On pose

$$\alpha_i(x_i) = p(x_i|y_{1:i}) \tag{16}$$

et

$$\beta_i(x_i) = p(y_{i+1:N}|x_i). \tag{17}$$

Posons ce qu'on veut obtenir. Veut-on obtenir la plus probable des séquences alors la séquence des états les plus probables ? La première séquence peut être approchée par l'algorithme de Viterbi et la deuxième est obtenue plus facilement. Pour la suite, on choisira la deuxième méthode.

$$\underset{x_i}{\operatorname{argmax}}\ p(x_i|y_{1:N}) = \underset{x_i}{\operatorname{argmax}}\ \alpha_i(x_i) * \beta_i(x_i) \tag{18}$$

On peut écrire $\alpha_{i+1}(x_{i+1})$ en fonction de $\alpha_i(x_i)$.

$$\begin{aligned} \alpha_{i+1}(x_{i+1}) &= p(x_{i+1}|y_{1:i+1}) \\ &= \frac{p(x_{i+1}, y_{1:i+1})}{p(y_{1:i+1})} \\ &= \frac{\sum_{x_i} p(x_i, x_{i+1}, y_{1:i+1})}{p(y_{1:i+1})} \\ &= \frac{\sum_{x_i} p(y_{i+1}|x_{i+1}, x_i, y_{1:i}) * p(x_{i+1}|x_i, y_{1:i}) * p(x_i, y_{1:i})}{p(y_{1:i+1})} \\ &= \sum_{x_i} p(y_{i+1}|x_{i+1}) * p(x_{i+1}|x_i) * \alpha_i(x_i) * \frac{p(y_{1:i})}{p(y_{1:i+1})} \end{aligned} \tag{19}$$

On peut donc calculer les $\alpha_i(x_i)$ à partir de $\alpha_1(x_1)$ jusqu'à $\alpha_N(x_N)$ avec $\alpha_1(x_1 = k)$ valant $\frac{p(x_1=k)*p(y_1|x_1=k)}{\sum_j p(x_1=j)*p(y_1|x_1=j)}$
On peut écrire $\beta_{i-1}(x_{i-1})$ en fonction de $\beta_i(x_i)$.

$$\begin{aligned}
\beta_{i-1}(x_{i-1}) &= p(y_{i:N}|x_{i-1}) \\
&= \frac{p(y_{i:N}, x_{i-1})}{p(x_{i-1})} \\
&= \frac{\sum_{x_i} p(y_{i:N}, x_{i-1}, x_i)}{p(x_{i-1})} \\
&= \frac{\sum_{x_i} p(y_i, x_{i-1}|y_{i+1:N}, x_i) * p(y_{i+1:N}, x_i)}{p(x_{i-1})} \\
&= \frac{\sum_{x_i} p(y_i|y_{i+1:N}, x_i) * p(x_{i-1}|y_i, y_{i+1:N}, x_i) * p(y_{i+1:N}|x_i) * p(x_i)}{p(x_{i-1})} \\
&= \sum_{x_i} p(y_i|x_i) * p(x_{i-1}|x_i) * \beta_i(x_i) * \frac{p(x_i)}{p(x_{i-1})} \\
&= \sum_{x_i} p(y_i|x_i) * p(x_i|x_{i-1}) * \beta_i(x_i)
\end{aligned} \tag{20}$$

On peut donc calculer les $\beta_i(x_i)$ à partir des $\beta_N(x_N)$ et récursivement jusqu'à $\beta_1(x_1)$ avec $\beta_N(x_N)$ valant 1 pour tous les états possibles. Les $\alpha_i(x_i)$ et $\beta_i(x_i)$ étant très petits, il faudra les normaliser : $\alpha_i(x_i = k) = \frac{\alpha_i(x_i=k)}{\sum_j \alpha_i(x_i=j)}$ et $\beta_i(x_i = k) = \frac{\beta_i(x_i=k)}{\sum_j \beta_i(x_k=j)}$

3.4 Chaînes de Markov couple

Y = vecteurs d'attributs X = classe de vitesse Algorithme forward-backward

3.5 Chaînes de Markov triplet

3.6 Champs de Markov cachée—conditionnels

Les champs de Markov sont beaucoup utilisés pour segmenter des images

Les champs de Markov sont beaucoup utilisés pour segmenter des images en différentes régions.

Les articles de base sur le sujet sont [GG84] et [Mar85]. Le premier a introduit ce qu'on appelle l'échantillonneur de Gibbs et le second

Soient $X = (X_s)_{s\in S}$ et $Y = (Y_s)_{s\in S}$ deux champs aléatoires. X_s prend ses valeurs dans un ensemble discret de classes (l'ensemble des vitesses maximales autorisées). Y_s est une variable aléatoire réelle (il faudra que ça soit plutôt de dimension 3 pour notre cas). Dans le cas d'un champ de Markov X, $P(X_s|X_t, t \in S, t \neq s) = P(X_s|X_t, t \in V_s)$ avec V_s le voisinage du pixel s. Il faut noter que les s dans S sont localisables (dans notre cas dans un espace en deux dimensions). Un voisinage est un ensemble dépendant d'une forme géométrique et qui inclut des pixels . Une clique est tout sous-ensemble de S tel qu'il soit un singleton ou tel que tous ses éléments sont mutuellement voisins. Soient s et t dans S, s et t appartiennent à un même voisinage équivaut à dire qu'ils sont voisins.

Le **théorème de Hammersley-Clifford** : Le champ X = $(X_s)_{s\in S}$ est markovien par rapport à un système de voisinages V = $(V_s)_{s\in S}$ avec la probabilité de toutes les réalisations non nulle, si et seulement si la fonction énergie de sa distribution $P_X(x) = \gamma\ exp(-U(x))$ est donnée par $U(x) = \sum_{c\in C} \phi_c(x_c)$, avec C l'ensemble des cliques associé au système de voisinages B. NP = nombre de valeurs possibles par pixels.

On peut donc faire quelques remarques : tout d'abord, si le système de voisinage (la manière de construire les relations de voisinage à un point) est simple ou plutôt ne contient que peu de voisins, alors la complexité du calcul (la somme sur toutes les cliques possibles) est vraiment plus faible (on passe de $O((NP)^{largeur*hauteur})$ dans le cas où tout dépend de tout à O($|C_s|$).

On pose $s \in S$ et $X^* = (X_t)_{t\in S, t\neq s}$. Calculons la probabilité qu'un pixel s appartiennent à la classe i sachant comment tous les pixels ont été classés :

$$
\begin{aligned}
P(X_s = i | X^* = x^*) &= \frac{P(X_s = i, X^* = x^*)}{\sum_{j=1}^{K} P(X_s = j, X^* = x^*)} \\
&= \frac{\gamma\ exp(-U(i, x^*))}{\sum_{j=1}^{K} \gamma\ exp(-U(j, x^*))} \\
&= \frac{exp(-\sum_{c \in D} \phi_c(i, x^*))}{\sum_{j=1}^{K} exp(-\sum_{c \in D} \phi_c(j, x^*))} \\
&= \frac{exp(-\sum_{c \in D_s} \phi_c(i, x^*)) * exp(-\sum_{c \in \bar{D}_s} \phi_c(i, x^*))}{\sum_{j=1}^{K} exp(-\sum_{c \in D_s} \phi_c(j, x^*)) * exp(-\sum_{c \in \bar{D}_s} \phi_c(j, x^*))} \\
&= \frac{exp(-\sum_{c \in \bar{D}_s} \phi_c(x^*)) * exp(-\sum_{c \in D_s} \phi_c(i, x^*))}{exp(-\sum_{c \in \bar{D}_s} \phi_c(x^*)) * \sum_{j=1}^{K} exp(-\sum_{c \in D_s} \phi_c(j, x^*))} \\
&= \frac{exp(-\sum_{c \in D_s} \phi_c(i, x^*))}{\sum_{j=1}^{K} exp(-\sum_{c \in D_s} \phi_c(j, x^*))}
\end{aligned}
\tag{21}
$$

Dans le cadre des champs de Markov, il est courant d'utiliser l'échantillonneur de Gibbs. Afin de générer une image en fonction d'une fonction énergie et des paramètres, on peut l'utiliser. Tour d'abord, on initialise l'image : $(x_s^0) = x_s^0$, ensuite, on tire x_s selon le voisinage grâce à la loi de probabilité donnée par le théorème de Hammersley-Clifford et on attribue la classe tirée on fait ça pour tous les pixels et un certain nombre de fois (15 à 30 ?!?).

Dans le cas des champs de Markov cachée, on pose très couramment (il faut expliquer ça) $p(y|x) = \prod_{s \in S} p(y_s|x_s)$ (les hypothèses sont que les observations Y sont indépendantes conditionnellement à X et $p(y_s|x) = p(y_s|x_s)$. $p(x, y) = \gamma\ exp(-\sum_{c \in C} \varphi_c(x_c)) \prod_{s \in S} p(y_s|x_s) = \gamma\ exp(-\sum_{c \in C} \varphi_c(x_c) + \underset{s \in S}{sum} log(p(y_s|x_s))$. D'où $p(x|y) = \frac{\gamma}{p(y)} exp(-\sum_{c \in C} \varphi_c(x_c) + \sum_{s \in S} log(p(y_s|x_s))$ qui est un champ de Markov.

3.7 Segmentation statistique floue

D'abord, classer les images de routes dans les catégories suivantes : {30, 50}, {50, 70}, {70, 90}, {90, 110}, {110, 130}. Puis affiner le classement en précisant la classe finale (classement binaire).

Dans [TB11], les classements se font sur différentes sortes de chevauchement de classes de routes. Les meilleures performances sont trouvées sur une classification à deux classes.

3.8 SVM

3.9 Adaboost

3.10 Réduction de dimension

3.10.1 Analyse en composante principale

3.10.2 ISOMAP

3.11 Modèle à maximum d'entropie

Déjà utilisé en TAL [Rat97] dans le domaine de la classification de textes ou encore dans l'étiquetage morpho-syntaxique [R+96], le modèle à maximum d'entropie part du principe général que lorsqu'on doit estimer la loi de probabilité ayant généré nos données d'apprentissage, il faut choisir la loi qui a la plus grande entropie parmi toutes

celles possibles. En effet Pour ce qui nous concerne, il existe déjà [GLS07]. Dedans, l'auteur utilise l'approche pas sac-de-mots puis propose une probabilité a posteriori pour classer les images en estimant les paramètres à l'aide du modèle à maximum d'entropie.

References

[BBLP10] Y-Lan Boureau, Francis R. Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 2559–2566. IEEE Computer Society, 2010.

[BMM07] Anna Bosch, Xavier Muñoz, and Robert Martí. Which is the best way to organize/classify images by content? *Image and vision computing*, 25(6):778–791, 2007.

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[CDF+04] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*, pages 886–893. IEEE Computer Society, 2005.

[FMC15] Philippe Foucher, Emmanuel Moebel, and Pierre Charbonnier. Route segmentation into speed limit categories by using image analysis. In José Braz, Sebastiano Battiato, and Francisco H. Imai, editors, *VISAPP 2015 - Proceedings of the 10th International Conference on Computer Vision Theory and Applications, Volume 2, Berlin, Germany, 11-14 March, 2015.*, pages 416–423. SciTePress, 2015.

[FS15] Mehdi Faraji and Jamshid Shanbehzadeh. Bag-of-visual-words, its detectors and descriptors; a survey in detail. *Advances in Computer Science: an International Journal*, 4(2):8–20, 2015.

[GG84] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

[GLS07] Sheng Gao, Joo-Hwee Lim, and Qibin Sun. Hidden maximum entropy approach for visual concept modeling. In *ICME*, pages 1387–1390. Citeseer, 2007.

[Har79] Robert M Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804, 1979.

[HGBRM10] Nguyen Vu Hoàng, Valérie Gouet-Brunet, Marta Rukoz, and Maude Manouvrier. Embedding spatial information into image content description for scene retrieval. *Pattern Recognition*, 43(9):3013–3024, 2010.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[JPD+12] Herve Jegou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Perez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *IEEE transactions on pattern analysis and machine intelligence*, 34(9):1704–1716, 2012.

[KB01] Timor Kadir and Michael Brady. Saliency, scale and image description. *International Journal of Computer Vision*, 45(2):83–105, 2001.

[KBMD15] Rahat Khan, Cécile Barat, Damien Muselet, and Christophe Ducottet. Spatial histograms of soft pairwise similar patches to improve the bag-of-visual-words model. *Computer Vision and Image Understanding*, 132:102–112, 2015.

[Low99] David G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, 1999.

[Mar85] Jose Luis Marroquin. Probabilistic solution of inverse problems. 1985.

[MBM+13] Luc Mioulet, Toby P Breckon, Andre Mouton, Haichao Liang, and Tak@inproceedingsmioulet2013gabor, title=Gabor features for real-time road environment classification, author=Mioulet, Luc and Breckon, Toby P and Mouton, Andre and Liang, Haichao and Morie, Takashi, booktitle=Industrial Technology (ICIT), 2013 IEEE International Conference on, pages=1117–1121, year=2013, organization=IEEE ashi Morie. Gabor features for real-time road environment classification. In *Industrial Technology (ICIT), 2013 IEEE International Conference on*, pages 1117–1121. IEEE, 2013.

[OBLS14] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[OT01] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

[PSV+14] Otávio Augusto Bizetto Penatti, Fernanda B. Silva, Eduardo Valle, Valérie Gouet-Brunet, and Ricardo da Silva Torres. Visual word spatial arrangement for image retrieval and classification. *Pattern Recognition*, 47(2):705–720, 2014.

[Que07] Pedro Quelhas. Scene image classification and segmentation with quantized local descriptors and latent aspect modeling. 2007.

[R+96] Adwait Ratnaparkhi et al. A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing*, volume 1, pages 133–142. Philadelphia, USA, 1996.

[Rat97] Adwait Ratnaparkhi. A simple introduction to maximum entropy models for natural language processing. *IRCS Technical Reports Series*, page 81, 1997.

[SPMV13] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *International journal of computer vision*, 105(3):222–245, 2013.

[SZ03] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *9th IEEE International Conference on Computer Vision (ICCV 2003), 14-17 October 2003, Nice, France*, pages 1470–1477. IEEE Computer Society, 2003.

[TB11] Isabelle Tang and Toby P Breckon. Automatic road environment classification. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):476–484, 2011.

[THC+16] Dang Duy Thang, Shintami C Hidayati, Yung-Yao Chen, Wen-Huang Cheng, Shih-Wei Sun, and Kai-Lung Hua. A spatial-pyramid scene categorization algorithm based on locality-aware sparse coding. In *Multimedia Big Data (BigMM), 2016 IEEE Second International Conference on*, pages 342–345. IEEE, 2016.

[VGVSG10] Jan C Van Gemert, Cor J Veenman, Arnold WM Smeulders, and Jan-Mark Geusebroek. Visual word ambiguity. *IEEE transactions on pattern analysis and machine intelligence*, 32(7):1271–1283, 2010.

[WB07] Simon AJ Winder and Matthew Brown. Learning local image descriptors. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[WHM+16] Oliver Watts, Gustav Eje Henter, Thomas Merritt, Zhizheng Wu, and Simon King. From hmms to dnns: where do the improvements come from? In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5505–5509. IEEE, 2016.

[WR11] Jianxin Wu and Jim M Rehg. Centrist: A visual descriptor for scene categorization. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1489–1501, 2011.

[WYY+10] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3360–3367. IEEE, 2010.

[XWY14] Yang Xiao, Jianxin Wu, and Junsong Yuan. mcentrist: A multi-channel feature generation mechanism for scene categorization. *IEEE Trans. Image Processing*, 23(2):823–836, 2014.

[YWY07] Junsong Yuan, Ying Wu, and Ming Yang. Discovery of collocation patterns: from visual words to visual phrases. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[YYGH09] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801. IEEE, 2009.