Implémentation d'une cascade de régresseurs pour l'alignement de points caractéristiques du visage

Arnaud Dapogny, Kevin Bailly

L'objectif de ce TP est de réaliser une méthode d'alignement de points caractéristiques du visage inspirée de [1]. A l'issue du TP, vous devez fournir :

- Un court rapport présentant votre méthode, les résultats, les réponses aux questions posées dans cet énoncé ainsi qu'une analyse critique.
- Les sources du projet (en langage python)

Quelques soit le moyen utilisé pour nous transmettre votre TP (DropSU, github, gitlab...), vous retirerez les données d'entrainement et de test que nous vous fournissons. Par ailleurs vous enverrez par mail (kevin.bailly@sorbonne-universite.fr)le lien pour y accéder avant la deadline fixée par l'encadrant et avec l'objet suivant **[IMA] TP analyse faciale**.

1 Préparation des données

Dans un premier temps, vous allez devoir préparer les données qui permettront d'apprendre et d'évaluer votre méthode d'alignement.

1.1 Téléchargement des données

Les données sont disponibles à partir de [ce lien] (1Go). En attendant la fin du téléchargement, vous pouvez travailler sur un échantillon de la base de données disponible via [ce lien] (l'arborescence de cet échantillon est la même que celle de la base principale)

Cette base est la réunion de 4 sous ensembles de données (Helen, AFW, LFPW et IBug) dont les images (extension .jpg ou .png) et les annotations (extension .pts) sont contenues dans les repertoires du même nom.

Vous trouverez également des fichiers .txt contenant la liste des données à utiliser pour l'apprentissage –`300w_train_images.txt` et `300w_train_landmarks.txt` respectivement pour la liste des images et des annotations– ainsi que pour les tests.

1.2 Visualisation des données

Pour prendre en main les données, la première étape consiste à *parser* le fichier `300w_train_images.txt` et à afficher aléatoirement une dizaine d'images avec les points caractéristiques correspondants.

1.3 Augmentation des données

Pour chaque image de l'ensemble d'apprentissage, vous devez :

1. Calculer les paramètres de la boite englobante des points caractéristiques du visage

2. Elargir cette boîte englobante de 30%, découper l'image avec ces nouvelles dimensions, redimensionner l'image en $128 * 128$ et enregistrer l'image.

3. Calculer les coordonnées des points caractéristiques pour cette nouvelle image. La position des points constituera la vérité terrain (vous afficherez l'image et les points caractéristiques pour vérifier que ce prétraitement est effectif)

4. Calculer la position moyenne des points caractéristique sur l'ensemble des images de la base d'apprentissage

5. Générer 10 perturbations aléatoires de la position des points caractéristiques (en translation et en facteur d'échelle) et enregistrer chacune de ces réalisations. L'amplitude de ces déformations sera de $\pm 20\%$ pour le facteur d'échelle et $\pm 20px$ pour les translations (indépendamment en x et y). Pourquoi applique-t-on ces transformations ? Proposer une méthode automatique pour déterminer l'amplitudes de ces déplacements ?

A la fin de cette étape vous disposez des données d'apprentissage dont vous aurez besoin pour apprendre votre modèle. Il faut appliquer les mêmes transformations aux images de test (sans les perturbations aléatoires) afin d'être dans les mêmes conditions pour l'évaluation.

2 Apprentissage d'un régresseur simple

Dans cette partie, nous souhaitons estimer un **déplacement** des points caractéristiques à partir d'un modèle de points moyen, *i.e.* apprendre en utilisant des outils élémentaires de l'apprentissage statistique et de la vision par ordinateur, une fonction $f : I, s_0 \rightarrow \delta_s$ telle que $s_0 + \delta_s \approx s^*$ au sens des moindres carrés.

2.1 Extraction de caractéristiques image

Pour ce faire, une première étape consiste à **abstraire** localement (*i.e.* au voisinage de chaque point caractéristique) la représentation image.

1. Pourquoi ne peut-on pas utiliser directement les valeurs des pixels de l'image comme représentation?

2. Créer, pour chaque point caractéristique courant du modèle moyen $s^i_{0 i=0,\ldots,68}$, un objet cv2.keyPoint en spécifiant ses coordonnées, et une taille de fenêtre (paramètre size, réglant le diamètre du voisinage) de 20 pixels.

3. Utiliser la fonction *sift.compute* de opencv pour calculer un descripteur SIFT au voisinage de chaque point caractéristique. Quel est la dimensionnalité de chacun de ces descripteurs?

4. Pour chaque image, concaténer l'ensemble des descripteurs obtenus pour tous les points caractéristiques. Quelle est la dimension de ce descripteur?

5. Citer deux exemples de représentation image qui pourraient être utilisés à la place des descripteurs SIFT, en expliquant brièvement le principe sous-jacent.

2.2 Réduction de dimensionalité

A ce stade, nous obtenons une matrice $\mathbf{X}_0 \in \mathbb{R}^{N \times M}$ dont les lignes contiennent les différentes images de la base de donnée, et les colonnes les dimensions des descripteurs obtenus (la concaténation des SIFTs obtenus au voisinage de chaque point caractéristique de la forme moyenne s_0^i). Une deuxième étape consiste à réduire la dimension des descripteurs obtenus.

Rappel: en calculant l'ACP, nous obtenons une matrice de projection $\mathbf{A}_0 \in \mathbb{R}^{M \times M'}$, avec $M' < M$, et telle que la variance totale (trace de la matrice de covariance) des données projetées $\tilde{\mathbf{X}}_0 = \mathbf{A}_0\mathbf{X}_0$, soit supérieure à 98% de la variance totale des données originales $\mathbf{X}_0$. En d'autres termes, nous conservons plus de 98% de la variabilité totale des données de départ.

1. Quel est l'intérêt principal de la réduction de dimensionalité en machine learning? Quelles sont les principales méthodes de réduction de dimensionalité?
2. Utiliser l'analyse en composante principale (ACP ou PCA en anglais - librairies opencv ou skimage, au choix) en conservant 98% de la variance totale des descripteurs.
3. Quelles sont les dimensions de la matrice $\tilde{\mathbf{X}}_0$ obtenue par réduction de dimensionalité?

2.3 Estimation du déplacement

Maintenant que nous disposons de représentations images locales compressées pour chaque image de la base de donnée $\tilde{\mathbf{X}} = \mathbf{A}_0\mathbf{X}_0$, nous souhaitons estimer une régression linéaire permettant de prédire un déplacement de l'ensemble des points caractéristiques δ_s à partir de ces représentation. Formellement, ce déplacement vérifie

$$\delta_s^0 = argmin_{\delta_s} ||\delta_s^* - \delta_s||^2 \tag{1}$$

Avec $\delta_s^* \in \mathbb{R}^{136}$ le déplacement optimal (différence, pour chaque coordonnée de chacun des 68 points caractéristiques, entre la vérité terrain et s_0). En posant $\delta_s = R_0\tilde{\mathbf{X}}_0 + b_0$, nous nous ramenons donc à chercher:

$$argmin_{R_0,b_0} ||\delta_s^* - R_0\tilde{\mathbf{X}}_0 - b_0||^2 \tag{2}$$

Ou encore, en posant $\tilde{\mathbf{Y}}_0 \in \mathbb{R}^{D \times M'+1}$ la matrice $\tilde{\mathbf{X}}$ avec une colonne additionnelle de 1,

$$argmin_{R_0} ||\delta_s^* - R_0\tilde{\mathbf{Y}}_0||^2 \tag{3}$$

Alors $R_0 = (\tilde{\mathbf{Y}}_0^T\tilde{\mathbf{Y}}_0)^{-1}\tilde{\mathbf{Y}}_0^T\delta_s^* \in \mathbb{R}^{M+1 \times 136}$ est la pseudo-inverse de Moore-Penrose, c'est-à-dire le régresseur linéaire (dont la dernière ligne est un biais correspondant à chaque coordonnée du déplacement) minimisant l'erreur au sens des moindres carrés.

1. Calculer la matrice $\tilde{\mathbf{Y}}_0$, la matrice de déplacements δ_s^*.
2. Calculer R_0 en appliquant une résolution aux moindres carrés et le déplacement $R_0\tilde{\mathbf{Y}}_0$ pour chaque image (ou chaque ligne).
3. Calculer l'erreur $\delta_s^* - R_0\tilde{\mathbf{Y}}_0$ et afficher, pour la première image de la base, les points caractéristiques initiaux s_0 (en vert) ainsi que les points déplacées $s_0 + \delta_s$. Que peut-on en conclure?

2.4 Validation sur un ensemble de test externe

A partir d'une position de départ s_0, les matrices A_0 et R_0 apprises respectivement par ACP et régression aux moindres carrés définissent le modèle d'alignement de points caractéristiques. Etant donné une nouvelle image, l'alignement est effectué en extrayant les descripteurs SIFTs autour de chaque point (en s_0), en compressant cette représentation (multiplication matricielle par A_0), en ajoutant le terme de biais permettant d'obtenir $\tilde{\mathbf{Y}}_0$, puis en appliquant une multiplication matricielle par R_0. Il conviendra donc d'enregistrer les valeurs contenues dans s_0 (forme moyenne), A_0 (matrice de projection de l'ACP), et R_0 (coefficients des régressions linéaires).

1. En quoi l'évaluation précédente n'est-elle pas pertinente pour évaluer la capacité de généralisation de la procédure d'alignement de points caractéristiques?
2. Calculer l'erreur $\delta_s^* - R_0\tilde{\mathbf{Y}}_0$ sur le set de test et afficher, pour la première image de la base, les points caractéristiques initiaux s_0 (en vert) ainsi que les points déplacées $s_0 + \delta_s$. Que peut-on en conclure?

3 Bonus : Cascade de régresseurs

Nous avons, dans la partie précédente, extrait des caractéristiques locales robustes (SIFT) à partir de l'image, réduit la dimension de ces descripteurs et appris un modèle de régression linéaire permettant d'estimer un déplacement $\delta_s^0 = R_0\tilde{\mathbf{Y}}_0$. Toutefois, l'erreur résiduelle d'apprentissage $\delta_s^* - R_0\tilde{\mathbf{Y}}_0$ est toujours significative. Afin de réduire davantage cette erreur, une solution consiste à itérer le processus mis en place à la partie 2, c'est-à-dire d'appliquer itérativement toutes les étapes vues plus haut à partir du déplacement du modèle $s_1 = s_0 + \delta_s^0$ afin d'apprendre un déplacement δ_s^1, puis δ_s^2 à partir de $s_2 = s_1 + \delta_s^1$, et ainsi de suite.

3.1 Itération du processus d'alignement

1. Ecrire une fonction `majPoints` qui, à l'itération $k+1$, calcule la position des nouveaux points caractéristiques étant donné une position initiale s_k et un déplacement δ_s^k
2. Ecrire une fonction `oneStepAlignment` qui calcule les étapes **2.1**, **2.2**, **2.3** ainsi que la question 1 du **3.1** précédente.
3. Appliquer itérativement $k = 5$ fois la Fonction `oneStepAlignment` à partir de s_0. Le vecteur s_0, ainsi que les matrices A_k et R_k pour $k = 0 \dots 4$ seront sauvegardées.
4. Evaluer les 5 modèles $k = 0 \dots 4$ et afficher, pour la première image de la base, les points caractéristiques initiaux s_k (en vert) ainsi que les points déplacées $s_k + \delta_s^k$. Que peut-on en conclure?
5. Evaluer les 5 modèles $k = 0 \dots 4$ sur le set de test et afficher, pour la première image de la base, les points caractéristiques initiaux s_k (en vert) ainsi que les points déplacées $s_k + \delta_s^k$. Que peut-on en conclure?

3.2 Detection des points caractéristiques sur d'autres images

On veut maintenant évaluer le modèle d'alignement de points caractéristiques obtenu en partie **3** des images réelles. Pour cela, vous devrez :

1. Collecter des données de visages : images de votre visage capturées à l'aide d'une webcam, ou images récupérées sur Internet. Vous choisirez des images présentant des cas simples (visage de face, sans expression et avec une bonne illumination) et plus complexe (variation de pose et d'expression).

2. Détecter de visage à l'aide de la méthode Viola Jones d'OpenCV (`detectMultiScale`).

3. Appliquer votre modèle d'alignement de points caractéristiques et observer qualitativement les résultats obtenus.

References

[1] Xuehan Xiong and Fernando De la Torre. "Supervised Descent Method and Its Applications to Face Alignment". In: CVPR '13. 2013, pp. 532–539.