Projet – Localisation Automatique par Carte Visuelle

Introduction

Définissons une « carte visuelle » comme une base de données $\bar{I} = \{(I_i, X_i)\}$ comprenant une séquence d'images I_i (ex. un vidéo) prises aux déplacements spatiaux réguliers connues X_i. Le but de ce projet s'agit d'estimer l'endroit X ou une nouvelle image I a été prise, vis à vis a une carte visuelle $\bar{I} = \{(I_i, X_i)\}$. Un tel système pourrait avoir plusieurs utilités, ex. de localiser un véhicule autonome.

Méthode Suggéré

Une approche probabiliste s'agit de:

1) Générer une distribution de probabilité a posteriori $p(X|I)$ sur l'endroit X étant donné la nouvelle image I.
2) Trouver la position maximum a posteriori $X^{MAP} = \max_X p(X|I)$ parmi les positions dans la carte visuelle.

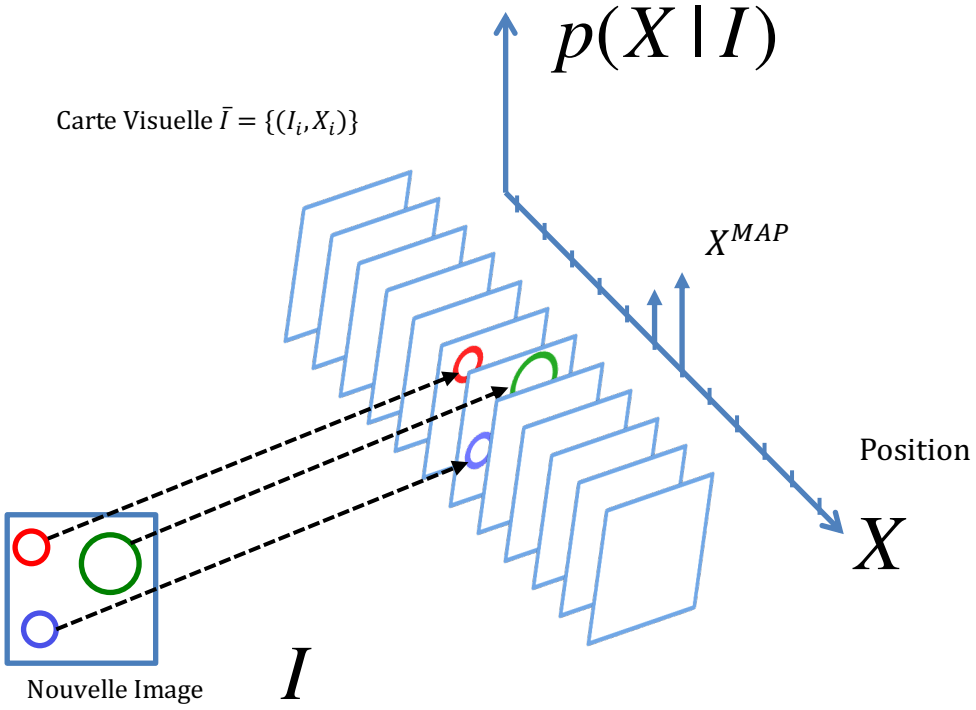


Figure 1: Localisation par carte visuelle, caractéristiques localles (cercles)

Algorithme :

1) Extraire des caractéristiques dans chaque image (Figure 1).
2) Calculer les proches voisins entre les caractéristiques d'image de la nouvelle image I et celles des images $\bar{I} = \{I_i\}$ de la carte visuelle (mots clés : « feature matching », « nearest-neighbor correspondence »)

3) Estimer une distribution de probabilité $p(X|I)$ de la position X étant donné la nouvelle image I, en accumulant un histogramme de correspondances à travers la carte visuelle. Trouvez le maximum X^{MAP}.

Expériences et Résultats

Évaluation quantitative : performance du système en termes d'erreur, temps de calcul, mémoire nécessaire, etc. Estimer l'erreur, ex. en calculant l'écart moyenne entre les positions estimées X^{MAP} et réelles X à travers toutes les images d'essai.

Évaluation qualitative : décrire le fonctionnement du système. Est-ce qu'il y a des images qui sont plus difficiles à localiser que d'autres ? Lesquelles ? Pourquoi ? Dans quels contextes le système serait-t-il le plus utile ? Le moins utile ?
Quels facteurs déterminent la quantité de mémoire, le temps de calcul ? Est-ce qu'il y a des façons d'améliorer l'erreur, temps de calcul, mémoire ?

Estimation d'erreur : Pour chaque scène, la caméra suit une séquence de 10 déplacements X=1…10, en prenant des photos, ex. deux séquences A1… A10, B1…B10. La prédiction pour chaque image est simplement le déplacement selon une séquence d'images indépendant, et l'erreur s'exprime en écart de déplacement.

Par exemple, vous avez deux séquences d'images de la même scène A=(A1...A10) et B=(B1...B10). On doit prédire les positions de A selon B, et aussi de B selon A, en évaluant de façon quantitative l'erreur des prédictions. Pour une image donnée A_i, l'erreur de prédiction B_j est l'écart de l'index selon la séquence

$$Erreur(A_i, B_j) = |i - j|$$

Par exemple, si la prédiction pour l'image A5 est le B6, l'erreur est le déplacement et |5-6| = 1. Pour chaque scène donnée, on devrait estimer la somme des erreurs pour toutes les images A selon B, et B selon A :

$$Erreur(A, B) = \sum_{i=1}^{10} Erreur(A_i, B_j) + \sum_{i=1}^{10} Erreur(B_i, A_j)$$

Revue de la littérature

Quelles sont les approches ou méthodes possibles pour la tâche de localisation ? Quelles sont les technologies clés et comment est-ce qu'elles fonctionnent ? Fournir des descriptions et des références bibliographiques, ex. Google Scholar.

Données

Les données vidéos sont fournies pour 3 scènes (Légumes, Neige, Magasin) pour développer et tester votre algorithme. 3 scènes supplémentaires seront fournies vers la fin de la session pour évaluation. Vous êtes encouragés de capter vos propres vidéos des scènes d'intérêts pour tester votre algorithme.

Chaque scène comporte deux vidéos (ex. séquence A, séquence B) prises sur la même trajectoire, une qui sert comme de carte visuelle (donnée d'apprentissage), une qui sert à évaluer l'algorithme (donnée d'essai). Tandis que l'algorithme nécessite deux séquences au minimum, la performance de l'algorithme va augmenter avec plus de séquences.

Développement de logiciel

Correspondance de caractéristiques locaux (SIFT, SURF, AKAZE, …) ou de vecteurs de CNN pré-entrainé extraites de chaque image, langages Python, C++ ou Matlab. Fournissez

- votre code source
- les instructions pour le compiler et le lancer
- quelques données d'essaye pour valider le fonctionnement du système

Interprétation du CNN par Points Clefs CNN:

Si vous abordez le projet par réseau CNN, veuillez explorer le fonctionnement du réseau à partir des points clefs. Utiliser un filtre Laplacien pour extraire des points informatifs dans une couche (ex. la première couche) de convolution. Ensuite trouver et visualiser des points similaires dans des images différentes par correspondances proche-voisin. Voici les étapes.

A. Extraction de Points Clés-CNN

Entrée : image I
Sortie : ensemble { (xi,yi,di) } de points clés (xi,yi) avec descripteurs di

1) Obtenir une couche d'activation CNN
    a. ex. VGG16 couche 1 : 224x224 pixels, N=64 canaux
2) Convoluer chaque canal de la couche CNN avec un filtre laplacien 3x3
    a. Entrée : couche CNN, ex. 224x224 pixels, N=64 canaux
    b. Sortie : couche laplacien, ex. 224x224 pixels, N=64 canaux
3) Calculer la magnitude euclidienne du vecteur laplacien
    a. Entrée : couche laplacien, ex. 224x224 pixels, N=64 canaux
    b. Sortie : image de magnitude, ex. 224x224 pixels, N=1 canal
4) Identifier des maxima locaux
    a. Entrée : image de magnitude laplacien, ex. 224x224 pixels, N=1 canal
    b. Sortie : ensemble { (xi,yi,di) } de
        i. point clés (xi,yi) maximum de laplacien
        ii. descripteur di prise de la couche CNN au coordonnée (xi,yi)

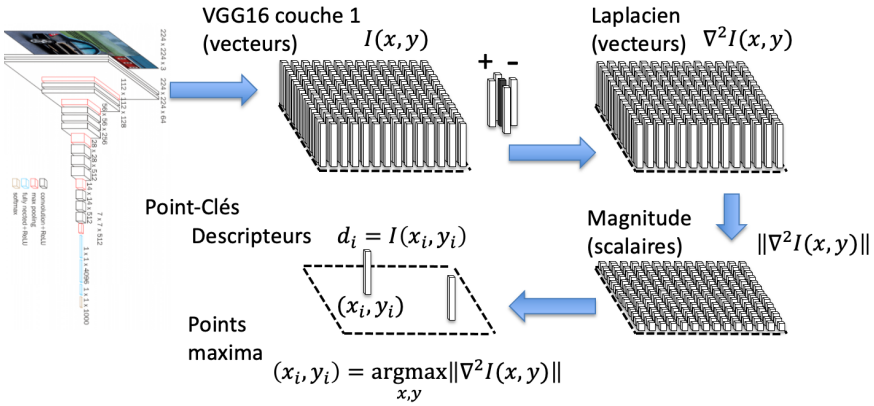
B. Correspondance de Points Clés-CNN

Entrée : deux images Ii, Ij
Sortie : ensemble de correspondance { (xi,yi)-> (xj,yj) }

1) Extraire des points clés de chaque image Ii->{ (xi,yi) } et Ij->{ (xj,yj) }
2) Trouver des proches voisins, distance euclidienne minimum

a. (xi,yi)-> (xj,yj)= argmin(|| di-dj ||)

A. Point-Clés CNN Laplacien : Extraction



B. Point-Clés CNN Laplacien : Correspondances

Pour chaque descripteur di, trouver le(s) descripteur(s) dj le(s) plus proche(s)