PROJET IG.2405

Reconnaissance automatique de la signalisation des lignes de Métro parisien

1. Sujet

La société IMAGIK souhaite développer un produit d'assistance à la mobilité dans le métro parisien pour les personnes malvoyantes. Pour cela, elle développe une application sur smartphone couplée à une paire de lunettes spéciales équipée d'une caméra. L'utilisateur indique dans l'application la station de métro destination. L'application récupère les images acquises par la caméra, les traite pour extraire les informations relatives à la signalisation du métro, et indique à la personne la direction à prendre.

La société IMAGIK charge des élèves de l'ISEP du module de reconnaissance des lignes de métro dans des images extraites du flux vidéo. Les lignes qui devront être reconnues sont indiquées dans la figure ci-dessous :

FIG. 1. Lignes du métro parisien

Ainsi, le programme de démonstration demandé devra fournir pour chaque image la localisation des signes de lignes de métro détectés et le numéro de la ligne pour chaque signe.

(a) Image source

(b) Détection et localisation

(c) Reconnaissance : lignes 1, 14, 7, 11

FIG. 2. Exemple de résultat d'analyse

Etant donné les capacités de mémoire et de calcul limitées sur un smartphone, il est exigé de concevoir un système qui fonctionne en deux étapes :

- Une sélection des zones d'intérêt dans lesquelles se trouvent des pictogrammes candidats
- Une reconnaissance des pictogrammes candidats sur ces zones d'intérêt.

2. Bases de données

Ce sujet est une application typique de vision par ordinateur, comprenant les étapes de prétraitements, segmentation, reconnaissance. Les étapes de segmentation et de reconnaissance nécessiteront de définir un certain nombre de paramètres, comme par exemple les couleurs recherchées, les dimensions des objets, des seuils de similarité, etc. ou d'entrainer un réseau de neurones. Il faut impérativement mettre au point le système sur une **base d'apprentissage**. La capacité de généralisation des algorithmes sur des images non apprises sera évaluée sur une **base de test**. C'est une méthodologie générale qu'il faut impérativement

respecter dans tout problème de reconnaissance supervisé (i.e. mis au point à partir de données connues).

L'ensemble des images est dans le répertoire **BD**. Il y en a à ce jour 261…

On a retenu 1/3 des images pour l'apprentissage (images 3, 6, 9, ….) et 2/3 des images pour le test (images 1, 2, 4, 4, 7, 8,…). Le fichier **Apprentissage.xls** résume les signes de métro qui sont dans les images d'apprentissage : pour chaque signe, numéro de l'image source, coordonnées de la boite englobante, numéro de la ligne de métro reconnue. Le fichier **Test.xls** a la même structure pour les images de la base de test.

Il peut être nécessaire de définir une base de validation pour optimiser des hyperparamètres. Dans ce cas vous prendrez sur la base d'apprentissage et vous ferez vous-même la séparation des images répertoriées dans **Apprentissage.xls**.

Les fichiers Apprentissage.mat et Test.mat sauvegardent les mêmes données, mais sous la forme de fichiers de données matlab. Ces fichiers sont facilement lisibles en python par les commandes.

3. Challenge et structure des programmes finals

Vous trouverez dans l'archive **ProjetMetroChallenge.zip** le cadre de développement permettant de participer au challenge.

- **BD_CHALLENGE :** les images du challenge (pour l'instant, ce sont des images de la BD actuelle, cela sera remis à jour pour le challenge avec de toutes nouvelles images).

- **metroChallenge.py** : le programme principal à lancer qui :
    - Vous permet de définir un facteur de redimensionnement, par exemple resize_factor = 0.5 si on veut travailler sur des images 2x plus petites.
    - Liste les images du répertoire **BD_CHALLENGE** .
    - Lance une boucle sur ces images pour les traiter séquentiellement par la fonction **processOneMetroImage** qui se trouve dans **myMetroProcessing.py**.
    - Concatène les résultats (bd) et les stocke dans le fichier .mat défini dans la variable **file_out**
    - Vous permet d'afficher image par image votre résultat (celui dans file_out) et la vérité terrain (dans **GTCHALLENGETEST.mat**) via la fonction **compareTestandRef** (dans **evaluationV2.py**).
    - Evalue les performances via la fonction **evaluation** (dans **evaluationV2.py**).

  La fonction **evaluation** permet de calculer automatiquement les métriques en comparant les résultats de reconnaissance (**file_out**) avec la vérité terrain (**GTCHALLENGETEST.mat**). La ligne de code :

  ```
  evaluation('GTCHALLENGETEST.mat', 'teamsEX.mat', 0.5)
  ```

  montre ce qui est calculé sur un exemple de reconnaissance stocké dans 'teamsEX.mat', les images ayant été dimensionnées d'un facteur 0.5 pour être traitées.

Pour le challenge, il faudra préparer une archive pour que le correcteur n'ait qu'à

1. Actualiser les images du répertoire et le fichier de la vérité terrain **GTCHALLENGETEST.mat**
2. Lancer metroChallenge

4. Rapport et soutenance

Rapport :

Ce rapport doit être un document bien écrit et rigoureux, rédigé à la manière d'un article scientifique. Il doit contenir les items suivants :

- Position du problème
- Etat de l'art (succinct). Citer les référence par des numéros [1],[2,], …
- Description des méthodes : attention, il ne s'agit pas de recopier du code, mais de formaliser par des équations les traitements appliqués. **Aucun code ou pseudo-code ne doit être présent dans le rapport.** Il faut commencer par présenter les principales étapes du programme (prétraitements, segmentation, …) par un schéma fonctionnel. Pour chaque étape, introduire les méthodes et les variables avec des phrases et formaliser mathématiquement les méthodes par des équations. Bien mettre en évidence les paramètres du programme. Illustrer par des images de résultats intermédiaires.
- Résultats expérimentaux et discussion.
- Conclusion et perspectives.
- Références bibliographiques numérotées (articles, liens internet …) qui sont citées dans le corps du rapport.

NB : il n'est pas interdit de reprendre des codes existants mais les sources doivent être citées avec exactitude, même s'il ne s'agit que d'une partie du programme final. **Sinon, le travail s'apparentera à un plagiat et cela sera sanctionné par un 0**. Les codes repris doivent être maîtrisés, expliqués et discutés, voire améliorés. La note attribuée prendra en compte l'apport personnel.

Soutenance :

La **soutenance dure 20mn** par équipe, avec 12 mn de présentation et 8mn de questions. Bien respecter ce timing, il ne sera pas possible de déborder sur le temps de présentation.

Préparer une présentation PowerPoint. La présentation PowerPoint devra être réalisée dans le même esprit que le rapport.

5. Livrables :

- Codes python : fournir une archive identique à ProjetMetroChallenge.zip, avec le répertoire BD_CHALLENGE vide, et l'ensemble de vos codes .py. Attention, le correcteur lancera le programme principal metroChallenge.py, et cela devra fonctionner sans qu'il ait à debugger quoi que ce soit ! Faites les tests nécessaires pour valider votre archive avant l'envoi.
- Le rapport au format pdf.

L'archive et le pdf doivent être envoyé à florence.rossant@isep.fr et Nan.Ding@isep.fr, avec comme titre du mail « **projet IG2405 Equipe n°X** », et avec tous les membres de l'équipe en copie, et ceci **avant le lundi 16 Juin 2025, 13h00**.

6. Notation

Compétences évaluées

- Capacité à concevoir un système de vision artificielle répondant à un problème posé. Pertinence

dans le choix des méthodes. Apport personnel.

- Vue systémique (schéma fonctionnel).
- Vue systémique (schéma fonctionnel).
- Capacité à expliquer et à formaliser les méthodes proposées (rapport, présentation).
- Qualité de la rédaction et de la présentation orale : structure, rigueur, orthographe, clarté, etc…
- Etude expérimentale : évaluation quantitative, analyse critique.
- Respect de la méthodologie et du cahier des charges : apprentissage, test, spécifications des formats de données et des programmes.
- Qualité de la programmation : programmation modulaire, structurée, paramètres bien définis (et non codés en dur), code bien commenté, etc.
- Qualité des résultats, esprit critique.

La note finale résultera de cette évaluation des compétences et prendra en compte les résultats du challenge. Les équipes seront classées sur 2 critères :

- Le score F1 de détection des signes de métro (indépendamment de la classification)
- L'accuracy élargie
- Le score F1 pondéré

```
-------------------------------------------------------
SIGN DETECTION
-------------------------------------------------------
         recall    = 0.975
         precision = 0.987
         F1-score  = 0.981
-------------------------------------------------------
GLOBAL AND ENLARGED ACCURACY
-------------------------------------------------------

Global enlarged accuracy = 0.865

-------------------------------------------------------
CLASS EVALUATION REPORT
-------------------------------------------------------
Ligne           Precision      Recall  f1-score       Support
   1              1.000         0.962    0.980          26
   2              0.933         0.875    0.903          16
   3              0.875         0.875    0.875           8
   4              1.000         0.818    0.900          11
   5              0.615         0.889    0.727           9
   6              0.778         1.000    0.875           7
   7              0.889         1.000    0.941           8
   8              0.846         0.688    0.759          14
   9              1.000         1.000    1.000           1
  10              0.500         0.600    0.545           5
  11              0.875         0.875    0.875           7
  12              0.941         0.842    0.889          18
  13              1.000         0.900    0.947          10
  14              0.941         0.941    0.941          17
-------------------------------------------------------
Macro             0.871         0.876    0.868         14 classes
Weighted          0.901         0.878    0.885         157 signs
```

Un bonus sera donné aux équipes qui arriveront sur le podium pour l'un de ces 3 critères au moins !