BE noté : Classification des cibles radar à partir des images images ISAR (Radar à synthese d'ouverture inverse)

Dans ce BE, nous traitons des données ISAR issues de la chambre anéchoïde de l'école pour mettre en place la chaine de reconnaissance de cibles. Pour le faire, nous réalisons les différentes étapes du processus de reconnaissance de formes (Pattern recognition).

Remarque : La phase d'acquisition des données radar et la reconstruction des images ISAR n'est pas l'objet de ce BE.

L'objectif est d'écrire des scripts python permettant de mettre en œuvre un système de reconnaissance de cibles. Ces scripts devront ainsi suivre la chaîne générale décrite en cours regroupant ainsi les phases de :

- Prétraitements
- Extraction des descripteurs
- Implémentation d'un modèle de classement (apprentissage)
- Classification des images
- Evaluation des performances (précision, temps de calcul, …)

Pour commencer la séance, vous aurez besoin de télécharger la base d'images ISAR disponible sur Moodle. Dans ce BE, il vous est demandé de fournir un rapport (il peut être un notebook regroupant les le code et les réponses aux questions et vos analyses en plus de vos codes. Vous pouvez utiliser les codes de vos séances précédentes.

Partie I. Analyse de données ISAR

1. Consulter la base de données et déterminer :
   a) Quel est le nombre de classes ? quel est le nombre d'images par classe ?
   b) Quel est la dimension de votre espace de données ? la valeur min, max ?
   c) Pour réduire la dimension des données, proposer une sélection une zone d'intérêt dans chaque image pour délimiter les cibles présentes aux centres des images.

2. Définir une fonction load_bdd() permettant de charger les images d'un répertoire donné en paramètre et nous fournira la matrice de données, et des classes (labels)

Partie II. Extraction des caractéristiques discriminantes – Image polaire

Au lieu de traiter les données dans l'espace initial, nous proposons une nouvelle signature. Le principe de la signature proposée dans cette section repose sur la transformée polaire. L'objectif est de présenter l'image initiale dans un espace polaire afin d'obtenir une nouvelle image appelée *image polaire* (IMP).

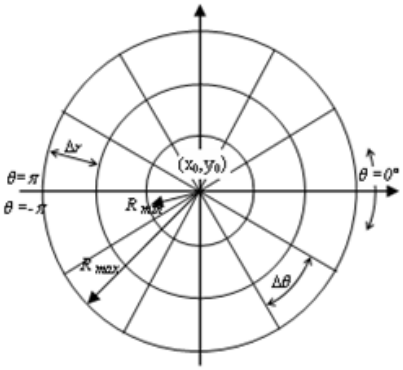

Figure 1. Plan polaire

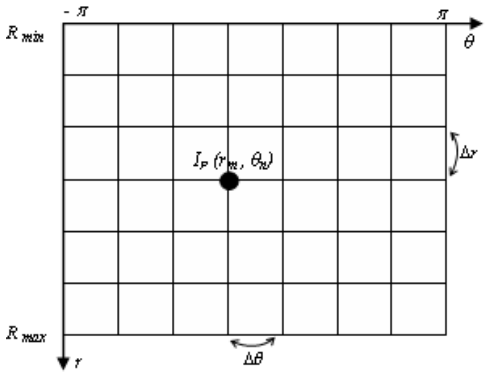

Figure 2. Image polaire

Pour une simplification dans la suite, nous notons une image ISAR par $I(x_i,y_j)$. avec $i=1,..,M$, $j=1,...,N$, où M x N représente la résolution de l'image. La transformation de l'image ISAR nous donne une image polaire notée $I_P\ (r_m\ ,\ \theta_n)$. $m=1,..,N_r$, $n=1,...,M_\theta$. *où* N_r est le nombre des points sur l'axe r et M_θ est le nombre des points sur l'axe θ.

La valeur de chaque pixel de l'image polaire de coordonnées (r_m,ϑ_n) est calculée via la valeur du pixel de l'image ISAR de coordonnées $(x_k\ ,\ y_k)$ par l'A*lgorithme-polaire* suivant :

$$I_P(r_m,\theta_n) = I(x_k,y_k)$$
$$(x_k,y_k) = (x_0,y_0) + (r_m\cos\theta_n\ ,\ r_m\sin\theta_n)$$
$$m = 1,\ldots,N_r\ et\quad n = 1,\ldots,N_\theta.et\quad k = 1,\ldots,N_rN_\theta$$
$$où\qquad r_m = R_{\min} + (m-1)\Delta r\qquad et\quad \theta_n = -\pi + (n-1)\Delta\theta$$
$$\Delta r = \frac{R_{\max} - R_{\min}}{N_r - 1};et\quad \Delta\theta = \frac{2\pi}{N_\theta - 1}$$

1. Ecrire une fonction *polaire()*[1] qui transforme une image sur le plan cartésien vers le plan polaire et retourne cette denière. Vous pouvez préciser par exemple N_r = M_θ = R_{max} = 50. *Quelle est la dimension de la nouvelle image ?*
    a. Tester cette fonction sur une image d'exemple au choix et afficher le résultat. Effectuer une rotation 45° sur la même image et afficher l'image polaire correspondante.
    b. Effectuer un changement d'échelle de l'image originale et afficher l'image polaire correspondante.
    c. Que constater vous sur l'image polaire par rapport à la rotation et changement d'échelle. Commenter en quelques lignes seulement?

2. Pour compléter cette phase d'extraction de caractéristiques, nous calculons un nouveau vecteur des caractéristiques composé de deux projections : La projection sur l'axe-r $I_r(r)$ et la projection sur l'axe-θ $I_\theta(\theta)$ données par :

$$I_r(r) = \int_{-\pi}^{\pi} I_p(r,\theta)d\theta \approx \sum_{n=1}^{N_\theta} I_P(r_m,\theta_n).\ \text{et}\ \ I_\theta(\theta) = \int_{R_{\min}}^{R_{\max}} I_p(r,\theta)dr \approx \sum_{m=1}^{N_r} I_P(r_m,\theta_n)$$

    a. Modifier la fonction polaire de la question 1 pour fournir non seulement l'image polaire mais aussi les deux vecteur Ir et I_θ. Tester votre fonction modifiée sur deux images ISAR

[1] https://scikit-image.org/docs/stable/auto_examples/registration/plot_register_rotation.html

d'une même cible et comparer le résultat visuel sur une autre image ISAR d'une cible différente. Commenter les résultats et vos observations sur Ir et I_θ.

b. Utiliser une métrique (distance euclidienne par exemple) pour évaluer la similarité entre une image et une image avec rotation. Qu'observez- vous ?

a. *Utiliser la* cross-correlation comme mesure de similarité et estimer la translation observée sur le vecteur I_θ entre deux images : image sans et avec rotation.

b. Quelle est la taille de ce vecteur de caractéristiques ? quel est le gain de dimension entre cet espace et l'espace initial de l'image ISAR initiale ?

3. Analyser ce vecteur sur la simulation de la question II. 1 et commenter vos observations.

III. Classification : reconnaissance

Nous souhaitons réaliser dans cette section, la partie classification des images ISAR en suivant l'architecteur suivante :

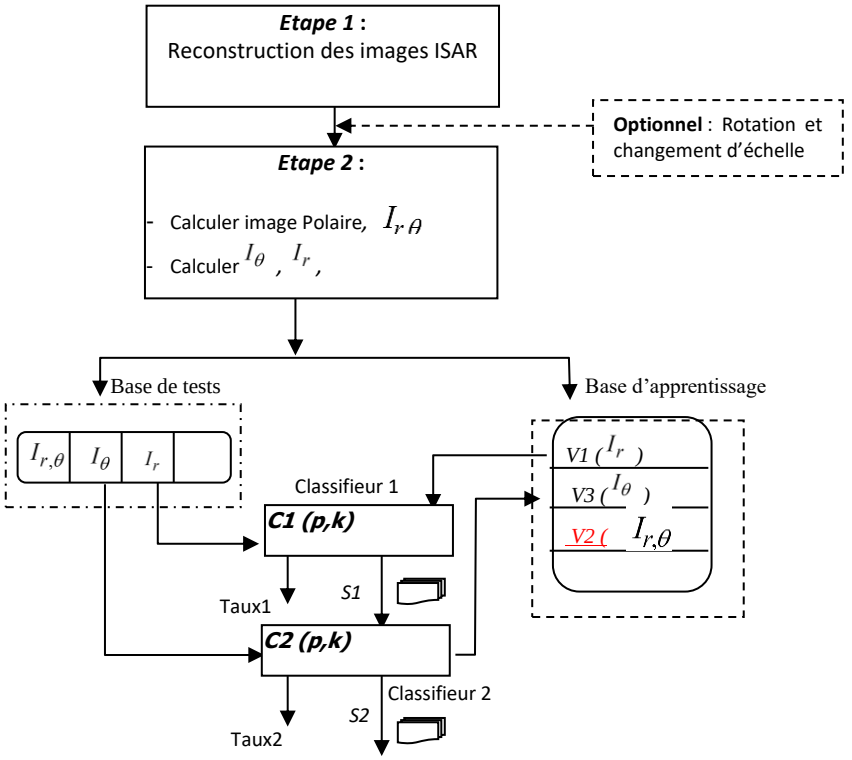

Figure 3. Architecteur de classification

Si nous notons P *comme* le nombre total des cibles dans la base d'apprentissage, et *K* est le nombre total des images de la cible *p* dans la base d'apprentissage, le premier classifieur C1 utilise le vecteur *Ir* de l'image à reconnaître (de la base de test) pour calculer les *P.K* mesure de similarité/dissimilarité tel que *P*x*K* soit le nombre total des images de la base d'apprentissage. Ensuite, les *P*x*K* coefficients sont ordonnés par ordre croissant (ou décroissant selon la métrique utilisée) décroissant et l'index des images correspondant à cet ordre est récupéré. Enfin, seulement un pourcentage λ % de cet index sera envoyé au classifieur C2. Nous notons *S1* le vecteur des index envoyé par le premier classifieur vers le second classifieur C2. Cette idée est fondée sur l'hypothèse que l'image la plus proche de l'image requête se trouve parmi les λ % images de la base d'apprentissage dont les mesures de similarité sont les plus petites. Dans la suite, la recherche au niveau de C2 sera focalisée seulement sur l'ensemble *S1* et non sur la totalité de la base d'apprentissage. Pour le classifieur C2, on calcule la mesure de similarité *C2*(*p,k*) sur le vecteur

des descripteurs I_θ entre une image requête et les images de l'index *S1* (ie $(p,k) \in S1$). Les coefficients $C2(p,k)$ sont ensuite ordonnés par ordre croissant (ou décroissant selon la mesure de similarité). A noter que l'image polaire ainsi que la projection sur l'axe-θ sont décalées par rapport à l'image originale. Il est donc nécessaire de prendre en considération ce décalage afin de bien mesurer la similarité/dissimilarité. Pour mettre en place ce système, nous réalisons les taches suivantes :

1. Ecrire un module DecripteursBDD.py' pour charger toute la base des images ISAR et calculer le vecteur des caractéristiques pour chaque image. Dans ce programme, une matrice 'data' (nb_images x descripteurs) est constituée et sauvegardée.

2. Ecrire un script 'reconnaissance.py' et commencer par charger la matrice 'data de la question précédente. Constituer deux ensembles, un d'apprentissage de 2/3 de la base totale et le reste est dédié pour la base de test. Générer pour les deux ensembles, leurs vecteurs des labels.

Pour le premier classifieur (noté C1), utiliser le classifieur k-ppv. Dans ce classifieur, nous utilisons seulement une partie de la signature polaire I_r (Cf. Figure 3).

3. Evaluer les performances du classifieur C1 en terme de taux de classification et temps de calcul) pour k=1 ,3 et 5 pour un vecteur caractéristique contenant seulement I_r. Commenter les résultats sur les différentes classes étudiées. Assurez- vous que le classifieur C1 retourne la ***matrice de distance*** entre chaque image de la base de test et toute la base de référence (base d'apprentissage)

4. Changer le vecteur caractéristique par I_θ et puis (I_θ + I_r) dans le classifieur C1 et évaluer les résultats obtenus. Commenter les résultats.

5. Mettre en place le classifieur C2 et évaluer ses performances (taux de classification et temps de calcul) pour k=1 ,3 et 5. Nous rappelons ici que pour le classifieur C2, seulement λ% (prenez 30% par exemple) de l'index envoyé par C1 est analysé et non pas toute la base d'apprentissage initiale.

    - Implémenter le classifieur C2 qui utilisera seulement le vecteur I_θ (Cf. Figure1) avec k=1,3 et 5. Utiliser le cross-corrélation comme métrique de similarité.
    - Evaluer les performances du classifieur C2 en terme de taux de classification et temps de calcul) pour k=1 ,3 et 5 et commenter les résultats.

Bonus

Cette partie est optionnelle mais elle reste recommandée pour ceux qui souhaitent aller plus loins.

Nous ajoutons dans l'architecture de la figure 1 un troisième classifieur C3 qui se présente en sortie du classifieur C2. En effet, les mésure de similarité/dés similarité $C2(p,k)$ sont ordonnés par ordre croissant/*décroissant* et de la même manière que précédemment, seulement les premières $\eta\%$ images de nouvel indexe sont retenues. Ce vecteur d'indexe noté *S2* est envoyé au classifieur C3. L'hypothèse est que l'image la plus proche de l'image requête se trouve parmi les $\eta\%$ images de ce nouvel indexe,

Le dernier classifieur quant à lui, cherche l'image la plus proche de l'image requête dans les images référencées par *S2* en se basant sur le vecteur V3 de la signature (voir figure 2). Par contre, une compression de l'image requête avec l'ACP, est réalisée en utilisant la matrice de projection calculée par application de l'ACP sur les images de la base d'apprentissage.

Dans la partie II, vous avez pu constater que l'image polaire ainsi que la projection sur l'axe-θ sont décalées par rapport à l'image originale. Il est donc nécessaire de calculer ce décalage afin de décaler l'image requête avant de la compresser. Une approche systématique qui estime la valeur du décalage de translation (*shifts translation*) entre deux vecteurs I_θ en utilisant la cross-correlation. C'est ainsi qu'il est estimé la valeur du décalage nécessaire entre les deux vecteurs I_θ et V2 pour aligner l'image polaire (image requête) avant de la compresser.

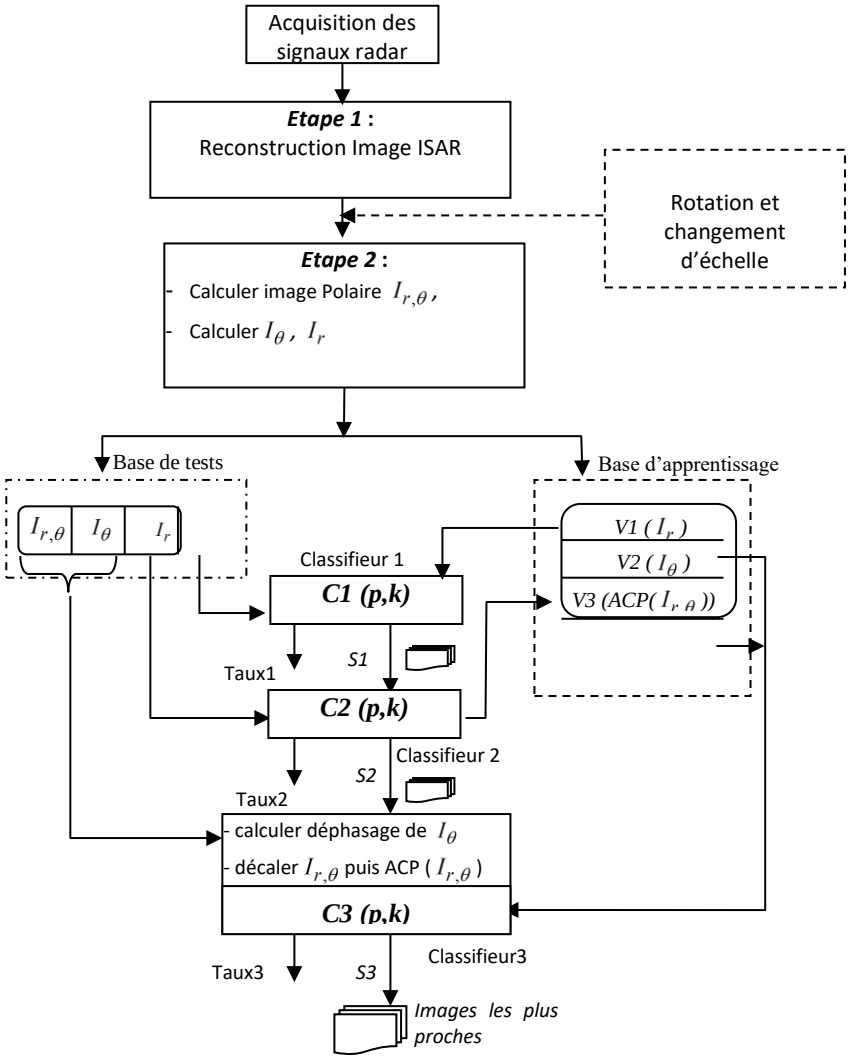

FIG. 2 – Architecture de reconnaissance.

- Implémenter le classifieur C3 qui utilisera l'image polaire compressée (Cf. Figure2) avec k=1,3 et 5. Utiliser la distance euclidienne comme métrique de similarité.
- Evaluer les performances du classifieur C3 en terme de taux de classification et temps de calcul) pour k=1 ,3 et 5 et commenter les résultats.