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

Dans ce BE noté, nous traitons des données ISAR issues de la chambre anéchoïde de l'école pour mettre en place la chaîne 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 (dans un notebook) 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 notebook regroupant les différentes fonctions demandées en rappelant la même numérotation des questions avec les réponses aux questions et vos analyses et commentaires. N'afficher pas des sorties/figures de scripts/fonctions sans les commenter. Vous pouvez utiliser les scripts de vos séances ML précédentes.

Avant de commencer ce BE, lisez la totalité du sujet et n'oublier pas d'indiquer les deux nom/prénom de votre binôme.

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 et afficher dans un histogramme cette répartition.
    b) Quel est la dimension de votre espace de données ? La valeur min, max ?
    c) Pour réduire la dimension des données, proposer comme premier traitement une sélection d'une zone d'intérêt contenant les cibles présentes aux centres des images. Vous pouvez délimiter les cibles par une zone plus réduite que l'image d'origine. Quel est votre gain dans votre espace de représentation ?

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 le vecteur 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[1]. L'objectif est de présenter l'image initiale dans un espace polaire afin d'obtenir une nouvelle image appelée *image polaire* (IMP) comme illustré par les figures 1et 2.

[1] Une transformation log-polaire est aussi utilisée dans ce contexte.

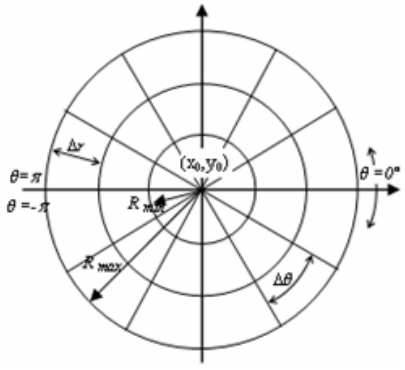

Figure 1. Plan polaire

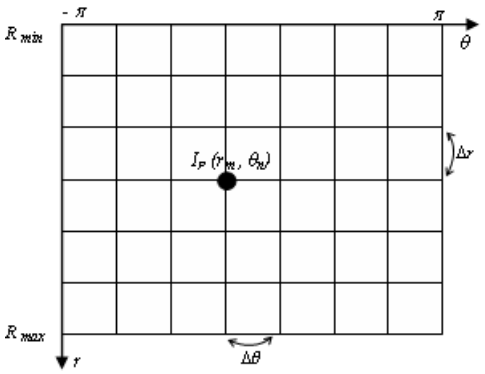

Figure 2. Image polaire

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. Nous notons que N_r x M_θ est la résolution de l'image polaire.

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 :

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

1. Ecrire une fonction *polaire()*[2] qui transforme une image sur le plan cartésien vers le plan polaire et retourne cette dernière. Vous pouvez préciser comme valeur par défaut par exemple $N_r = M_\theta = R_{max} = 50$. Quelle est la dimension de la nouvelle image ? quelle est la taille de l'espace de représentation ?
2. Nous souhaitons évaluer l'invariance de cette image polaire à certaines transformations géométriques (rotation, changement d'échelle, translation, etc).
   a. Tester l'invariance de m'image polaire à la rotation (par exemple 45°, 90°, 180°) sur un exemple d'image au choix et afficher et commenter les résultats.
   b. Effectuer un changement d'échelle d'une image originale et afficher l'image polaire correspondante en ajustant les paramètre d'entrée de la fonction *polaire* de la question 1 et commenter les résultats.
   c. Sans proposer un script, comment peut on assurer une invariance à la translation ?

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

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

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

a. Modifier la fonction *polaire() (vous pouvez la noter polaire1())* de la question 1 pour fournir non seulement l'image polaire mais aussi les deux vecteur I_r et I_θ. Evaluer l'invariance de ces deux vecteurs à la rotation et au changement d'échelle et commenter les résultats ? Réaliser une distance euclidienne pour évaluer la similarité entre une image et sa rotation en se basant sur le vecteur I_r , I_θ et l'image polaire. Commenter les résultats.

b. *Utiliser la* cross-corrélation ou une autre métrique comme mesure de similarité et estimer la translation observée sur le vecteur I_θ entre une image et elle-même avec rotation.

c. Si on prend comme vecteur de caractéristique, l'image polaire et les deux vecteurs de projection, quelle est la dimension de l'espace de caractéristiques ? quel est le gain de dimension entre cet espace et l'espace initial de l'image ISAR initiale ?

Partie III. Classification : reconnaissance

Nous souhaitons réaliser le système de reconnaissance de cibles en utilisant dans une premère partie la combinaison série de deux classifieurs supervisés suivant l'architecteur présentée par la figure 3 ou c1 et c2 sont de classifieurs (de type k-ppv)

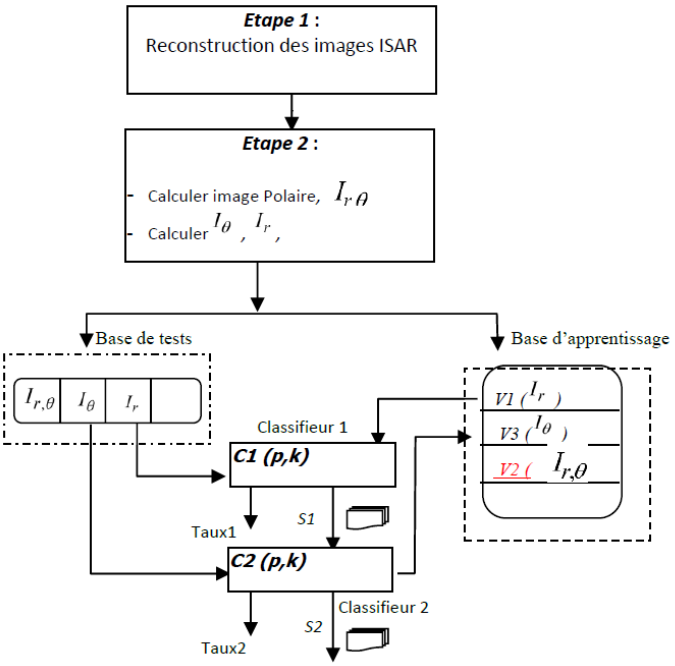

Figure 3. Architecteur de classification

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 *PxK* soit le nombre total des images de la base d'apprentissage. Ensuite, les *PxK* 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 Iθ entre une image requête et les images de l'index S1 (ie (p,k)∈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 une fonction *decripteursBDD()* qui charge depuis un répertoire donné 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)+*labels* est constituée et sauvegardée en disque. Afficher la taille de cette matrice.

2. Ecrire la fonction *reconnaissance ()* qui charge la matrice 'data' donnée en paramètre et commencer par constituer la base d'apprentissage (contenant 2/3 de la base totale) et 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 (Cf. Figure 3).

3. Evaluer les performances du classifieur C1 en terme de taux de classification et en 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.
    a. Implémenter le classifieur C2 qui utilisera seulement le vecteur I_θ (Cf. Figure1) avec k=1,3 et 5. Utiliser le cross-corrélation (ou l'autre métrique implémentée dans la question II.3.b) comme métrique de similarité. Nous souhaitons aussi avoir les translations estimées sur toutes les images indexées dans S1.
    b. Evaluer les performances du classifieur C2 en termes de taux de classification et temps de calcul) pour k=1 ,3 et 5 et commenter les résultats.
6. Afin d'étudier la robustesse de notre architecture, appliquer une rotation et changement d'échelle aléatoires en aval de l'étape 2 et donner les performances de bonne classification pour le c1 et c2 sur 5 simulations (garder toujours un tirage aléatoire sur la base d'apprentissage/test)

Partie III.A Bonus

Cette partie est optionnelle mais elle reste notée et très recommandée pour ceux qui souhaitent aller plus loin.

Nous ajoutons dans l'architecture de la figure 3 un troisième classifieur C3 qui se présente en sortie du classifieur C2 comme illustré par la figure 4. En effet, les mesures de similarité/dissimilarité $C2(p,k)$ sont ordonnées par ordre croissant/décroissant de la même manière que précédemment, seulement les premières ŋ % 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 ŋ % 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 3). Par contre, une compression (ou réduction) de l'image requête avec l'ACP, est réalisée en utilisant la matrice de projection calculée par application de l'ACP sur toute les images de la base d'apprentissage.

Dans la partie II, vous avez pu constater que la projection sur l'axe- θ n'est pas invariante à la rotation qui se présente sous format d'un décalage par rapport à l'image sans rotation. Il est donc nécessaire de calculer ce décalage afin de décaler l'image requête avant de la compresser (avant d'applique la matrice de rotation calculée depuis la base d'apprentissage). Une approche systématique qui estime la valeur du décalage de translation (shifts translation) entre deux vecteurs I_θ en utilisant la cross-corrélation (ou autre métrique utilisée dans II.3.b). 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.

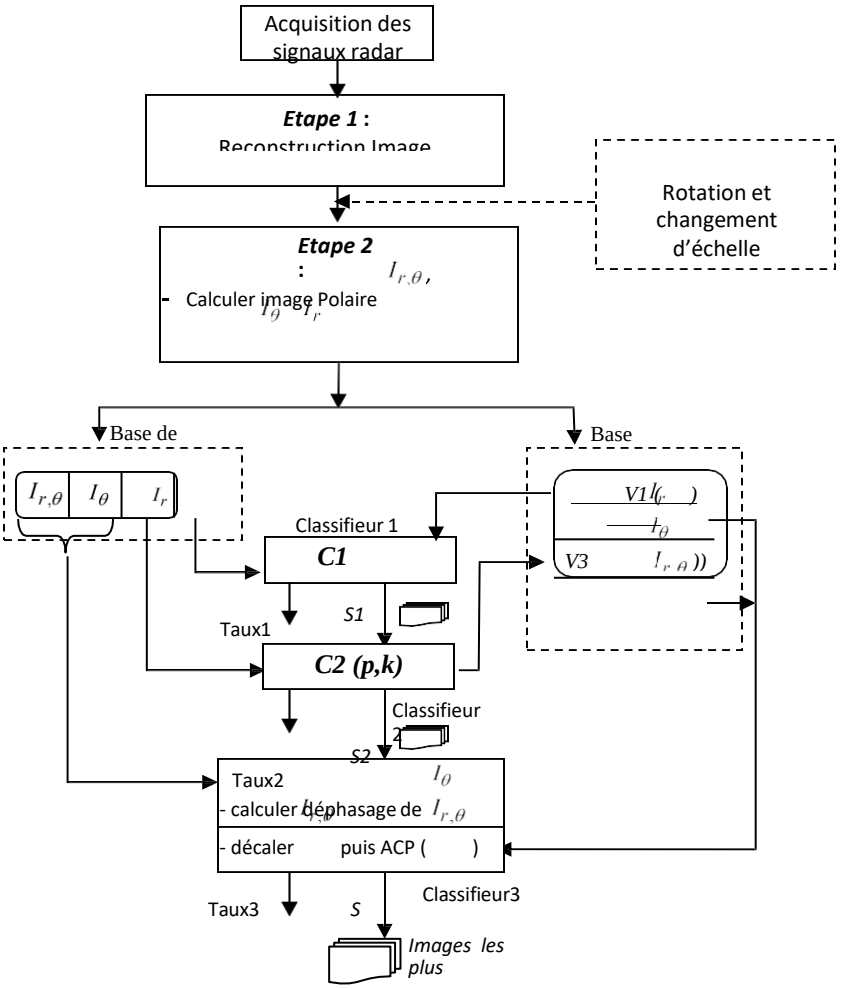

FIG. 4 – Architecture complète de reconnaissance.

a. Implémenter le classifieur C3 qui utilisera l'image polaire compressée comme vecteur caractéristique avec k=1,3 et 5. Utiliser la distance euclidienne comme métrique de similarité.

b. Evaluer les performances du classifieur C3 pour k=1 ,3 et 5 et commenter les résultats en ajoutant une rotation et changement d'échelle aléatoires sur seulement la base de test.