TD V : Inspection de formes par vision

V.1/ Descripteur de Fourier

L'image '`piece_ref.bmp`' représente une pièce étalon. A partir de cette image, on souhaite construire une « signature visuelle » du contour de cette pièce. On utilisera pour cela les descripteurs de Fourier (Annexe 1).

a/ Charger l'image '`piece_ref.bmp`'. La pièce étant grise sur fond clair, binariser l'image afin de séparer les pixels de la pièce de ceux du fond.

```
Ib = not(im2bw(I,.85));
```

b/ Extraire les contours de la pièce grâce à la fonction `bwboundaries()` qui calcule retourne les chaines de pixels formant des contours fermés.

```
b = bwboundaries(Ib);
b = b{1};
figure(1); hold on, plot(b(:,2),b(:,1),'.');
```

c/ Ecrire la fonction `f=descripteur_fourier(b,nb_elem)` pour construire le descripteur de Fourier en suivant les étapes ci-dessous :

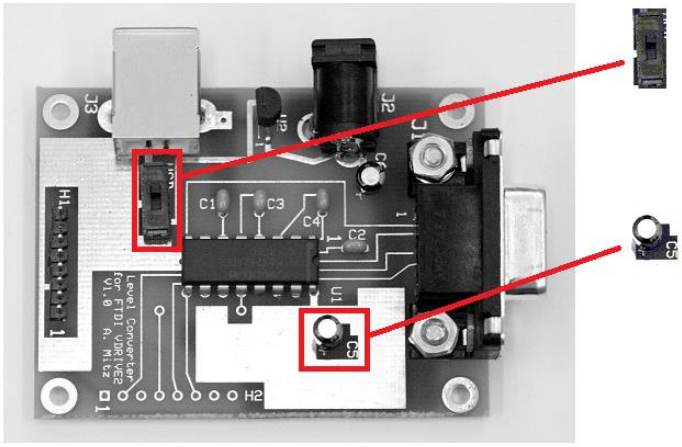
- Construire la suite de nombres complexes à partir des pixels du contour
  `s = b(:,1) + i*b(:,2);`
- Calculer les coefficients de la décomposition en séries de Fourier grâce à la fonction `fft`.
- Garder uniquement les `nb_elem` premiers coefficients significatifs.
- Mettre le premier coefficient à 0.

d/ Afficher le module du descripteur grâce à la fonction stem.

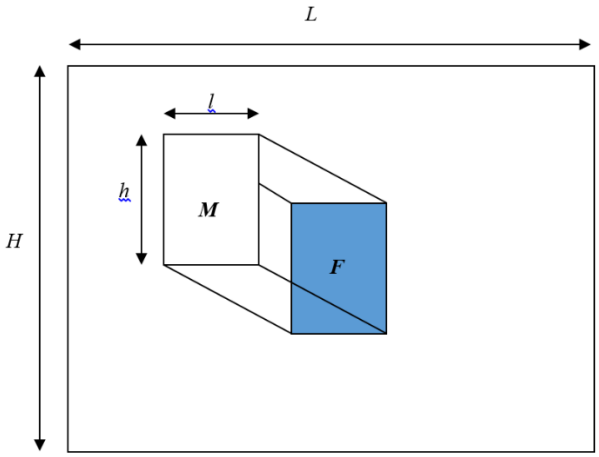
e/ Ecrire une script capable de trier les pièces de l'image '`im_test.bmp`' (conforme/non conforme) en utilisant la distance entre les descripteur de Fourier et celui de la pièce de référence.

V.2/ Exemple de contrôle par mesure d'inter-corrélation entre deux régions

Dans le cadre d'une application de contrôle qualité, on souhaite automatiser l'inspection d'une carte électronique en bout de chaine de production. Le contrôle consiste à vérifier la présence de certains composants sur la pièce finie avant sa commercialisation. Les composants en question sont enregistrés dans une base de données sous forme d'imagettes de références prises dans des conditions dites normales.



Le traitement consiste à acquérir l'image courante d'une pièce en bout de chaine de production puis de l'inspecter grâce au score de corrélation entre cette image et chacune des imagettes représentant un composant. Si le test est concluant (c-à-d si les éléments sont retrouvés et au bon endroit) on valide la pièce en mettant en évidences ces éléments dans l'image.



Le score de corrélation centrée normée entre deux signaux 2D, ici le motif $\boldsymbol{M}$ et la fenêtre $\boldsymbol{F}$ de même taille h x l est donné par la formule ci-dessous :

$$corr(M,F) = \frac{\sum_{i=1}^{h}\sum_{j=1}^{l}(M(i,j)-\bar{M}).(F(i,j)-\bar{F})}{\sqrt{A.B}}$$

avec

$A = \sum_{i=1}^{h}\sum_{j=1}^{l}(M(i,j)-\bar{M})^2$

$B = \sum_{i=1}^{h}\sum_{j=1}^{l}(F(i,j)-F)^2$

a/ Ecrire la fonction de corrélation 2D centrée normée ci-dessus qui reçoit comme arguments deux imagettes de même dimension et qui retourne le score de corrélation. Pourquoi dit-on de cette mesure qu'elle centrée et normée ? Quel est l'intérêt de centrer et de normaliser la mesure dans le cadre de cette application ?

b/ Le principe de la reconnaissance de forme consiste à calculer le score de corrélation du motif recherché en balayant toute l'image dans laquelle on recherche sa présence (comme pour la convolution). Si ce score dépasse un seuil *s* (à déterminer pour chaque motif), alors on décide que la ressemblance est suffisante.

Tester la fonction précédente en recherchant, dans l'image « pcb_1.bmp », le motif X représenté par l'image « motif_X.bmp ». Encadrer d'un rectangle rouge chaque fenêtre de l'image ayant obtenu un score de corrélation avec le motif supérieur au seuil. Utiliser pour cela les instructions ci-dessous :

```
figure(1), hold on;
h = rectangle('Position',[j_max(1),i_max(1), l, h]);
get(h);
set(h,'EdgeColor','red');
```

c/ Refaire le test avec l'image « pcb_2.bmp » puis « pcb_3.bmp ». Expliquer les différences entre ces cas et leur effet (ou pas) sur la mesure de corrélation. Conclusion ?

d/ On souhaite écrire une fonction de corrélation qui puisse prendre en compte les rotations (pour traiter le cas de l'image « pcb_3.bmp »). Réécrire la fonction précédente en prenant le soin de redresser les pixels de l'image tournée d'un angle ϑ adéquat.

Pour cela écrire la fonction «I_r= rotate_image(I,ϑ)» qui permet de calculer pour chaque pixel de l'image retournée I_r, sa position dans l'image originale et de le remplacer par le niveau de gris lu à cette position. En d'autres termes, on exprime les pixels dans un nouveau repère dont les axes forment un angle ϑ avec les axes du premier repère image.

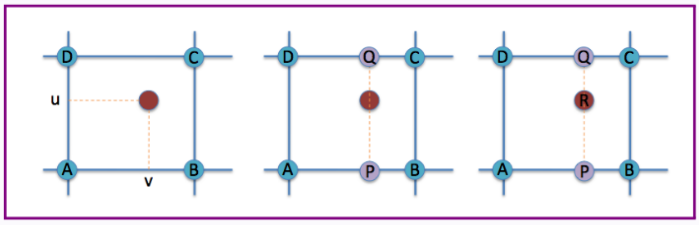
Le pixel d'indices (p,q) dans la nouvelle image, correspond au pixel (i,j) dans l'ancienne. La relation entre les deux systèmes de coordonnées s'écrit :

$i = (p-u_0)*\cos(\vartheta) - (q-v_0)*\sin(\vartheta)) + u_0$;

$j = (p-u_0)*\sin(\vartheta) + (q-v_0)*\cos(\vartheta)) + v_0$;

Certains pixels de l'image retournée correspondent à des pixels inexistants dans l'image d'origine (en dehors du cadre). Il faudra donc les remplacer par un niveau de gris nul.

Cette transformation ne donne pas forcément des indices entiers. Noter la qualité du résultat obtenu à cause de cet « arrondis ». Pour améliorer ce résultat, il faut calculer le niveau de gris en interpolant les niveaux des quatre pixels voisins comme expliqué sur la figure ci-dessous.



$$
\begin{aligned}
P &= (1-v)A + vB \\
Q &= (1-v)D + vC \\
R &= (1-u)P + uQ \\
&= (1-v)(1-u)A + (1-u)vB + uvC + u(1-v)D
\end{aligned}
$$

e) Compléter le test par la vérification du bon positionnement des différents éléments. Tester votre programme sur l'image « pcb_4.bmp ». Utiliser l'image 1 comme référence.