

AIV1 - TP05
Moments d'une forme

Suliac Lavenant et Antoine Nollet

11 February 2022

Table des matières

| | | |
|----------|---|----------|
| 1 | Code Python | 3 |
| 2 | Moments cartésiens d'une forme | 3 |
| 2.1 | Barycentre d'une forme | 3 |
| 2.2 | Moment cartésien centré d'une forme | 4 |
| 2.3 | Matrice d'inertie | 4 |
| 2.4 | Exemples pratiques 1 | 5 |
| 2.5 | Exemples pratiques 2 | 6 |
| 3 | Moments normalisés | 7 |
| 4 | Moments invariants | 8 |

1 Code Python

Nous nous intéressons à la reconnaissance de forme dans une image dans le cadre du domaine du traitement d'images. Pour cela, les indices de formes sont utilisées. Un indice de forme est un attribut de forme (qui décrit une forme) qui sera défini indépendamment de la position, de la rotation et de l'échelle de cette même forme. Ainsi donc, nous nous intéressons aux moments cartésiens.

Le moment cartésien d'ordre pq (où p et q sont des entiers positifs) d'une forme est défini ainsi :

$$M_{pq} = \sum_x \sum_y x^p y^q I(x, y)$$

Il sera codé de cette manière :

```
1 import numpy as np
2
3 def moment( im, p, q ):
4     ny, nx = image.shape
5     x = np.arange( nx )
6     y = np.arange( ny )
7     xp = np.power( x, p )
8     yq = np.power( y, q )
9     return np.dot( yq, np.dot( im, xp ) )
```

En effet, par un soucis d'optimisation du temps de calcul, nous utiliserons la librairie numpy de python. Dans les variables nx et ny seront affectées respectivement à la largeur et à la hauteur de l'image dont on veut calculer le moment. Dans les variables x et y seront affectées des listes allant de 0 jusqu'à nx-1 et ny-1. Ensuite, les variables xp et yq seront les listes contenant les valeurs de x et y qui seront montées à la puissance p et q.

Enfin, nous utiliserons le produit scalaire sur yq et le produit scalaire de l'image et xp.

Cela s'accompagnera de plusieurs particularités de numpy :

- 1 - un entier fois une liste renvoie la liste dont tout les éléments ont été multiplié par l'entier
- 2 - une liste plus une autre liste renvoie la liste des sommes des éléments des listes

Donc, lorsque nous ferons le produit scalaire de l'image et des xp, nous ferons la somme des produits des éléments de xp (des entiers) et des lignes de l'image (des listes). Les produits nous retournerons des listes (1) que nous allons additionner (2) pour obtenir le produit scalaire qui sera donc ici une liste. Enfin, nous ferons le produit scalaire de deux listes de une dimension, yq et celle obtenue précédemment, qui nous retournera un entier qui correspondra au moment cartésien voulu.

2 Moments cartésiens d'une forme

Nous avons vu le moment cartésien, mais cet attribut de forme n'est pour l'instant pas un indice de forme, il est n'est pas indépendant de la position, rotation et échelle. Dans cette partie, nous allons voir comment obtenir un attribut de forme indépendant de la position et de la rotation.

2.1 Barycentre d'une forme

Le barycentre est issu d'une combinaison particulière de moments cartésiens. Le barycentre est ainsi défini par la formule : $(\bar{x}, \bar{y}) = (M_{10}/M_{00}, M_{01}/M_{00})$

Pour obtenir notre barycentre il nous faut donc réutiliser les moments cartesiens calculés précédemment. Il correspond au centre de la forme.

Il sera codé ainsi :

```
1 m00 = moment( image, 0, 0 )
2 m10 = moment( image, 1, 0 )
3 m01 = moment( image, 0, 1 )
4
5 barycentreX = m10 / m00
6 barycentreY = m01 / m00
```

2.2 Moment cartésien centré d'une forme

Grâce au barycentre de la forme, nous allons pour calculer le moment cartésien centré. Il est défini ainsi :

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x, y)$$

Et sera codé ainsi :

```
1 def momentCentre(im, barycentre, p, q):
2     ny, nx = im.shape
3
4     x = np.arange( nx )
5     xx = x-barycentre[0]
6     xxp = np.power(xx, p)
7
8     y = np.arange( ny )
9     yy = y-barycentre[1]
10    yyq = np.power(yy, q)
11
12    return np.dot(yyq, np.dot(im, xxp))
```

En effet, dorénavant, le calcul du moment se fait grâce au barycentre. De ce fait, le moment cartésien centré devient un attribut de forme indépendant de la position, il est donc invariant de translation. Il faut ensuite trouver un moyen d'obtenir un attribut invariant de rotation...

2.3 Matrice d'inertie

Pour obtenir un attribut de forme invariant de translation et de rotation, nous nous intéressons à la matrice d'inertie qui est défini par le moment centré. La matrice d'inertie est composée de plusieurs moments cartésiens centrés d'une forme que l'on peut maintenant calculer.

Cette matrice d'inertie est défini par la matrice suivante :

$$I = \begin{pmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{pmatrix}$$

La matrice d'inertie indique comment la masse de l'objet est répartie dans l'espace par rapport aux différents axes de rotation.

Elle sera codé comme cela :

```
1 def getInertieMatrice(im, barycentre, theMomentCentre=momentCentre):
2     mu11 = theMomentCentre(im, barycentre, 1, 1)
3     mu20 = theMomentCentre(im, barycentre, 2, 0)
4     mu02 = theMomentCentre(im, barycentre, 0, 2)
5
6     return np.array([mu20, mu11], [mu11, mu02])
```

Grâce à cette matrice d'inertie, nous pourrons obtenir les axes principaux d'inertie et les moments principaux d'inertie de la forme. En effet, en diagonalisant cette matrice, nous obtenons des valeurs propres qui correspondent aux axes principaux, et nous obtenons des vecteurs propres qui correspondent aux moments principaux.

Nous allons pouvoir utiliser la fonction `numpy.linalg.eig` pour obtenir les valeurs et vecteurs propres de notre matrice d'inertie.

Nous pouvons ainsi déterminer (arbitrairement) l'axe principale d'inertie de la forme comme étant celui à la plus basse valeur. Nous pouvons donc interpréter cette valeur d'axe principale comme étant un attribut de forme invariant de translation et de rotation. Nous pouvons même calculer l'orientation du moment principal d'inertie associé.

Cette orientation de moment, définie par les axes principaux (valeurs propres) et par les moments principaux (vecteurs propres), sera codée de la manière suivante :

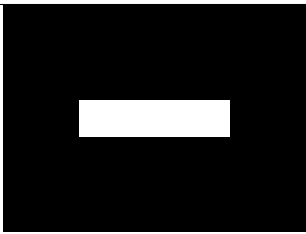
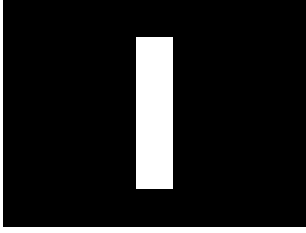
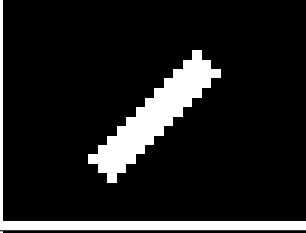
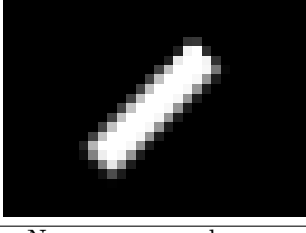
```

1 def getOrientation(valeurs, vecteurs):
2     if valeurs[0] > valeurs[1]:
3         vec = vecteurs[1]
4     else:
5         vec = vecteurs[0]
6
7     return np.arctan2(vec[1], vec[0])

```

2.4 Exemples pratiques 1

Nous allons pouvoir tester notre attribut de forme invariant de translation et de rotation sur quelques exemples :

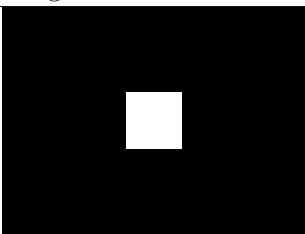
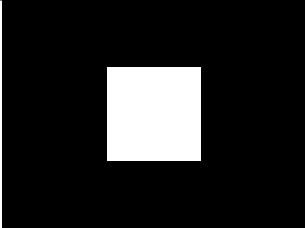
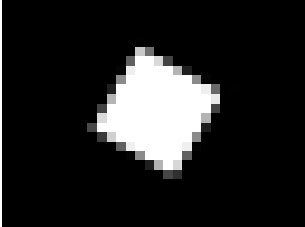
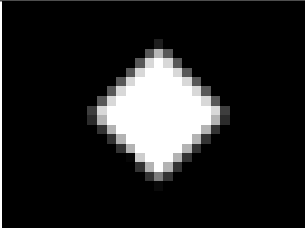
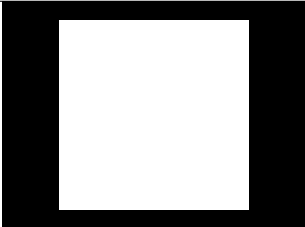
| Image | Valeurs propres | Vecteurs propres | orientation |
|---|------------------------|---|-------------|
|  | 1360 et 80 | [1. 0.] et [0. 1.] | 1.57079 |
|  | 80 et 1360 | [1. 0.] et [0. 1.] | 0.0 |
|  | 1298 et 59 | [0.70710 0.70710] et [-0.70710 0.70710] | 2.35619 |
|  | 99.67303 et 1393.74270 | [-0.70803 0.70617] et [-0.70617 -0.70803] | 2.35750 |

Nous pouvons donc constater que notre attribut de forme est en effet invariant de translation et de rotation. Notamment avec les exemples figure 1 et figure 2, ces rectangles ont des valeurs propres et des vecteurs propres identiques, la différence entre les deux est visible dans l'orientation ou l'un en a une de 0 et l'autre de 1.5.

Par la suite en comparant les figure3 et figure4 (qui sont ressemblantes, la figure4 est la version lissée de la figure3) on remarque que leurs valeurs propres et vecteurs propres sont différentes malgré que ces figures soient ressemblantes et que leurs orientations soient presque identiques.... Cela est normal, en lissant une image, cela affecte ses valeurs de niveaux de gris $I(x,y)$. Les valeurs $I(x,y)$ intervenant dans le calcul du moment cartésien que nous avons présenté au début de ce document, il est normal que les résultats finaux en soient impactés.

... Mais que se passe t-il lorsque les formes n'ont pas la même échelle .. ?

2.5 Exemples pratiques 2

| Image | Valeurs propres | Vecteurs propres | orientation |
|---|------------------------|---|-------------|
|  | 105 et 105 | $[1\ 0]$ et $[0\ 1]$ | 0.0 |
|  | 825 et 825 | $[1\ 0]$ et $[0\ 1]$ | 0.0 |
|  | 842.42025 et 843.28148 | $[-0.95469\ -0.29759]$ et $[0.29759\ -0.95469]$ | -2.83942 |
|  | 841.51713 et 838.53593 | $[0.77710\ -0.62936]$ et $[0.62936\ 0.77710]$ | 0.89005 |
|  | 13300 et 13300 | $[1\ 0]$ et $[0\ 1]$ | 0.0 |

Les vecteurs propres et rotations des 3 carrés de taille différentes sont identiques mais leurs valeurs propres sont très différentes, cela est dû aux changements de tailles de nos carrés, nos attributs de forme ne sont donc pas invariants d'échelle.

Du côté des trois carrés de taille 10, on remarque des valeurs propres relativement proches contrairement à leur vecteurs propres et donc leurs orientations.

Il est préférable, lorsqu'on utilise un attribut de forme en analyse d'image, qu'il s'agisse d'un indice de forme. Il nous manque cependant l'invariance d'échelle...






3 Moments normalisés

Nous allons maintenant essayer d'obtenir un attribut de forme invariant de translation, de rotation et d'échelle. Pour cela, nous utilisons, à la place du moment cartésien centré, le moment cartésien centré normalisé. Ce moment cartésien centré normalisé, qui réutilise le moment cartésien centré et ses données, est défini ainsi :





$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{1+(p+q)/2}}$$

Et est codé ainsi (on réutilise la fonction momentCentre écrite précédemment) :





```
1 def momentCentreNormalise(im, barycentre, p, q):
2     centre = momentCentre(im, barycentre, p, q)
3     zero = momentCentre(im, barycentre, 0, 0)**(1+(p+q)/2)
4     return centre/zero
```

| image | Valeurs propres | Orientation |
|---|-------------------------|---------------------|
|  | [0.08101852 0.08101852] | 0.0 |
|  | [0.0825 0.0825] | 0.0 |
|  | [0.08402441 0.08411031] | -2.8394246574107886 |
|  | [0.0854334 0.08513074] | 0.8900570210875899 |
|  | [0.083125 0.083125] | 0.0 |

Nous constatons que dorénavant les carrés ont tous des valeurs propres d'environ 0.08, peu importe leurs échelles.

| image | Valeurs propres | Orientation |
|---|-------------------------|--------------------|
|  | [0.33203125 0.01953125] | 1.5707963267948966 |
|  | [0.01953125 0.33203125] | 0.0 |
|  | [0.38585018 0.01753864] | 2.356194490192345 |
|  | [0.02395988 0.33503453] | 2.3575080364659993 |

Les rectangles n'ont pas de changement d'échelle et on remarque leurs valeurs propres très proches. Le seul changement ici causé par l'utilisation du moment normalisé est le fait que les valeurs propres sont comprises entre 0 et 1.

| image | Valeurs propres | Orientation |
|---|-------------------------|--------------------|
|  | [0.09484009 0.1007766] | 2.936307360635347 |
|  | [0.09483242 0.10033918] | 2.655956434221538 |
|  | [0.10050438 0.09515709] | 2.2004992649074597 |
|  | [0.09336958 0.10154882] | 2.8924373100717844 |

Ces triangles ne subissent pas de changements d'échelles mais juste des rotations, on remarque leurs valeurs propres très proches.

En vue des résultats obtenues et de la nature invariante en translation, rotation et échelle de cet attribut de forme, on peut

4 Moments invariants

Dans cette partie nous parlons des 5 premiers moments invariants de Hu et de leur utilisation en tant qu'indice de forme. Voici leurs définitions :

$$\Phi_1 = \eta_{20} + \eta_{02} \quad (1)$$

$$\Phi_2 = (\eta_{20} - \eta_{02})^2 + (2\eta_{11})^2 \quad (2)$$

$$\Phi_3 = (\eta_{30} - 3\eta_{12})^2 + (\eta_{03} - 3\eta_{21})^2 \quad (3)$$

$$\Phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{03} + \eta_{21})^2 \quad (4)$$

$$\Phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{03} + \eta_{21})^2] \quad (5)$$

$$+ (\eta_{03} - 3\eta_{21})(\eta_{03} + \eta_{21})[(\eta_{03} + \eta_{21})^2 - 3(\eta_{30} + \eta_{12})^2] \quad (6)$$

$$(7)$$

Et voici la manière dont on l'a codé :





```

1 def hu5(im, barycentre, affiche=False):
2     n20 = momentCentreNormalise(im, barycentre, 2, 0)
3     n02 = momentCentreNormalise(im, barycentre, 0, 2)
4     n11 = momentCentreNormalise(im, barycentre, 1, 1)
5     n30 = momentCentreNormalise(im, barycentre, 3, 0)
6     n12 = momentCentreNormalise(im, barycentre, 1, 2)
7     n03 = momentCentreNormalise(im, barycentre, 0, 3)
8     n21 = momentCentreNormalise(im, barycentre, 2, 1)
9     n31 = momentCentreNormalise(im, barycentre, 3, 1)
10    res = {}
11    res[1] = n20 + n02
12    res[2] = (n20-n02)**2 + (2*n11)**2
13    res[3] = (n30-3*n12)**2 + (n03-3*n21)**2
14    res[4] = (n30+n12)**2 + (n03+n21)**2
15    res[5] = (n30 - 3*n12) * (n30 + n12) * ( (n30 + n12)**2 - 3*(n03 + n21)**2 )
16    res[5] += (n03 - 3*n21) * (n03 + n21) * ( (n03 + n31)**2 - 3*(n30 + n12)**2 )
17    if affiche:
18        [print("Hu_ n " +str(i)+" : "+res[i]) for i in range(1,6)]
19        print("\n")
20    return res






```

Pour obtenir le ieme moment de Hu dans le résultat de la fonction, il suffit de récupérer l'élément d'indice i.





Voici les figures et leurs moments de Hu associés :

| image | hu-1 | hu-2 | hu-3 | hu-4 | hu-5 |
|---|---------|---------|---------|---------|----------|
|  | 0.35156 | 0.09766 | 0.00000 | 0.00000 | 0.00000 |
|  | 0.35156 | 0.09766 | 0.00000 | 0.00000 | 0.00000 |
|  | 0.40339 | 0.13565 | 0.00000 | 0.00000 | 0.00000 |
|  | 0.35899 | 0.09677 | 0.00000 | 0.00000 | -0.00000 |

Ici, le rectangle est pivoté et conserve quasiment les mêmes valeurs de moments invariants. Il y a un changement de valeur lorsque le rectangle est diagonal, mais il est moindre et il n'est plus présent une fois le rectangle lissé.

| image | hu-1 | hu-2 | hu-3 | hu-4 | hu-5 |
|---|---------|---------|---------|---------|----------|
|  | 0.16204 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
|  | 0.16500 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
|  | 0.16813 | 0.00000 | 0.00000 | 0.00000 | -0.00000 |
|  | 0.17056 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
|  | 0.16625 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |

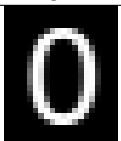
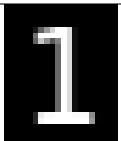


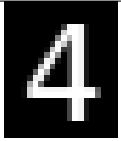

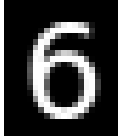



Ici, le carré est pivoté et il subit des changements d'échelles. Malgré cela, les valeurs de moments invariants sont très proches.

| image | hu-1 | hu-2 | hu-3 | hu-4 | hu-5 |
|---|---------|---------|---------|---------|----------|
|  | 0.19562 | 0.00004 | 0.00455 | 0.00000 | 0.00000 |
|  | 0.19517 | 0.00003 | 0.00457 | 0.00000 | 0.00000 |
|  | 0.19566 | 0.00003 | 0.00461 | 0.00000 | -0.00000 |
|  | 0.19492 | 0.00007 | 0.00440 | 0.00000 | 0.00000 |

Enfin, les valeurs de moments invariants du triangle pivoté sont elles aussi très proches les unes des autres.

En vue des résultats obtenues, les moments invariants de Hu semblent invariants de rotation, de translation et d'échelle. Les moments invariants de Hu seraient donc un indice de forme utilisable dans la reconnaissance de forme.

Ces moments pourraient donc être utiles pour différencier différentes formes ? Voici un exemple sur l’affichage de chiffres :

| image | hu-1 | hu-2 | hu-3 | hu-4 | hu-5 |
|---|---------|---------|---------|---------|----------|
|  | 0.45615 | 0.01753 | 0.00000 | 0.00000 | -0.00000 |
|  | 0.61660 | 0.24605 | 0.01458 | 0.00289 | 0.00001 |
|  | 0.59671 | 0.13557 | 0.00745 | 0.00241 | 0.00000 |
|  | 0.52738 | 0.08907 | 0.01622 | 0.00459 | -0.00004 |
|  | 0.34661 | 0.01676 | 0.01509 | 0.00023 | 0.00000 |
|  | 0.51351 | 0.07646 | 0.00520 | 0.00196 | 0.00001 |
|  | 0.38381 | 0.01670 | 0.00173 | 0.00032 | -0.00000 |
|  | 0.59516 | 0.16054 | 0.10783 | 0.02111 | 0.00094 |
|  | 0.37628 | 0.01920 | 0.00014 | 0.00001 | 0.00000 |
|  | 0.38798 | 0.01808 | 0.00267 | 0.00053 | -0.00000 |

Plusieurs constats de ces résultats sont à appuyer :

- Le 6 et le 9 ont des valeurs casi similaires (le 6 devient un 9 après une rotation de 180°)
- Le 5 et le 3 ont des valeurs très proches
- Le 8 a des valeurs proches du 6 et du 9

Avec ces constats, on peut affirmer qu’il n’est pas acceptable d’utiliser le moment invariant de Hu afin de reconnaître les différents chiffres, il pourrait les confondre. En conclusion, les indices de formes sont très utiles mais

le contexte de la reconnaissance de forme voulu est à prendre en compte afin de savoir s'il est pertinent ou non de les utiliser.