Introduction à l'imagerie numérique

3I022-fev2018

Détection d'objets

Licence d'informatique

Février 2018

Détection des objets

- Une tâche “classique” du traitement des images.
- Reconnaissance de forme.
- Objets : une forme particulière, par exemple :
  - des points,
  - des segments de droites,
  - des contours,
  - des formes plus complexes.

Détection d'objets simples

Plan du cours

1. Points isolés
2. Détecteur de coins
3. Segment de droites
4. Détecteur de droites (transformé de Hough)
5. Généralisation à des formes paramétrées quelconque

Avertissement

La plupart des exemples donnés dans ce cours sont reproductibles à partir des images disponibles dans le dépôt de l'UE et de la séquence de commandes Shell/Inrimage indiquée ou du matériel donné et développé en TME.

Détection de points

- Attention : problème différent de la détection de contours,
- il s'agit de détecter des points isolés ou dans une configuration particulière,
- les détecteurs de Sobel, Marr, . . . ne sont donc pas adaptés.
- Considérons d'abord le cas du point isolé ...

Points (contours) isolés

- Un filtre (un noyau de convolution) qui répond plus fortement sur un point de contour **isolé** que sur un groupement de points de contour :

-1	-1	-1
-1	8	-1
-1	-1	-1

- Pourquoi les filtres de type Sobel :

-1	0	1
-2	0	2
-1	0	1

ne conviennent-ils pas ?

Détection de points

suite

- On va vérifier que la réponse est plus forte sur un point isolé que sur les points d'une droite, d'un croisement, d'un bord de région.

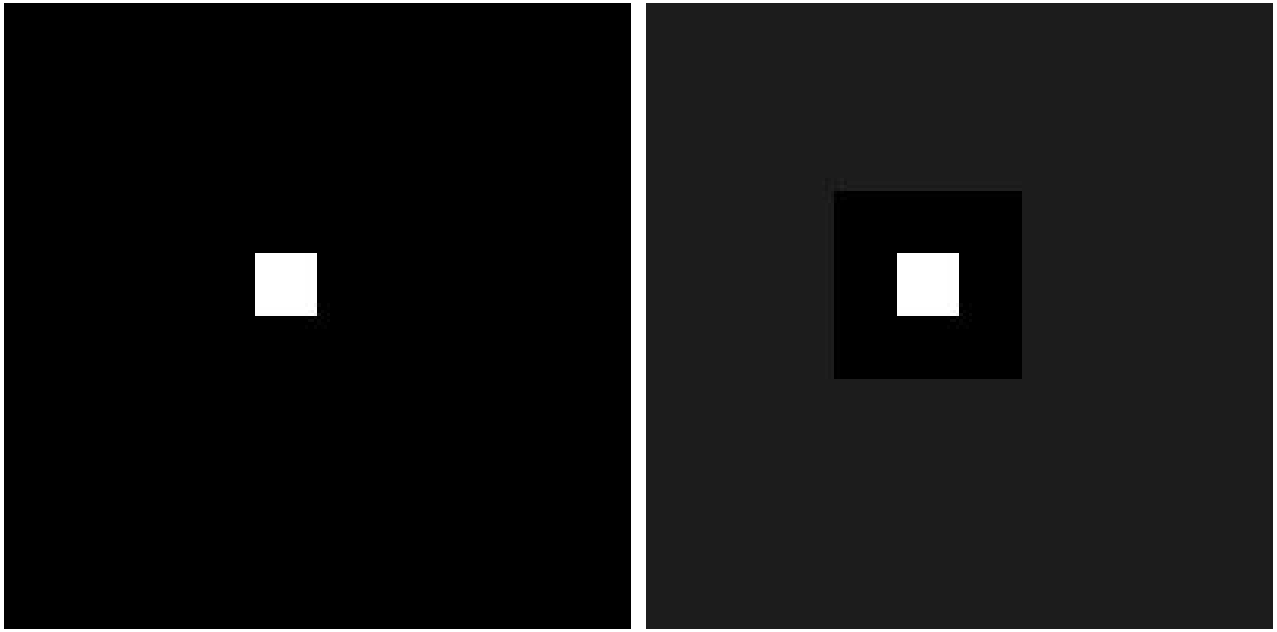
```
raz -x 50 -y 50 point
printf '##!edit(on,C 255,p 25 25)\n##!EXIT\n' > point.xv
xvis -ed point point -xsh point.xv
```

- Calculons la réponse du filtre au point $(25, 25)$:
  $0 \times -1 + \cdots + 0 \times -1 + 8 \times 1 = 8$
- au voisinage de $(25, 25)$: tous les termes sont nuls sauf un :
  $-1 \times 1 = -1$
- ailleurs : tout est nul.

```
echo -1 -1 -1 -1 8 -1 -1 -1 -1 | cim -x 3 -y 3 -r > detp
conv -dir point detp pointd
ical pointd -1.000000 0.000000 8.000000
```

Détection de points

suite



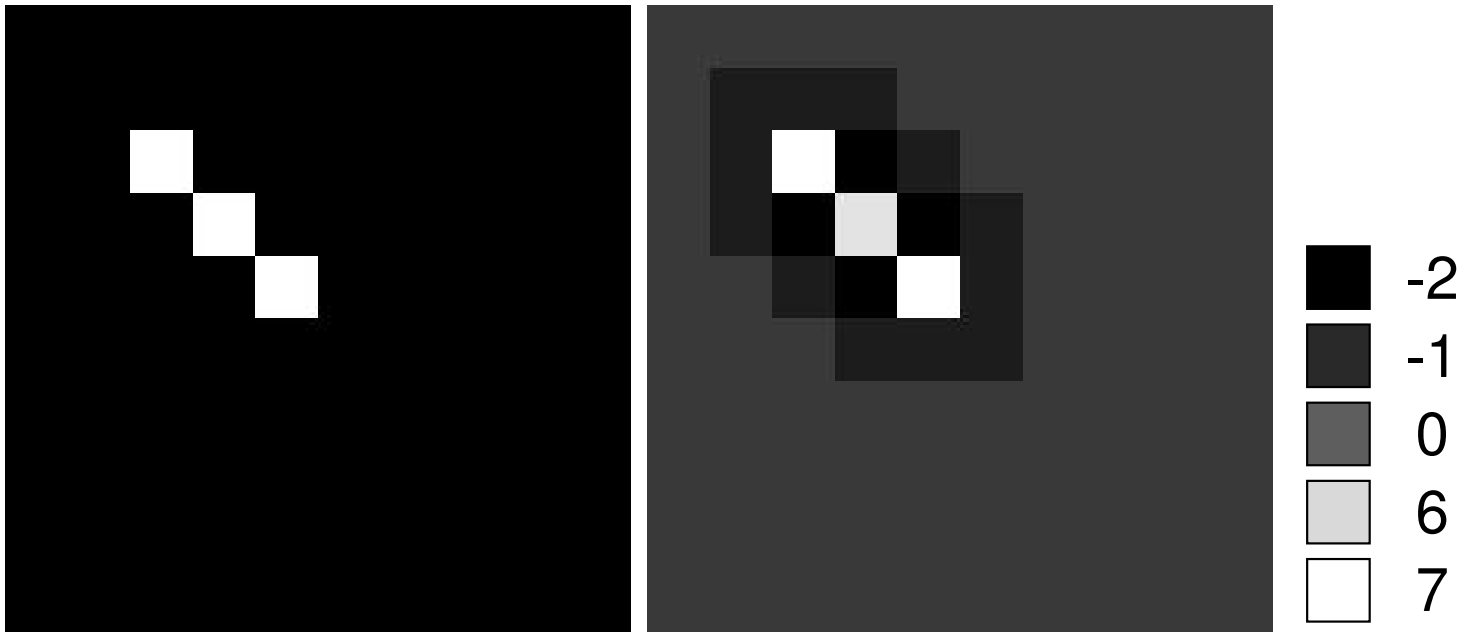
(a) `xvis -p -20 point` (b) `norma pointd | xvis -p -20`

Détection de points

suite

- Réponse sur un segment de droite :

```
raz -x 10 -y 10 >line
printf '##!edit(on,C 255,p 3 3,l 5 5)\n##!EXIT\n' > line.xv
xvis -ed line line -xsh line.xv
conv -dir line detp  | ical
     -2.000000       0.000000       7.000000
```



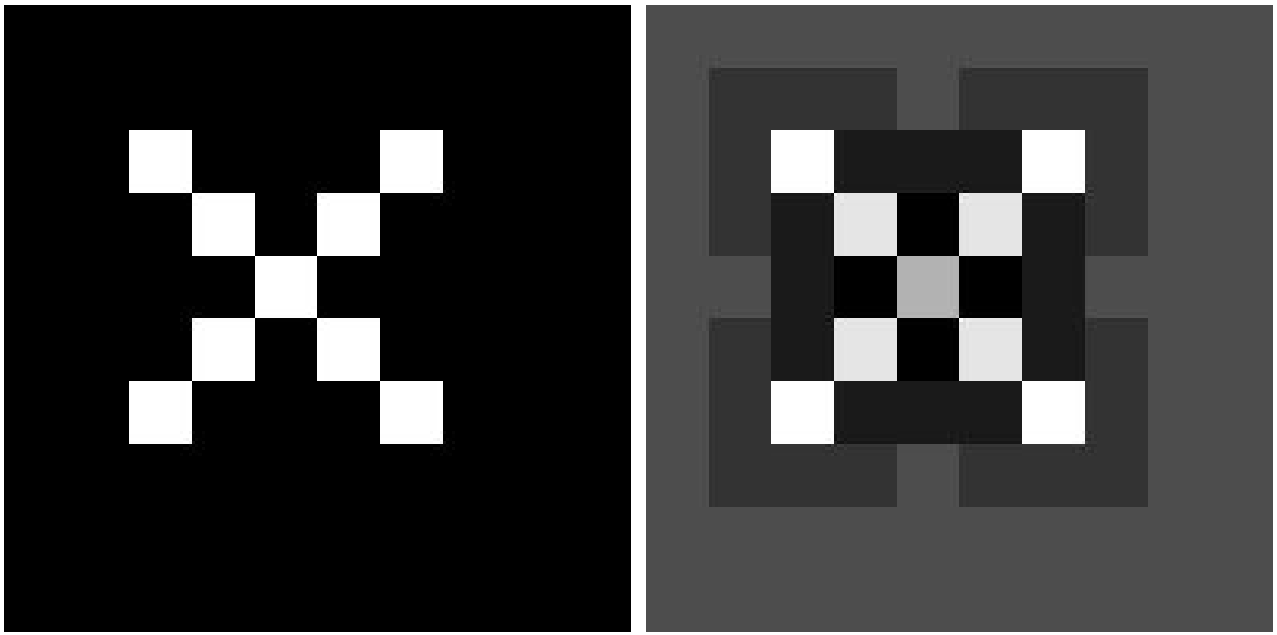
(c) (d) (e)

Détection de points

suite

- Réponse sur une intersection de droites :

```
raz -x 10 -y 10 cross
printf '##!edit(on,C 255,p 3 3,l 7 7,p 3 7,l 7 3)\n##!EXIT\n' > cross.xv
xvis -ed cross cross -xsh cross.xv
conv -dir cross detp | ical
     -3.000000      0.000000      7.000000
```



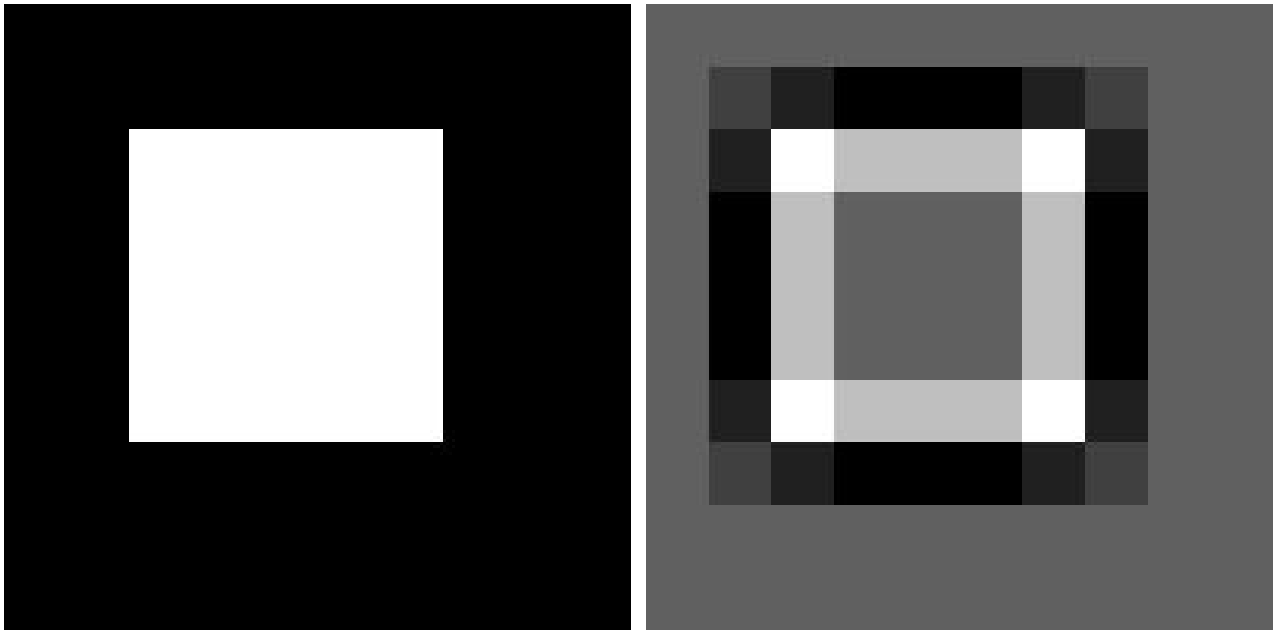
(f) (g)

Détection de points

suite

- Réponse sur une région :

```
raz -x 10 -y 10 >region
raz -x 5 -y 5 | logic -inv | melg - -ixo 3 -iyo 3 region
conv -dir region detp | ical
     -3.000000      0.000000      5.000000
```



(h) (i)

Détecteur de coins

- Comment détecter un coin ?

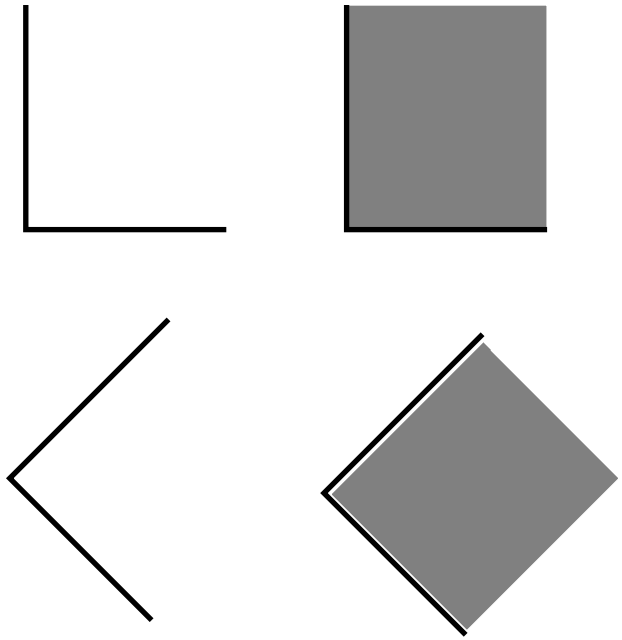


FIGURE – Des coins dans une image ...

- Comment gérer les problèmes d'orientation ?
- Moravec (1980) propose un critère simple basé sur l'*auto-similarité*.

Détecteur de Moravec

Mesure de l'auto-similarité

- Pour détecter la présence d'un coin au pixel p :
  - Considérer une fenêtre W centrée en p (voisinage de p).
  - Vérifier que "l'aspect" dans la fenêtre ne change pas lorsqu'on déplace celle-ci autour de p.

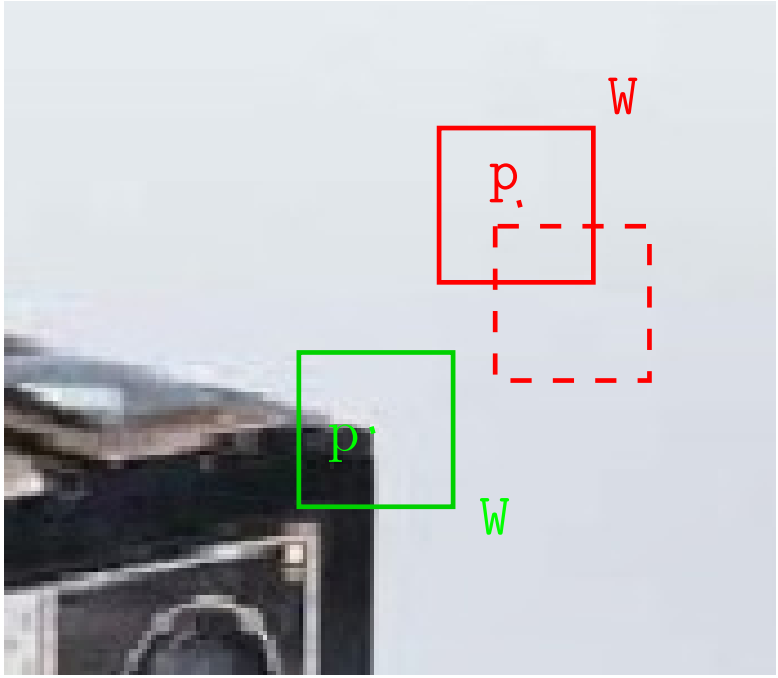


FIGURE – fenêtre similaire dans un voisinage ?

Détecteur de Moravec

Auto-similarité

- Supposons que l'on sache mesurer "l'aspect".
- Pour vérifier que cet algorithme est pertinent, étudions 5 configurations typiques (à vérifier sur la figure page 12) :
  1. région uniforme : quelle que soit la direction où l'on déplace W, l'aspect ne change pas ;
  2. bord d'une région (figure) : l'aspect de la fenêtre ne change pas le long du bord de la région ;
  3. contours : même cas que précédent ;
  4. coin d'une région : où que l'on déplace la fenêtre, l'aspect est changé ;
  5. point isolé : même cas que précédent.

Détecteur de Moravec

Mesurer la similarité

- un critère de similarité : doit être robuste au bruit, idéalement la corrélation mais cela implique des calculs lourds.
- Moravec propose un critère plus simple mais suffisant : pour une image I, mesurer la moyenne de la différence quadratique entre la fenêtre W et sa translatée d'un vecteur $t = (t_x, t_y)$:

$$E(W, t) = \sum_{p \in W} (I(p) - I(p + t))^2 \quad (1)$$

ou encore :

$$E(W, t_x, t_y) = \sum_{(i,j) \in W} (I(i, j) - I(i + t_x, j + t_y))^2$$

ou encore :

$$E(W, t_x, t_y) = \sum_{(i,j)} w(x, y)(I(i, j) - I(i + t_x, j + t_y))^2$$

en notant $w(i, j) = 1$ si $(i, j) \in W$ et 0 sinon.

Détecteur de Moravec

Algorithme

```
algo Moravec(p:pixel);
   min := 0;
   Wp := fenetre centree sur p
   pour chaque deplacement t telque Wp(t+p)=1 faire
        e := E(Wp,t);
        si e < min alors min := e;
   fin pour
   si min > SEUIL alors
      marquer p comme coin;
   fin si
fin algo // Moravec
```

- `E` est la fonction définie par l'équation (1).
- `SEUIL` et **`Wp`** sont les deux paramètres de l'algorithme.

Détecteur de Moravec

Conclusion

- Il faut choisir le seuil (empirique).
- Le calcul de E reste lent : en particulier on doit évaluer $I(p+t)$ plusieurs fois.
- le filtre n'est pas **isotropique** (c.a.d. identique quelle que soit la direction) :
  - $I(i,j) - I(i-1,j)$ mesure la différence entre deux points distants d'un pixel,
  - $I(i,j) - I(i-1,j-1)$ mesure la différence entre deux points distants de $\sqrt{2}$ pixel.

  $\Rightarrow$ on n'aura donc pas la même réponse entre une image et sa rotation d'angle $\frac{\pi}{4}$.

Détecteur de Harris (1988)

- Comment régler les deux limitations du détecteur de Moravec (lenteur et non isotropie) ?
- Considérer E et utiliser un développement limité d'ordre 1 de I au voisinage de (i, j)
- Rappel : D.L. ordre 1 :

$$I(i + t_x, j + t_y) = I(i, j) + \frac{\partial I}{\partial x}(i, j)t_x + \frac{\partial I}{\partial y}(i, j)t_y + \mathcal{O}(t_x^2, t_y^2)$$

- Le critère E devient :

$$E(W, t_x, t_y) \sim \sum_{(i,j)} w(i, j) \left(\frac{\partial I}{\partial x}(i, j)t_x + \frac{\partial I}{\partial y}(i, j)t_y \right)^2$$

Détecteur de Harris

suite

- Avantages :
  1. rapidité accrue : on calcule une seule fois les gradients de l'image $(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$.
  2. $E(W, t_x, t_y)$ est évaluable pour tout (t_x, t_y) réel : on peut construire facilement un filtre isotropique.
- Inconvénient :
  - c'est approximatif car on a négligé $\mathcal{O}$: ne fonctionne que pour des valeurs petites pour t_x et t_y.
- Choix pour la fenêtre :
  - on peut garder $w(i,j) = 1$ lorsque $(i,j) \in W$
  - Harris propose une fenêtre gaussienne :

$$w(i,j) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i - x_W)^2 + (j - y_W)^2}{\sigma^2}\right)$$

(x_W, y_W) désigne le centre de la fenêtre W.

Détecteur de Harris

Fenêtre gaussienne

- Le détecteur d'Harris, s'écrit alors :

$$E(W, t_x, t_y) = \sum_{i,j} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i - x_W)^2 + (j - y_W)^2}{\sigma^2}\right) e_{t_x,t_y}(i, j)$$

avec :

$$e_{t_x,t_y}(i, j) = \left(\frac{\partial I(i, j)}{\partial x} t_x + \frac{\partial I(i, j)}{\partial y} t_y\right)^2$$

- En pratique, on fixe la taille de la fenêtre à $\sim 3\sigma^2$ car au delà de cette taille les valeurs de la gaussienne sont très proches de zéro.
- On a donc $E(x_W, y_W, t_x, t_y) = E(W, t_x, t_y)$ la réponse du filtre de Harris au point x_w, y_w pour une direction (t_x, t_y).
- E est donc une convolution discrète gaussienne :

$$E(x_W, y_W, t_x, t_y) = G_\sigma \star e_{t_x,t_y}(x_w, y_w)$$

Détecteur de Harris

Interprétation géométrique du critère de similarité

- Notons $I_x = \frac{\partial I(i,j)}{\partial x}$ et $I_y = \frac{\partial I(i,j)}{\partial y}$
- On a $t_x I_x + t_y I_y = (t_x, t_y) \begin{pmatrix} I_x \\ I_y \end{pmatrix}$ (produit scalaire)
- E peut se réécrire :

$$\begin{aligned} E(W, t_x, t_y) &= \sum_{i,j} w(i,j) \left((t_x, t_y) \begin{pmatrix} I_x \\ I_y \end{pmatrix} \right)^2 \\ &= \sum_{i,j} w(i,j) (t_x, t_y) \begin{pmatrix} I_x \\ I_y \end{pmatrix} \left((t_x, t_y) \begin{pmatrix} I_x \\ I_y \end{pmatrix} \right)^T \\ &= (t_x, t_y) \left(\sum_{i,j} w(i,j) \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix} \right) \begin{pmatrix} t_x \\ t_y \end{pmatrix} \\ &= (t_x, t_y) A \begin{pmatrix} t_x \\ t_y \end{pmatrix} \end{aligned}$$

Interprétation géométrique du critère de similarité

- Considérons une fenêtre W réduite à 1 point, dans ce cas :

$$A = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}$$

- Cette matrice a une valeur propre nulle $(\det(\mathrm{A}) = 0)$
- Le vecteur propre est le gradient de I : $\begin{pmatrix} I_x & I_y \end{pmatrix}^T$, A identifie la direction du gradient.
- Sur une fenêtre W suffisamment grande, on peut observer (selon les configurations) différentes orientations du gradient.
- En moyennant sur une fenêtre W, A identifiera les deux directions principales du gradient.

Détecteur de Harris

Interprétation géométrique du critère de similarité

- La matrice A est donc :

$$A = \begin{pmatrix} \sum_{i,j} w(i,j) I_x^2 & \sum_{x,y} w(i,j) I_x I_y \\ \sum_{i,j} w(i,j) I_x I_y & \sum_{x,y} w(i,j) I_y^2 \end{pmatrix}$$

- Les vecteurs propres de A identifient les deux directions privilégiées du gradient dans un voisinage W.
- Les valeurs propres de A donnent les composantes du gradient sur la base des vect. prop.

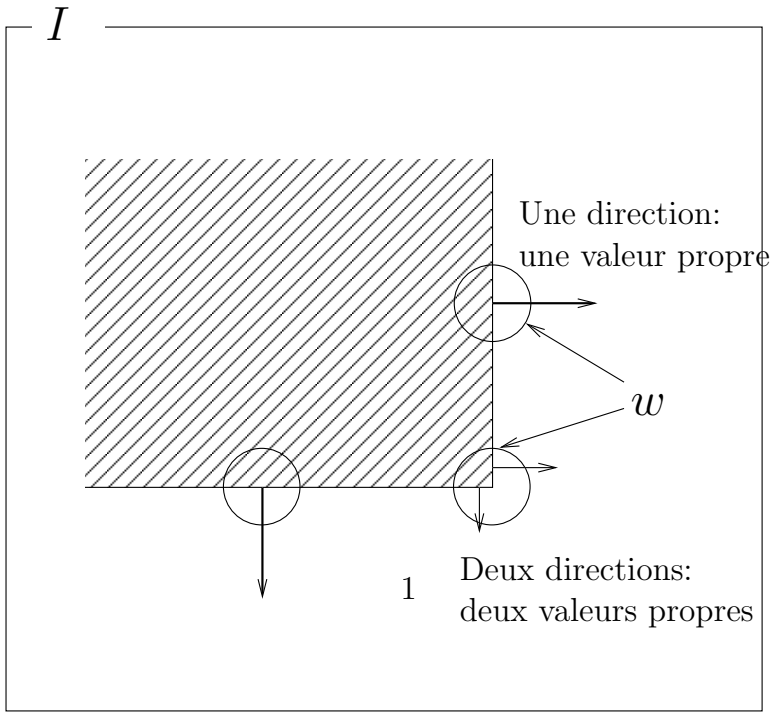


FIGURE – Orientation du gradient.

Détecteur de Harris

Interprétation géométrique du critère de similarité

- L'orientation du gradient est donnée par les deux valeurs propres de la matrice A (λ_1 et λ_2).
- Ainsi, on peut caractériser localement la forme du contour à partir des valeurs propres de A :
  1. si les deux v.p. sont proches de 0 : gradient nul, dans une région d'aspect uniforme ;
  2. si l'une des v.p. est proche de 0 (et l'autre positive) : on a un bord (car une seule direction est privilégiée) ;
  3. si les deux v.p. sont positives : on a un coin (car deux directions sont privilégiées).
- Ceci est vrai quelle que soit l'orientation des gradients (invariance par rotation).

Critère de Harris

- Plutôt que d'analyser E et de rechercher des maxima locaux avec des seuils difficiles à déterminer, il est plus efficace d'analyser les valeurs propres de A.
- Harris propose le critère suivant :

$$R = \det(A) - \kappa \, \mathrm{trace}(A)^2$$

où κ est un paramètre strictement positif.

- En effet, il faut savoir que :
  1. Le déterminant d'une matrice carrée est égal au produit de ses valeurs propres,
  2. La trace d'une matrice est égale à la somme de ses valeurs propres.
- Ainsi : $R = \lambda_1 \lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$

Critère de Harris

- Réponse de R selon les trois configurations type ?
  1. région (soit $\lambda_i = 0$, $i = 1, 2$) : R est proche de zéro,
  2. bord (soit une v.p. proche de zéro et l'autre positive) : $R < 0$,
  3. coin (soit $\lambda_i > 0$, $i = 1, 2$) : quelle condition sur κ pour que $R > 0$?

$$\lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2 > 0$$

$$\kappa < \frac{\lambda_1\lambda_2}{\lambda_1^2 + \lambda_2^2 + 2\lambda_1\lambda_2} \leq \frac{1}{4}$$

les v.p. sont positives, il faut donc prendre κ assez petit, en pratique entre 0,04 et 0,15.

- En calculant R en tout point, on obtient alors une image : la réponse du filtre de Harris : les points à valeur positive sont des coins.
- À cause du bruit, on a beaucoup de 'coins' qui n'en sont pas :
  - on garde que les réponses les plus fortes, i.e. les valeurs au dessus d'un seuil t
  - on ne garde que les maxima locaux (car un coin est toujours isolé)

Détecteur de Harris

Calcul de R

- Rappel :

$$\begin{aligned} R &= \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2) \\ &= \det(A) - \kappa\,\mathrm{trace}(A) \end{aligned}$$

- Il est plus simple de calculer la trace et le déterminant d'une matrice 2×2 que ses valeurs propres :

$$\begin{aligned} A &= \begin{pmatrix} A_{11} & A_{12} \\ A_{12} & A_{22} \end{pmatrix} \\ \mathrm{trace}(A) &= A_{11} + A_{22} \\ \det(A) &= A_{11}A_{22} - A_{12}^2 \end{aligned}$$

Détecteur de Harris

Algorithme

```
algo Harris(I:image);
  Ix := convol(I,sobelx);
  Iy := convol(I,sobely);
  gauss := noyau_gaussien(sigma);
  A11 := convol(Ix^2,gauss);
  A22 := convol(Iy^2,gauss);
  A12 := convol(Ix*Iy,gauss);
  pour tout pixel p de R faire
    R(p) := A11(p)*A22(p)-A12(p)*A12(p) -
            kappa * (A11(p)+A22(p))^2;
  fin pour
  pour tout pixel p de R faire
    si p est maximum local et R(p)>t alors
       marquer p comme coin;
    fin si
fin algo
```

Détecteur de Harris

Démonstration

- L'algorithme possède 3 paramètres :
  - la variance σ,
  - le seuil t,
  - le paramètre du critère de Harris κ.
- Faisons varier ces paramètres et étudions l'impact sur les résultats...

Détecteur de Harris

Le paramètre σ

FIGURE – $\sigma = 0.5$

Détecteur de Harris

Le paramètre σ

FIGURE – $\sigma = 1$

Détecteur de Harris

Le paramètre σ

FIGURE – $\sigma = 2$

Détecteur de Harris

Le paramètre σ

FIGURE – $\sigma = 4$

- Élimination des hautes fréquences avec σ croissant.

Détecteur de Harris

Le paramètre κ

FIGURE – $\kappa = 0.4$ $(\sigma = 2)$

Détecteur de Harris

Le paramètre κ

FIGURE – $\kappa = 0.2$ $(\sigma = 2)$

Détecteur de Harris

Le paramètre κ

FIGURE – $\kappa = 0.02$ ($\sigma = 2$)

Détecteur de Harris

Le paramètre κ

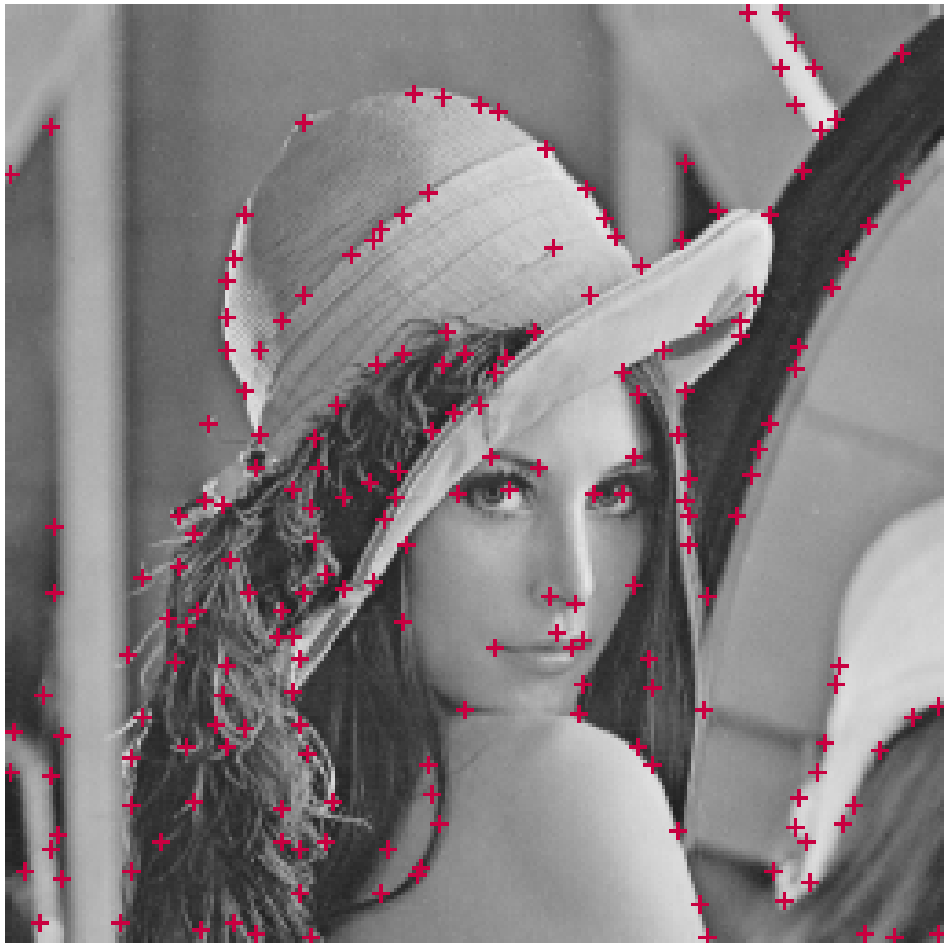


FIGURE – $\kappa = 0.002$ ($\sigma = 2$)

Détecteur de Harris

Le paramètre κ

- Une valeur trop grande ne détecte rien.
- Une valeur trop proche de zéro, le détecteur de Harris se comporte comme un détecteur de contours.
- En pratique, on fixe ce paramètre à une valeur entre 0,04 et 0,15 et l'on n'y touche guère par la suite.

Détecteur de Harris

Le paramètre t

FIGURE – t =10% des maxima locaux ($\kappa = 0.02$, $\sigma = 2$)

Détecteur de Harris

Le paramètre t

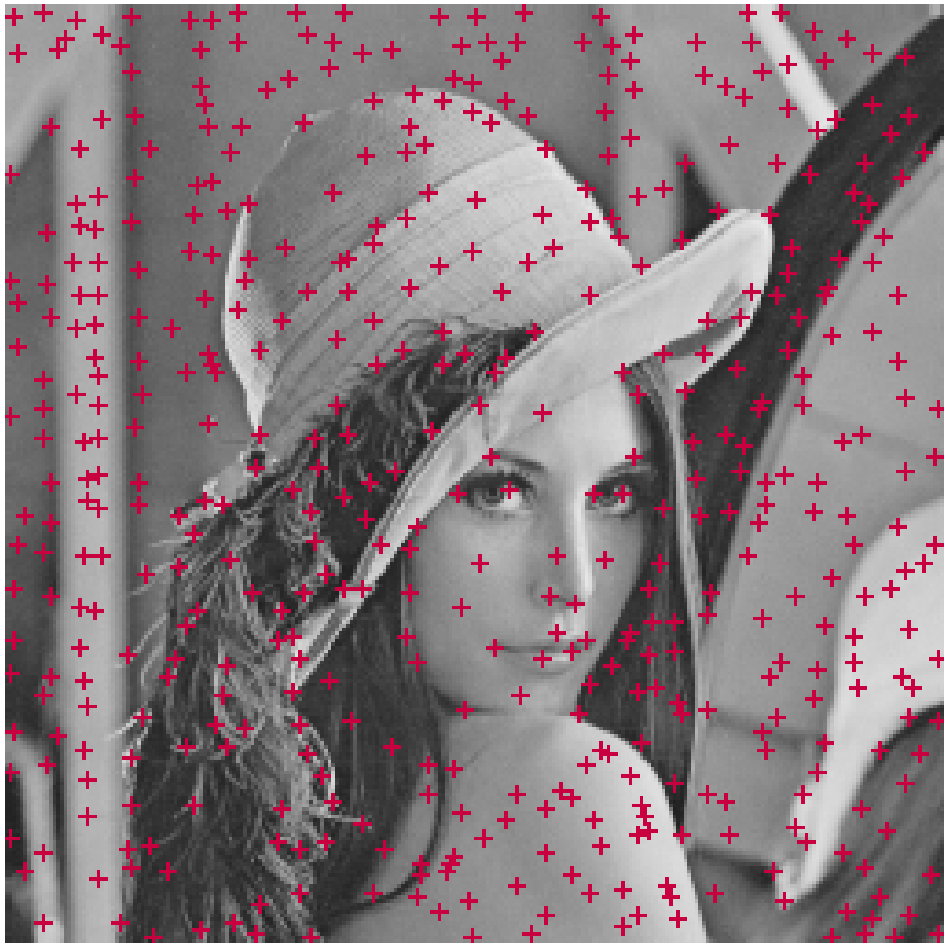


FIGURE – t =100% des maxima locaux ($\kappa = 0.02$, $\sigma = 2$)

Détecteur de Harris

Le paramètre t

- Un seuil trop bas, et on récupère des maxima locaux qui ne sont pas des coins : du bruit, des points isolés.
- On peut compenser en remontant σ ...

Détecteur de Harris

Le paramètre t

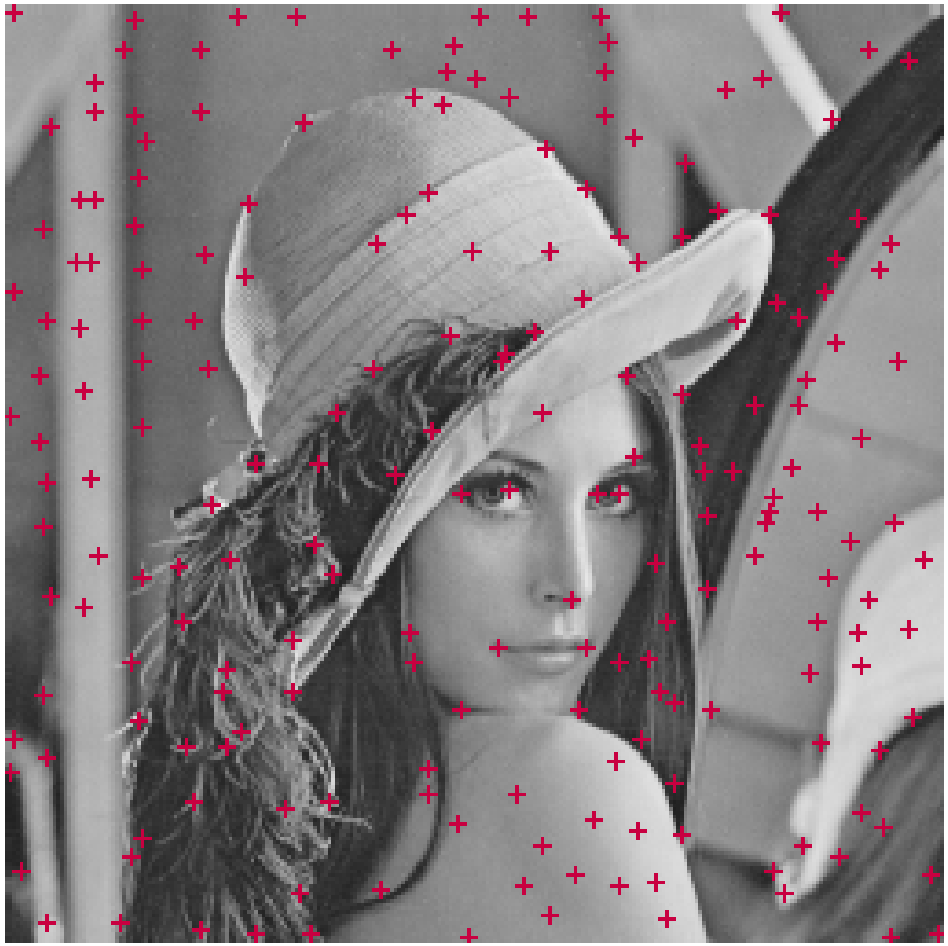


FIGURE – $\sigma = 3$ (t =100% des maxima locaux, $\kappa = 0.02$)

Détecteur de Harris

Le paramètre t

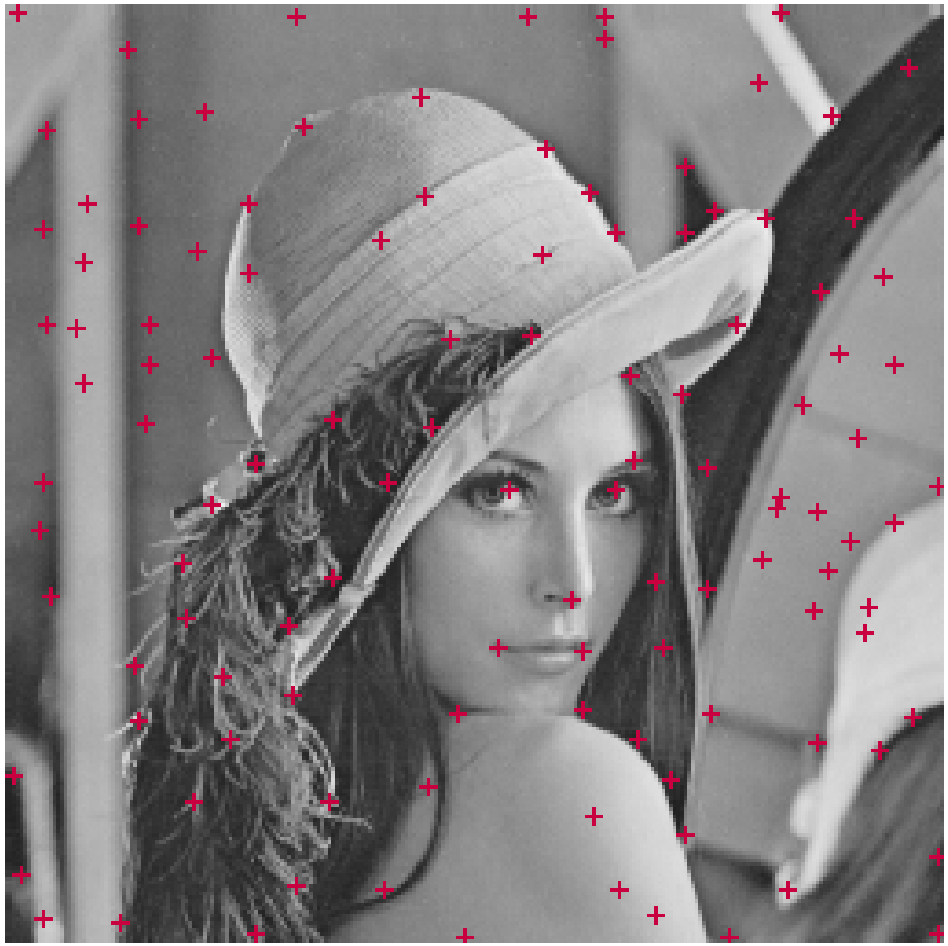


FIGURE – $\sigma = 4$ (t =100% des maxima locaux, $\kappa = 0.02$)

Détection de segments de droite

Filtrage spatial

- On peut détecter des portions de segment de droite par l'utilisation de filtres spatiaux avec une approche similaire à celle employée pour la détection de points isolés.
- Par exemple, le filtre :

-1	-1	-1
2	2	2
-1	-1	-1

aura une réponse forte sur les segments de droite horizontaux.

Détection de lignes

Exemple sur données synthétiques

- Création des données : on aura besoin de dessiner des droites dans les images avec le script suivant :

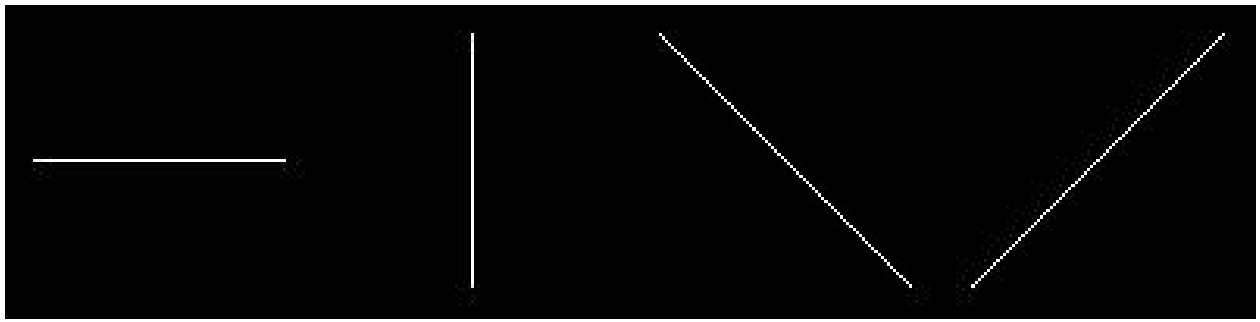
```
cat > drawline <<EOF
#! /bin/bash
# drawline file dimx dimy x0 y0 x1 y1
#   creer une image file de taille dimx par dimy et y
#   dessine le segment [(x0,y0),(x1,y1)]
par \$1 || raz \$1 -x \$2 -y \$3
echo "##!edit(on,C␣255,p␣\$4␣\$5,l␣\$6␣\$7)
##!EXIT" > /tmp/line
xvis -ed \$1 \$1 -xsh /tmp/line
rm -f /tmp/line
EOF
chmod +x drawline
```

Détection de lignes

Exemple sur données synthétiques

- On peut maintenant créer les images tests :

```
./drawline line1 100 100 10 50 90 50
./drawline line2 100 100 50 10 50 90
./drawline line3 100 100 10 10 90 90
./drawline line4 100 100 90 10 10 90
xvis -grp 4 -nu -hz 4 line?
```

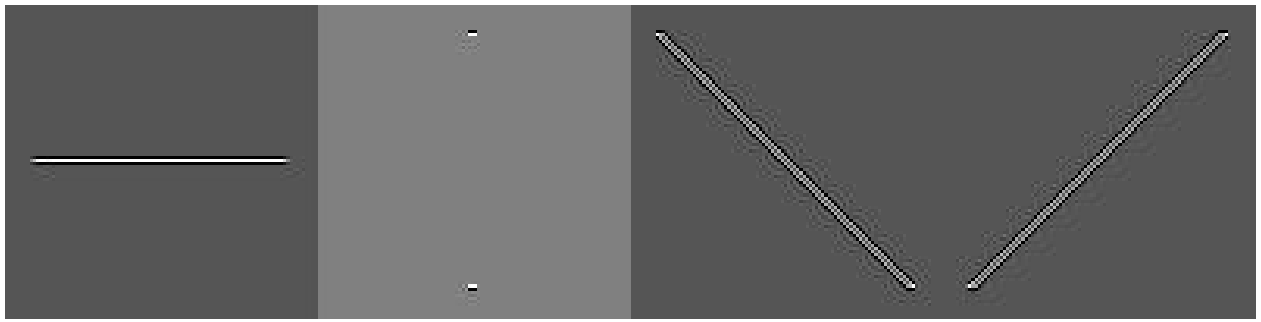


Détection de lignes

Exemple sur données synthétiques

- Convolution avec le filtre détecteur de lignes :

```
echo -1 -1 -1 2 2 2 -1 -1 -1 | cim -x 3 -y 3 -r > detl1
for i in 1 2 3 4 ; do
  conv -dir line$i detl1 > line$i.detl1
done
ical line?.detl1
line1.detl:    -3.000000      0.000000      6.000000
line2.detl1:   -1.000000      0.000000      1.000000
line3.detl1:   -1.000000      0.000000      2.000000
line4.detl1:   -1.000000      0.000000      2.000000
```

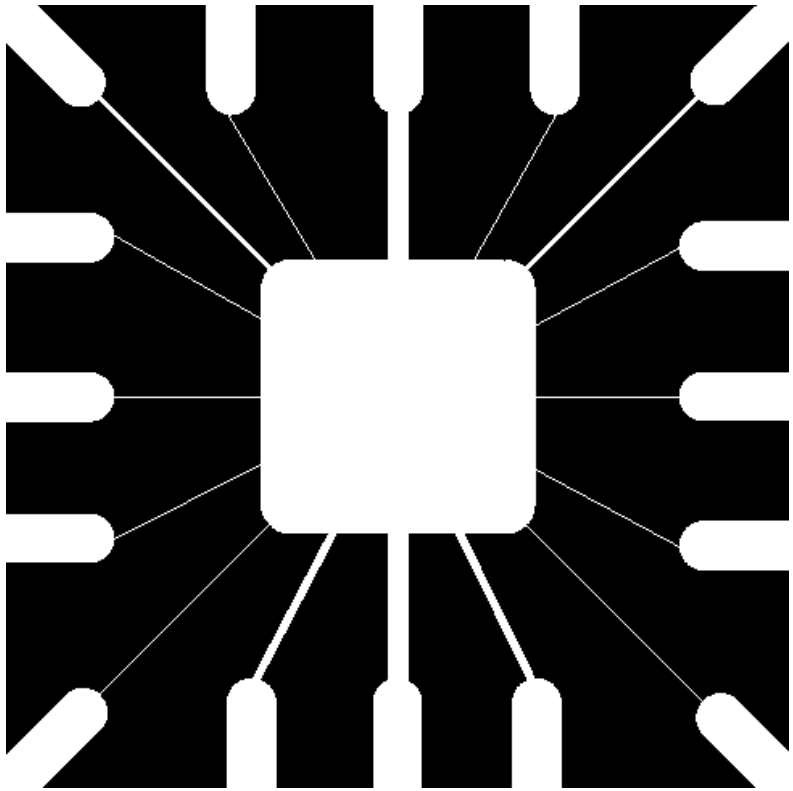


Détection de lignes

Exemple sur données synthétiques

- On voit que la réponse idéale est autour de 6 pour cette image.
- Il faut donc seuiller judicieusement pour garder les droites avec la bonne orientation.
- Exemple avec cette image :

```
xvis -nu puce.inr
```



Détection de lignes

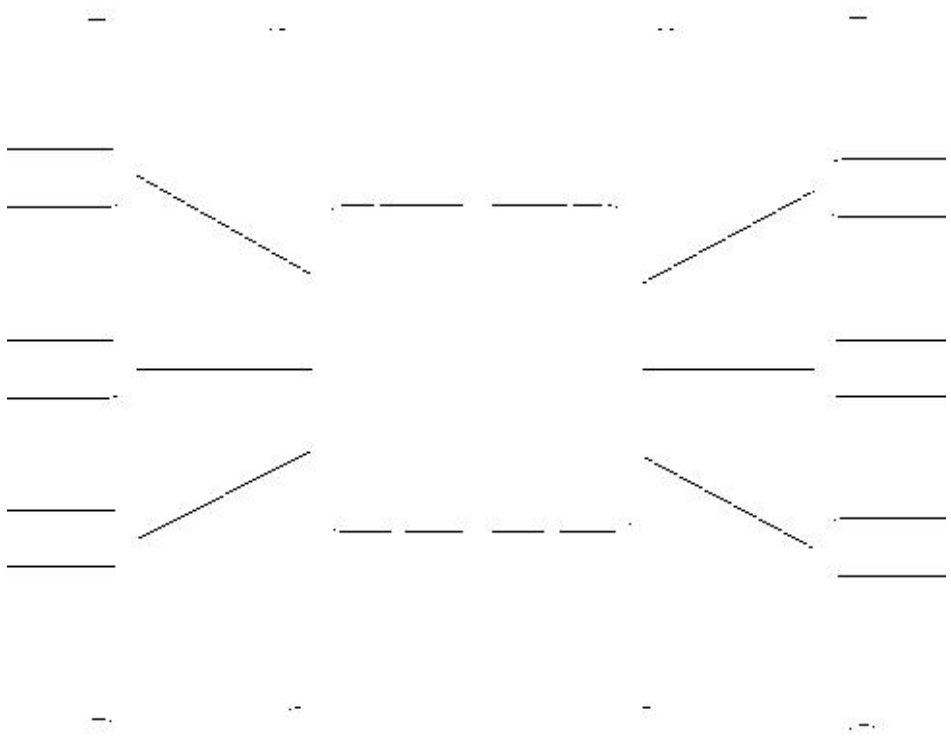
Exemple sur données synthétiques

```
conv -dir puce.inr detl1 | ical
     -4.000000       0.000000       6.000000
conv -dir puce.inr detl1 | mb -n 5.9 | xvis -nu
```

Détection de lignes

Exemple sur données synthétiques

```
conv -dir puce.inr detl1 | mb -n 2 | xvis -nu
```



Détecteur de lignes

Autres orientations

- On filtre avec les noyaux orientés verticalement et dans les deux diagonales :

-1	2	-1
-1	2	-1
-1	2	-1

-1	-1	2
-1	2	-1
2	-1	-1

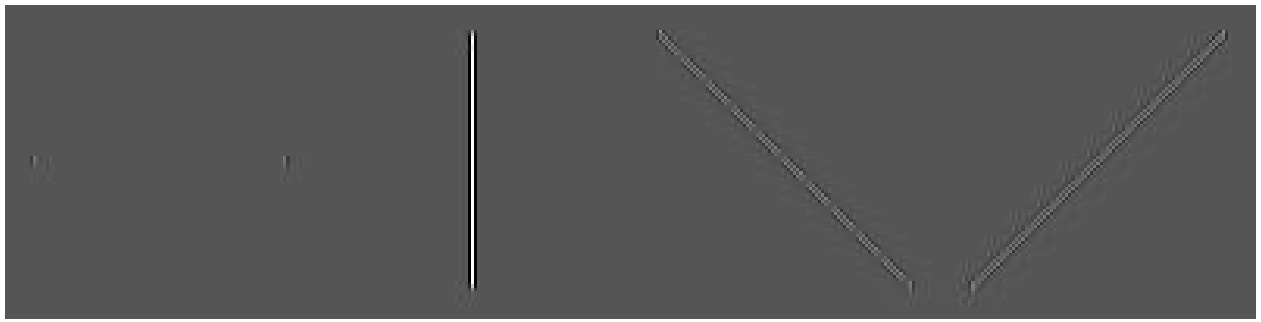
2	-1	-1
-1	2	-1
-1	-1	2

```
melg line1 lines -x 400 -y 100
melg line2 -ixo 101 lines
melg line3 -ixo 201 lines
melg line4 -ixo 301 lines
```

Détecteur de lignes

Autres orientations

```
echo -1 2 -1 -1 2 -1 -1 2 -1 | cim -x 3 -y 3 -r | \
        conv -dir lines | norma | xvis -nu
```

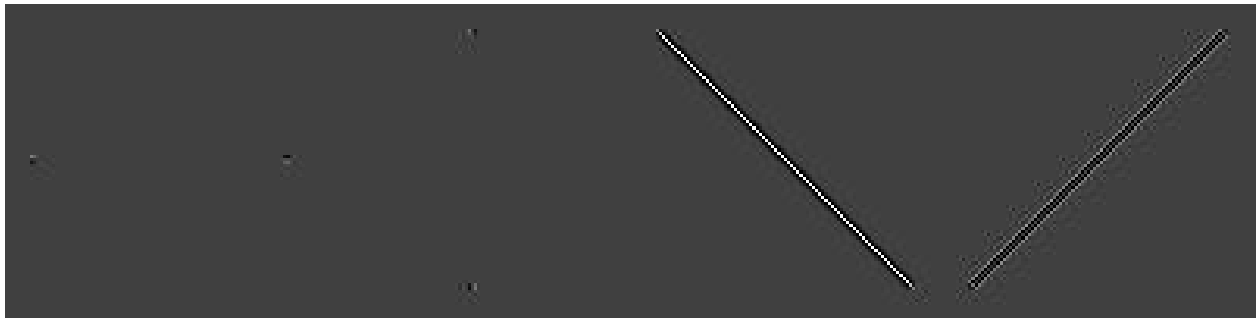


```
echo -1 -1 2 -1 2 -1 2 -1  -1 | cim -x 3 -y 3 -r | \
        conv -dir lines | norma | xvis -nu
```

Détecteur de lignes

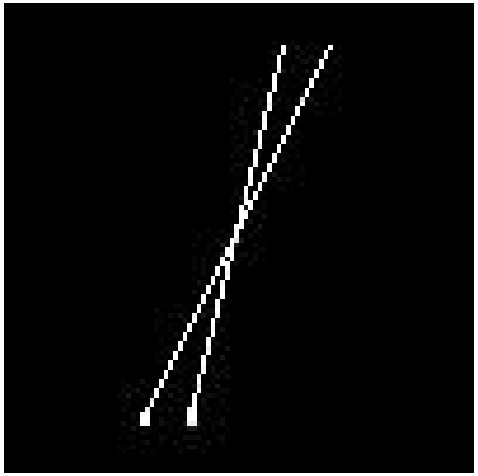
Autres orientations

```
echo 2 -1 -1 -1 2 -1 -1 -1 2  | cim -x 3 -y 3 -r | \
        conv -dir lines | norma | xvis -nu
```



Encore d'autres orientations ?

```
./drawline lines12 100 100 60 10 40 90
./drawline lines12 100 100 70 10 30 90
xvis -nu lines12
for a in 1 2 3 4; do
  conv -dir lines12 detl$a | \
    melg - -x 400 -y 100 \
      -ixo $(((a-1)*100+1)) lines12detl
done
norma lines12detl | xvis -nu
```



Détection des droites

Bilan du filtrage spatial

- Certaines orientations sont inacessibles : on peut créer des opérateurs plus grands (5×5, 7×7, ...) pour accéder à ces orientations.
- Difficulté du choix de l'opérateur.
- La détection peut se faire en n étapes distinctes (une pour chaque direction), il faut ensuite rassembler les résultats pour former une unique image (somme + normalisation éventuelle).

Détection de droites par transformée de Hough

Principe

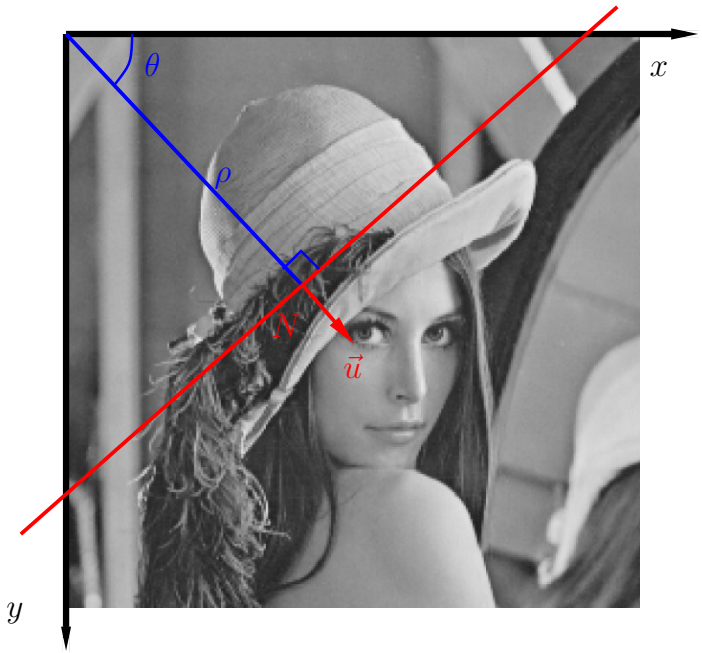
- Le filtre précédent est local, il ne permet pas de détecter une ligne entière.
- La transformée de Hough repose sur le principe d'exploration **systématique** de toutes les droites intersectant le domaine de l'image :
  - Une droite : $y = ax + b$, en faisant varier a et b on obtient toutes les droites.
  - On comptabilise le nombre de fois où la droite intersecte les contours de l'image : un nombre significativement grand indique une droite.
  - Finalement, il s'agit d'un histogramme 2D : on utilise un tableau appelé **accumulateur** : à chaque intersection de la droite $y = ax + b$ avec un contour, on incrémente l'accumulateur au point (a, b).
  - Les maxima locaux de l'accumulateur indiquent des droites candidates.

Transformée de Hough

Représentation Polaire

- Le modèle de droite : $y = ax + b$ possède l'inconvénient de ne pas pouvoir représenter les droites d'équation $x = x_0$ (tangente verticale).
- On utilise plutôt une représentation *polaire* de la courbe c'est-à-dire :

$$x \cos\theta + y \sin\theta = \rho$$



- $\rho \in [0, \rho_{max} = \sqrt{dimx^2 + dimy^2}]$
- $\theta \in [-\frac{\pi}{2}, \pi]$
- L'équation est facile à retrouver :

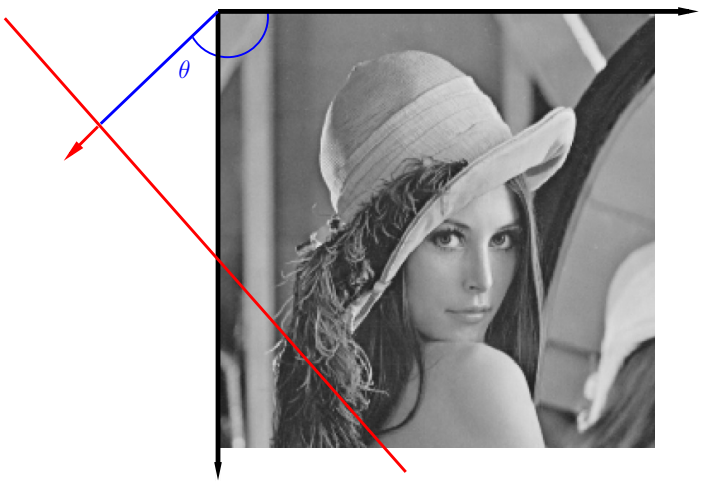
$$\begin{aligned} \overrightarrow{NM}.\vec{u} &= 0 \\ (x - x_N)\cos\theta + (y - y_N)\sin\theta &= 0 \\ x\cos\theta + y\sin\theta &= \rho \\ \rho \equiv x_N \cos\theta + y_N \sin\theta & \end{aligned}$$

Transformée de Hough

Représentation Polaire

- Le modèle de droite : $y = ax + b$ possède l'inconvénient de ne pas pouvoir représenter les droites d'équation $x = x_0$ (tangente verticale).
- On utilise plutôt une représentation *polaire* de la courbe c'est-à-dire :

$$x \cos\theta + y \sin\theta = \rho$$



- $\rho \in [0, \rho_{max} = \sqrt{dimx^2 + dimy^2}]$
- $\theta \in [-\frac{\pi}{2}, \pi]$
- L'équation est facile à retrouver :

$$\begin{aligned} \overrightarrow{NM}.\vec{u} &= 0 \\ (x - x_N)\cos\theta + (y - y_N)\sin\theta &= 0 \\ x\cos\theta + y\sin\theta &= \rho \end{aligned}$$

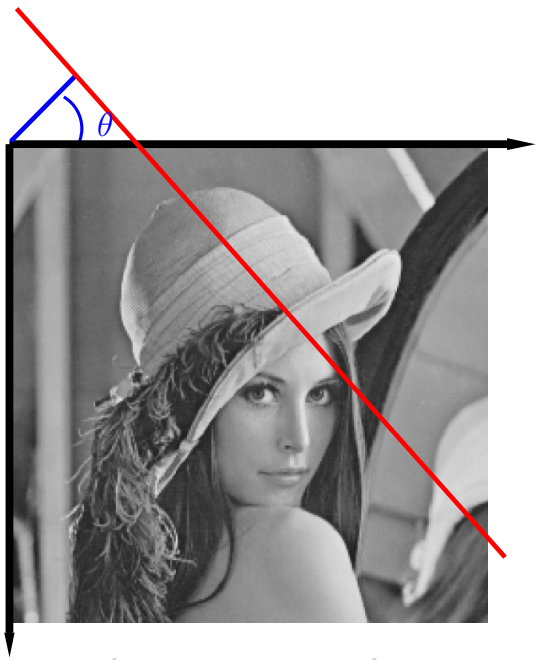
$$\rho \equiv x_N \cos\theta + y_N \sin\theta$$

Transformée de Hough

Représentation Polaire

- Le modèle de droite : $y = ax + b$ possède l'inconvénient de ne pas pouvoir représenter les droites d'équation $x = x_0$ (tangente verticale).
- On utilise plutôt une représentation *polaire* de la courbe c'est-à-dire :

$$x \cos\theta + y \sin\theta = \rho$$



- $\rho \in [0, \rho_{max} = \sqrt{dimx^2 + dimy^2}]$
- $\theta \in [-\frac{\pi}{2}, \pi]$
- L'équation est facile à retrouver :

$$\begin{aligned} \overrightarrow{NM}.\vec{u} &= 0 \\ (x - x_N)\cos\theta + (y - y_N)\sin\theta &= 0 \\ x\cos\theta + y\sin\theta &= \rho \\ \rho \equiv x_N \cos\theta + y_N \sin\theta & \end{aligned}$$

Transformée de Hough

Choix dans les représentations

- Pas d'unicité du repère image. Par exemple on représente fréquemment par ((x_I, y_I) coordonnée du centre de l'image) :

$$\begin{aligned} (x - x_I)\cos\theta + (y - y_I)\sin\theta &= \rho \\ \rho \in [-\frac{\rho_{max}}{2}, \frac{\rho_{max}}{2}] & \quad \theta \in [0, \pi] \end{aligned}$$

- Ou encore :

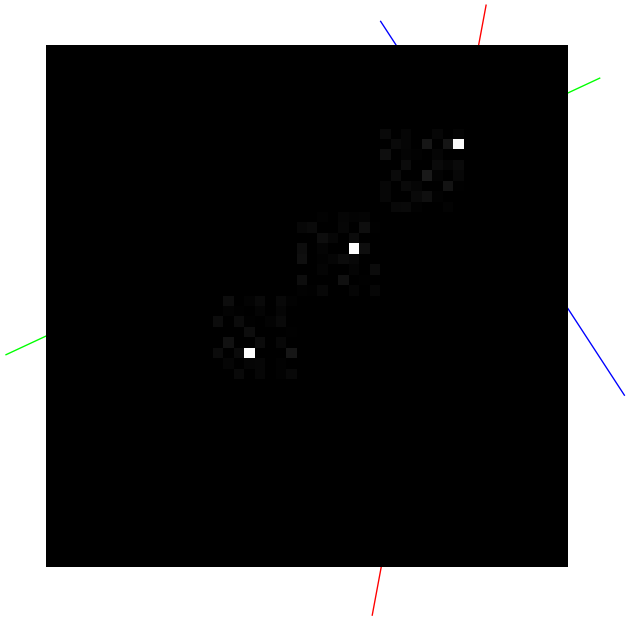
$$\begin{aligned} (x - x_I)\cos\theta + (y - y_I)\sin\theta &= \rho \\ \rho \in [0, \rho_{max}] & \quad \theta \in [0, 2\pi] \end{aligned}$$

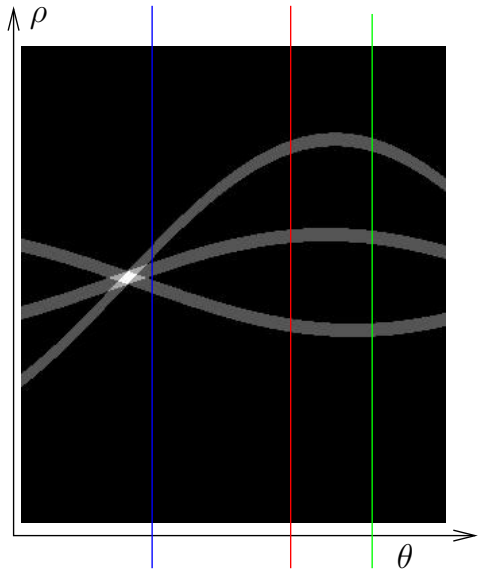
- ...

Transformée de Hough

Exploration du domaine

- Par un point, passe une infinité de droites : une courbe dans l'accumulateur.
- L'accumulateur vaut alors 0 ou 1 (sur les courbes).

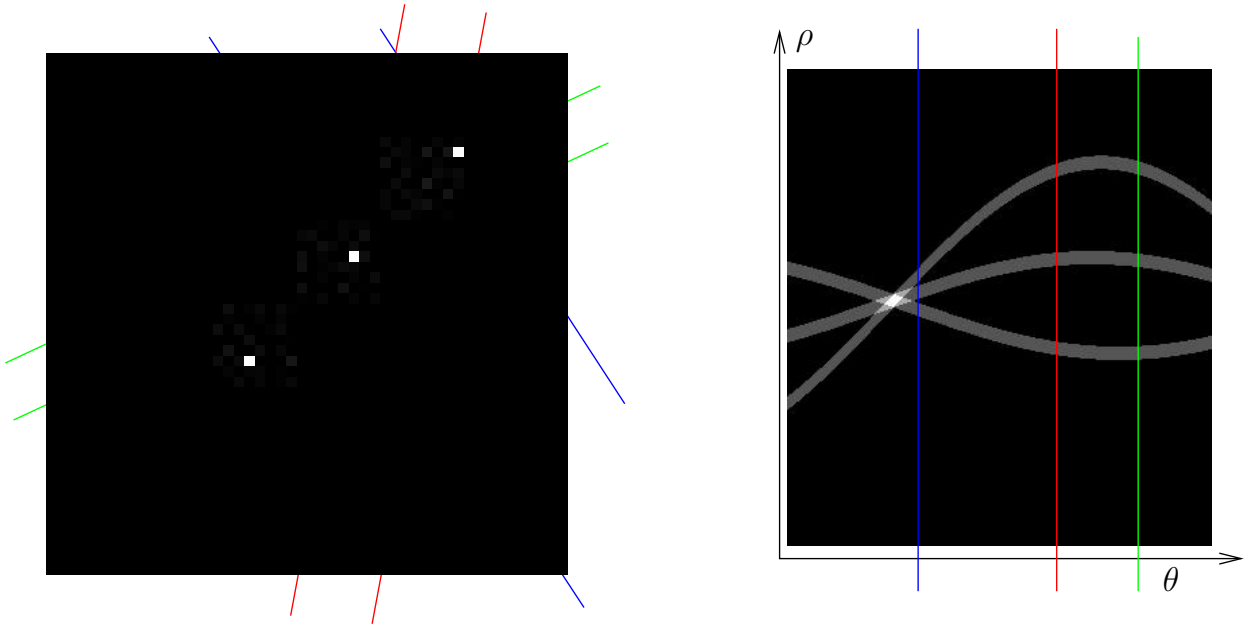




Transformée de Hough

Exploration du domaine

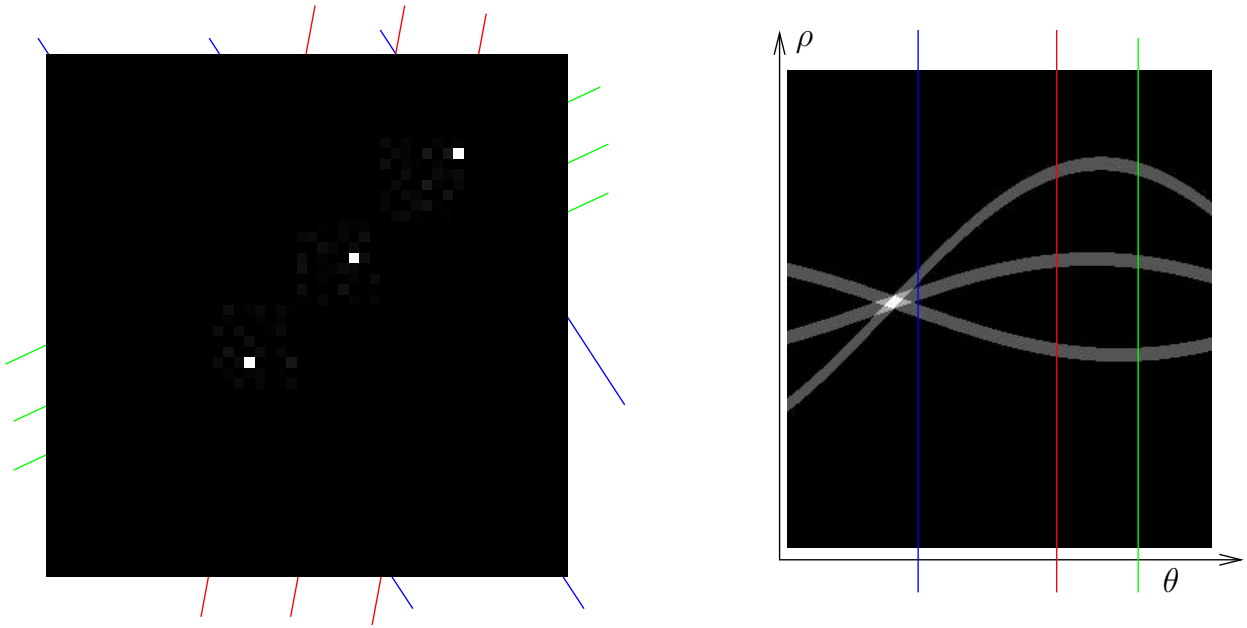
- Avec deux points : deux courbes dans l'accumulateur.
- Par deux points : au plus une seule droite.
- Au point d'intersection : 2, sinon 1 sur les courbes, 0 ailleurs.



Transformée de Hough

Exploration du domaine

- Avec n points : n courbes dans l'accumulateur.
- n points alignés : on augmente le score à l'intersection des n courbes : n.



Transformée de Hough

Application aux images naturelles

- Comment reconnaître des droites dans les images naturelles ?
- En utilisant un détecteur de contours !
- On peut identifier alors les pixels contours comme appartenant à la forme que l'on recherche.
- En pratique, l'accumulateur de Hough est calculé sur une image binaire généralement issue d'un détecteur de contours.

Transformée de Hough

Algorithme

```
algo hough( I: image, dtheta, drho, H: accumulateur);
// l'angle theta varie entre 0 et Pi
// le parametre rho varient de facon a couvrir
// le domaine de l'image soit [-rmax,+rmax]
rmax := (I.dimx^2+I.dimy^2)^0.5;
allouer_et_initialiser(H);
pour rho variant de -rmax a rmax par pas de drho faire
  pour theta variant de 0 a Pi par pas de dtheta faire
    pour chaque pixel p faire
      si I[p] == contour alors
        si appartient_droite( p, theta, rho) alors
          H[theta][rho] ++;
        fin si
      fin si
    fin pour
  fin pour
fin algo
```

Transformée de Hough

Algorithme

```
algo appartient_droite(p:pixel,theta,rho): booleen;
  x := p.x - I.x/2;
  y := p.y - I.y/2;
  si |cos(theta)*x+sin(theta)*y-rho| < SEUIL alors
    retourne VRAI;
  sinon
    retourne FAUX;
  fin si
fin algo
```

Transformée de Hough

Transformation inverse

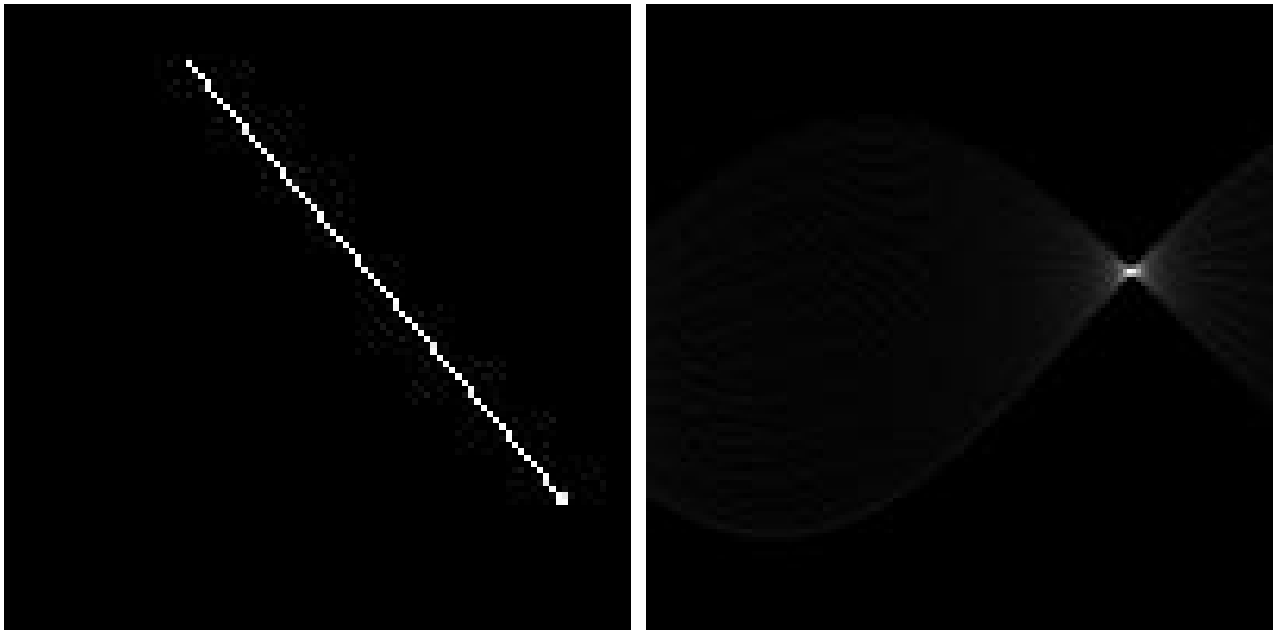
- Après calcul de la transformée de Hough, les maxima locaux dans l'accumulateur correspondent aux paramètres pour une ligne détectée.
- La transformée de Hough donc être inversible afin de reconstruire la forme détectée.
- Exemple : soit (θ_1, ρ_1) un maximum local dans l'accumulateur. La droite reconstruite a pour équation : $x \cos\theta_1 + y \sin\theta_1 = \rho_1$.
- Pour tracer la droite il suffit de calculer les intersections avec les bords du domaine image, soit :

$$\begin{aligned} x_a = 1 \quad &\text{et} \quad x_a \cos\theta_1 + y_a \sin\theta_1 = \rho_1 \\ x_b = DIMX \quad &\text{et} \quad x_b \cos\theta_1 + y_b \sin\theta_1 = \rho_1 \\ y_a = 1 \quad &\text{et} \quad x_a \cos\theta_1 + y_a \sin\theta_1 = \rho_1 \\ y_b = DIMY \quad &\text{et} \quad x_b \cos\theta_1 + y_b \sin\theta_1 = \rho_1 \end{aligned}$$

Transformée de Hough

Analyse de l'accumulateur

```
./drawline l1 100 100 30 10 90 80
xvis -nu l1
```



(a) image `l1`

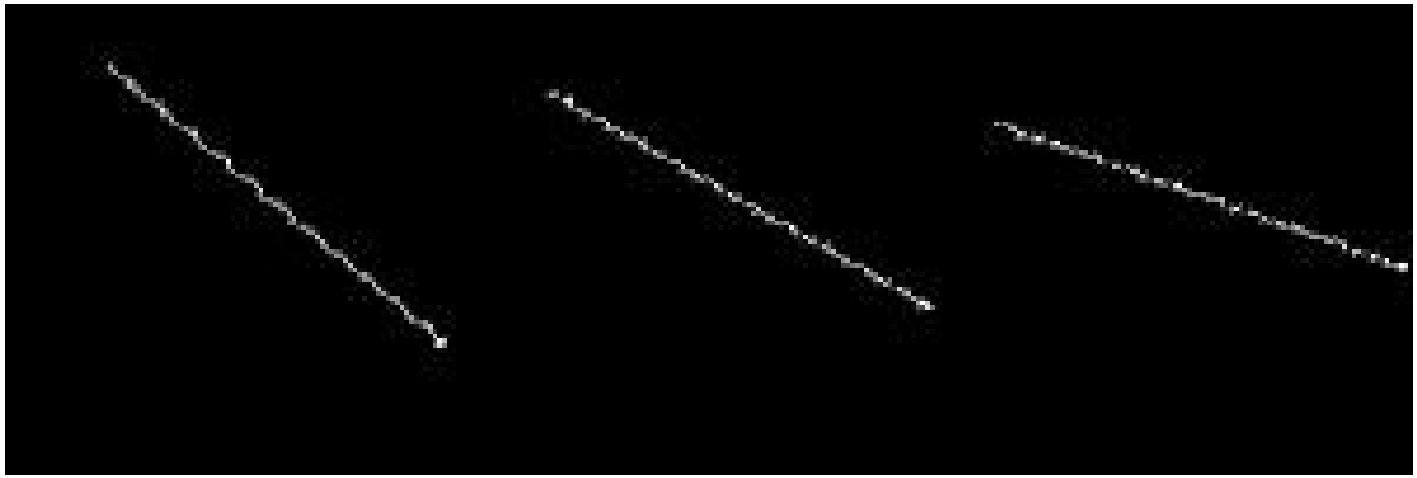
(b) accumulateur (DIMX=θ, DIMY=ρ)

- Maximum en (2.43363, -10).

Transformée de Hough

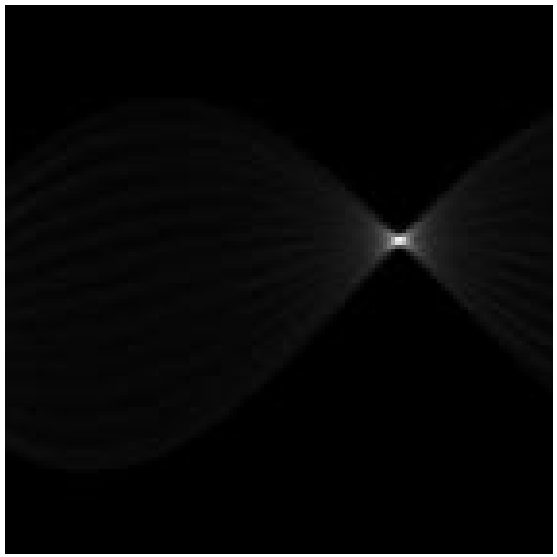
Autres angles et impact dans l'accumulateur

```
for a in 10 20 30 ; do
  rot l1.inr >l1-a$a.inr `par -x -y l1.inr` -a $a
done
xvis -nu -grp 3 -nhz 3 l1-a*.inr
```



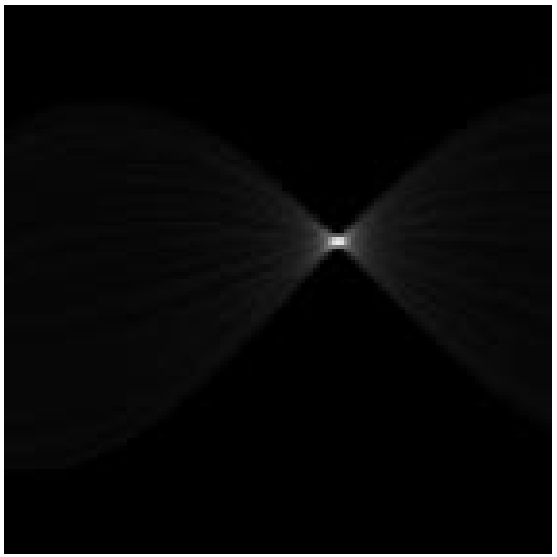
Transformée de Hough

Autres angles et impact dans l'accumulateur



(c) détecté : $\theta = 2.26$

(d) détecté : $\theta = 2.08$



(e) détecté : $\theta = 1.90$

Transformée de Hough

Autres angles et impact dans l'accumulateur

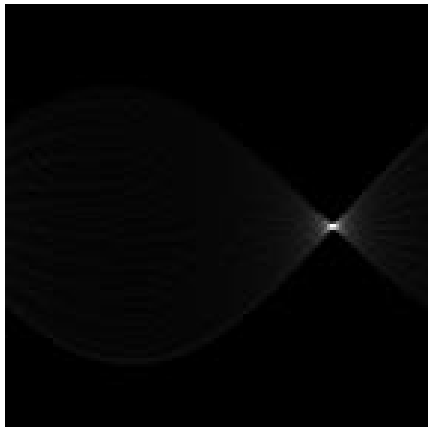
Transformée de Hough

Variation de ρ

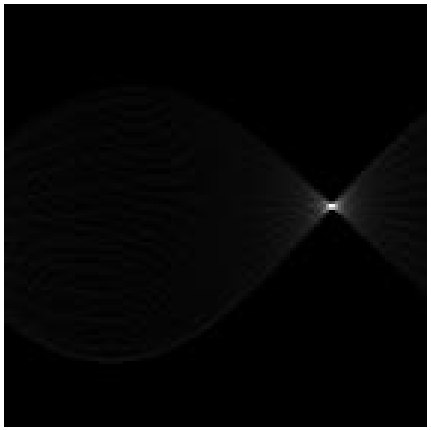
```
for d in -10 -5 5 10; do
  rot l1.inr > l1-d$d.inr `par -x -y l1.inr` \
      -a 0 -tx $d -ty $((-d))
done
```

Transformée de Hough

Variation de ρ

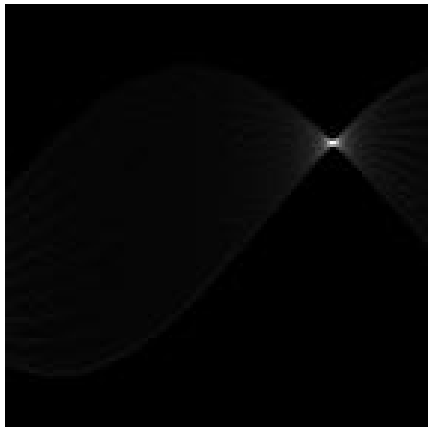


(f) $\rho = -10$



(g) $\rho = -5$

(h) $\rho = 5$

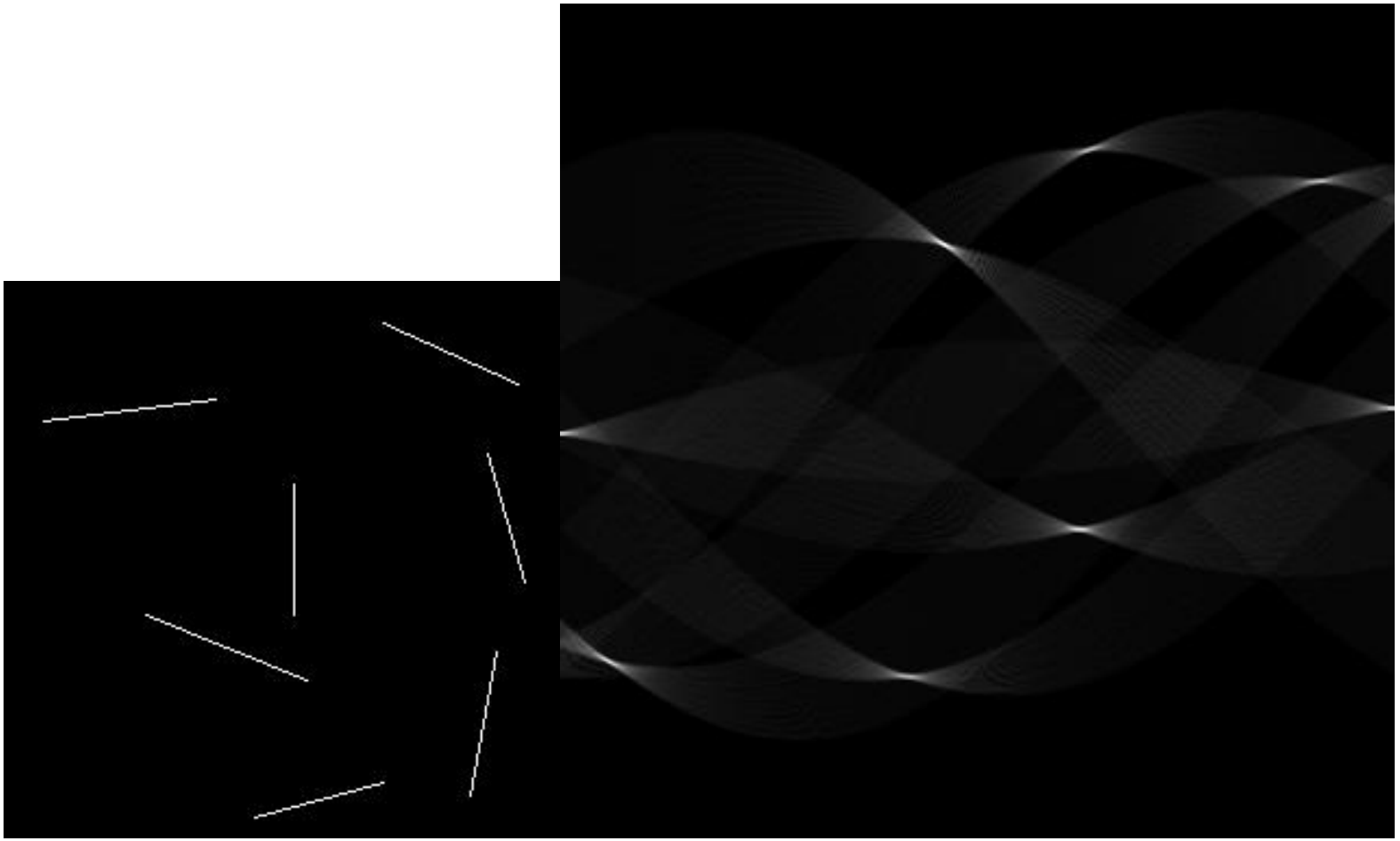


(i) $\rho = 10$

Transformée de Hough

Plus de lignes

- Différents segments, de différentes longueurs :



Transformée de Hough

Plus de lignes

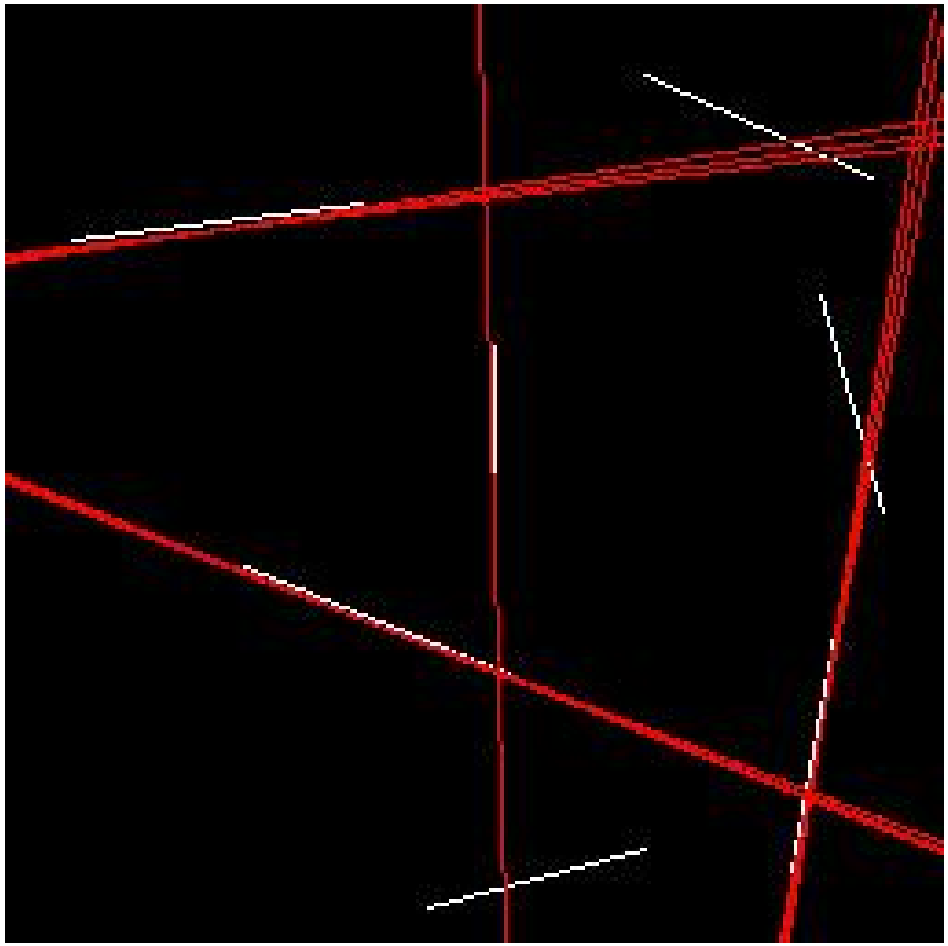


FIGURE – Les 10 premiers maxima locaux

Transformée de Hough

Plus de lignes

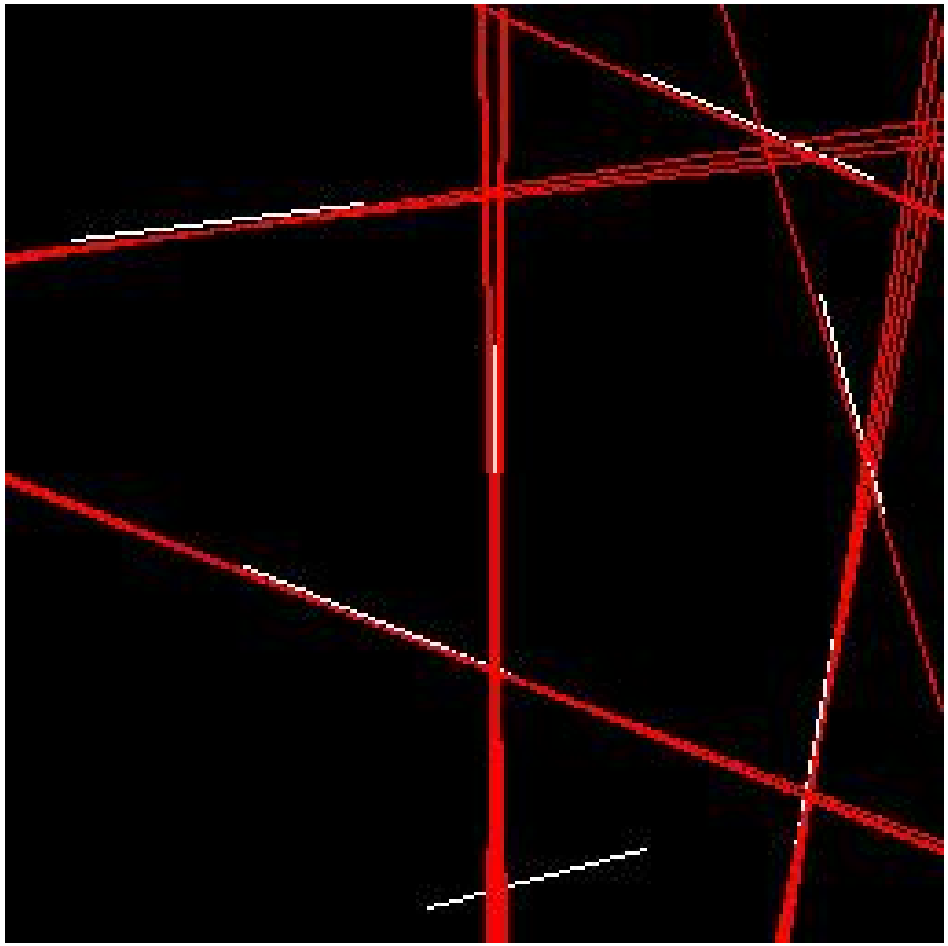


FIGURE – Les 20 premiers maxima locaux

Transformée de Hough

Plus de lignes

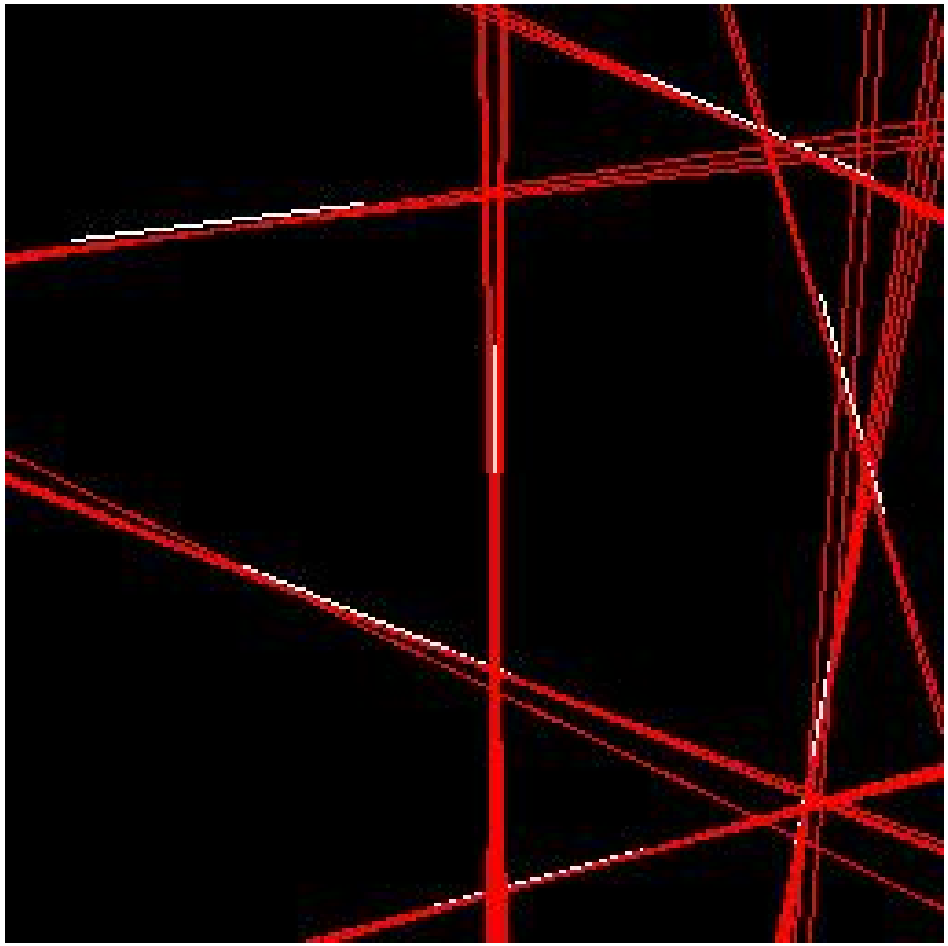
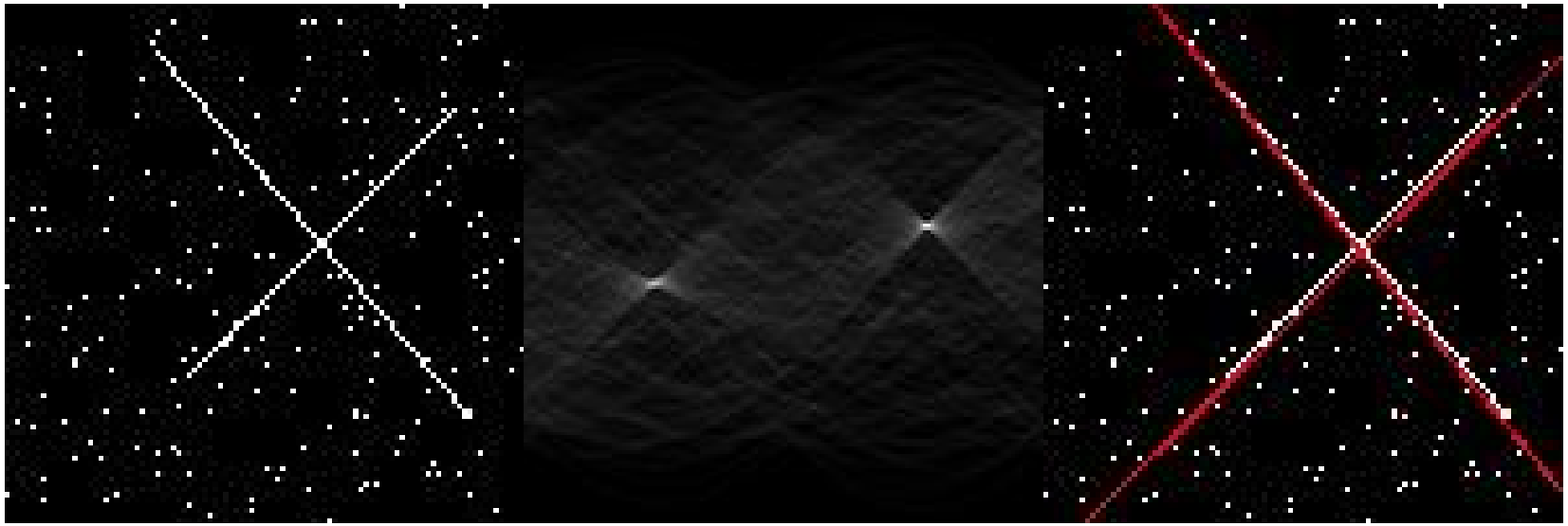


FIGURE – Les 30 premiers maxima locaux

Transformée de Hough

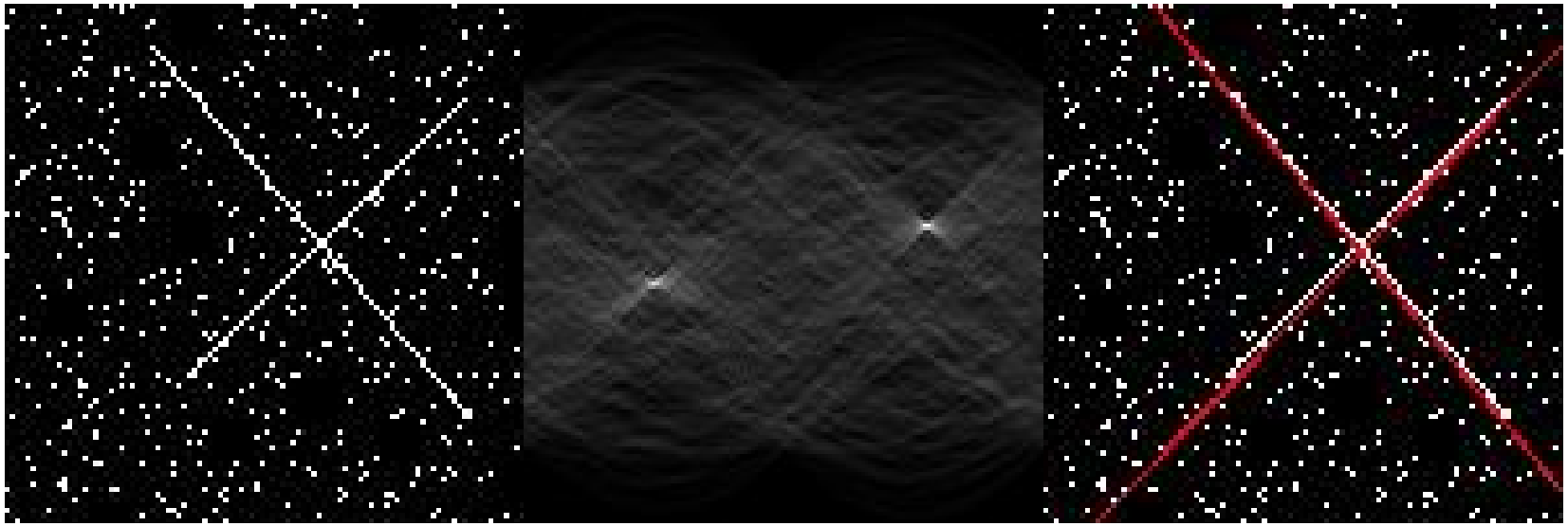
Robustesse au bruit

- Robustesse au bruit : on teste sur des images bruitées (bruit additif gaussien).



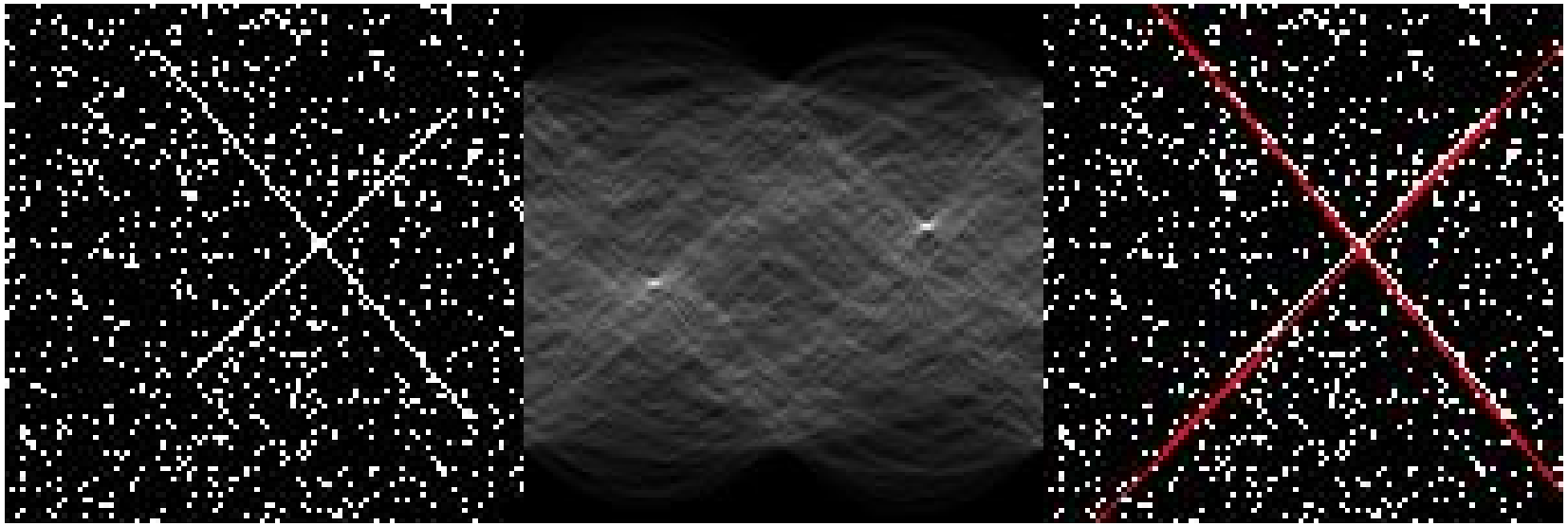
Transformée de Hough

Robustesse au bruit



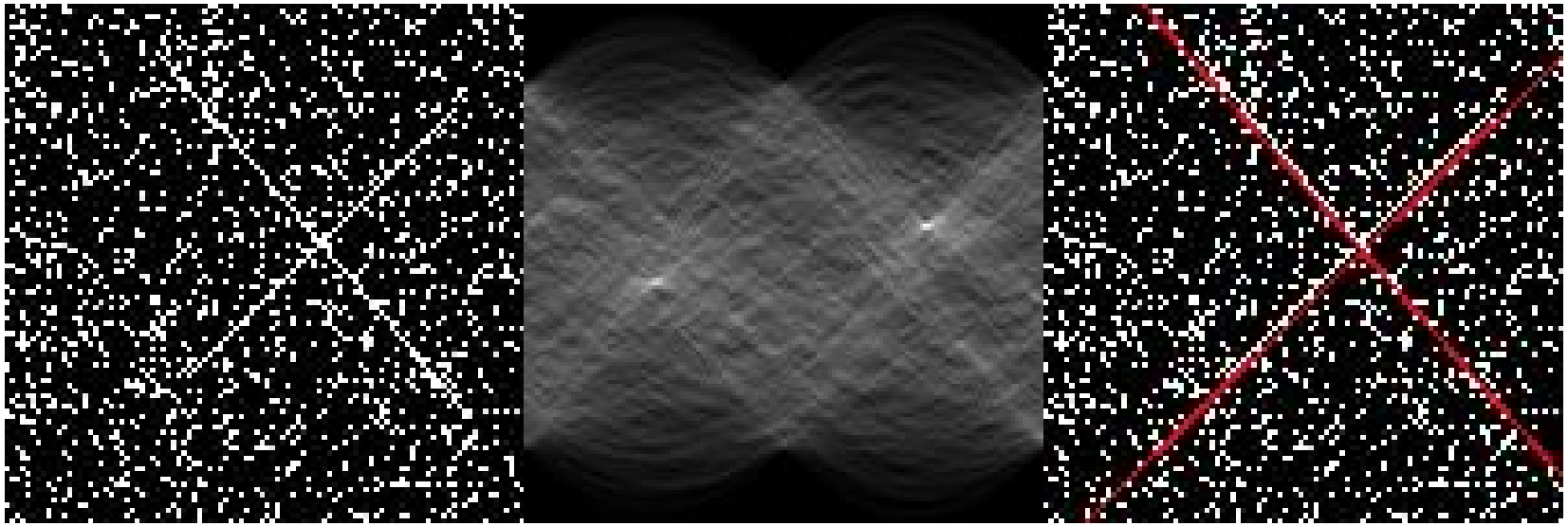
Transformée de Hough

Robustesse au bruit



Transformée de Hough

Robustesse au bruit

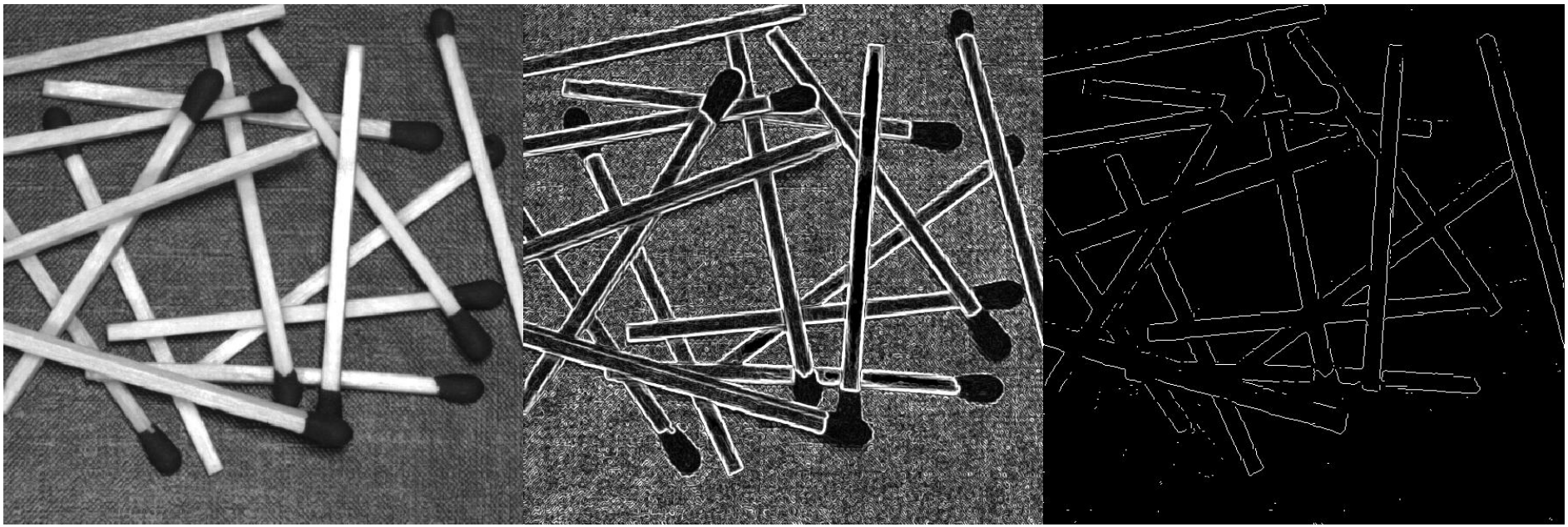


Transformée de Hough

Application sur des images naturelles

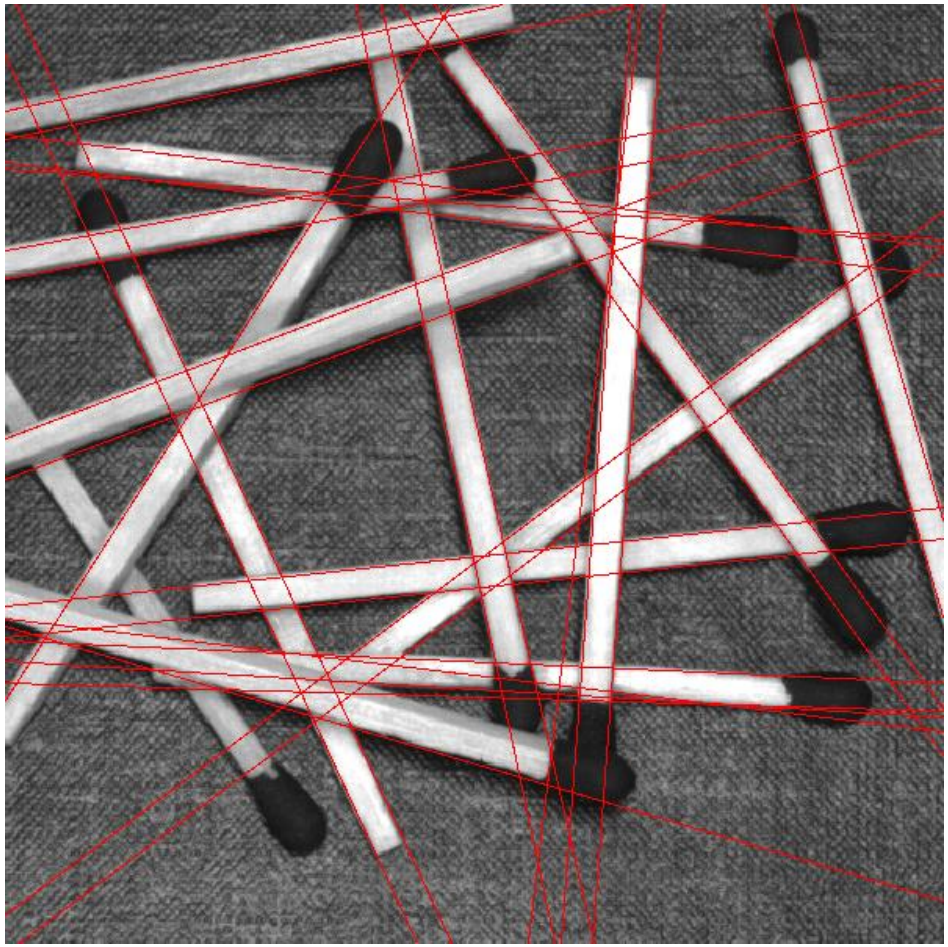
- application d'un détecteur de contours.

```
detc -sob matches > matches.cont
ical matches.cont
      0.000000       0.365724       2.694811
mh -n 1 matches.cont | aff > matches.bin
```



Transformée de Hough

Application sur des images naturelles



Généralisation de la transformée de Hough

- On peut facilement étendre la transformée de Hough pour la recherche de toutes formes “paramétrisables”.
- Par exemple l’ellipse : un pixel de coordonnées (x, y) appartient à l’ellipse de centre (x_0, y_0), de petit et grand rayons a et b si :

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} = 1$$

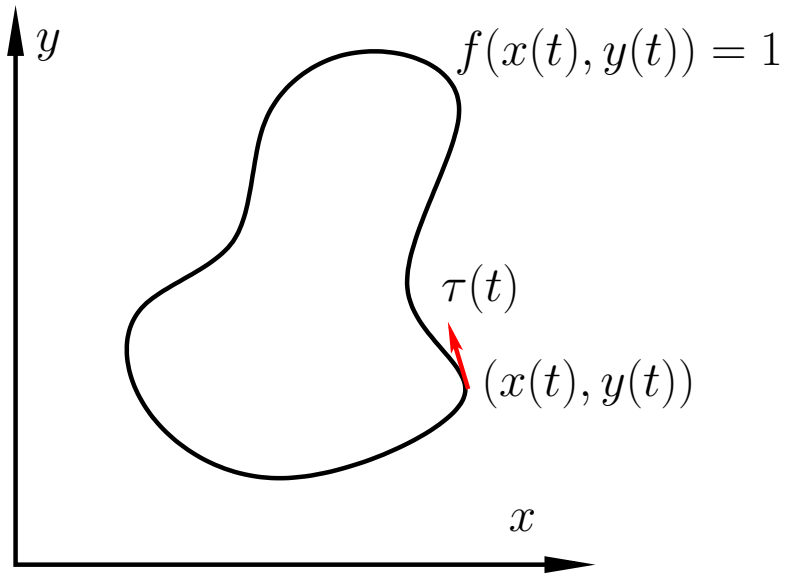
- On construit donc un accumulateur à 4 dimensions et on recherche les maxima locaux.
- On reconstruit une ellipse avec la définition paramétrique de l’ellipse soit :

$$\begin{aligned} x(\theta) &= a\cos\theta + x_0 \\ y(\theta) &= b\sin\theta + y_0 \\ \theta &\in [0, 2\pi[\end{aligned}$$

Généralisation de la transformée de Hough

Réduction de l'espace de recherche

- Reconnaître une forme dans une image de contours, c'est chercher une fonction $f(x, y) = 1$, et on prend f comme une fonction dépendant de n paramètres $\Theta = (\theta_1, \cdots, \theta_n)$.
- Nécessité de construire un accumulateur à n dimensions : coûteux !
- Une astuce pour réduire le nombre de paramètres :



- Soit une forme (une courbe) vérifiant l'équation (implicite) : $f(x, y) = 1$.
- C'est une courbe : il existe un paramétrage (t) qui décrit tous les points de la courbe : $x = x(t)$ et $y = y(t)$

Généralisation de la transformée de Hough

Réduction de l'espace de recherche

- En tout point (x, y) de la courbe, on a $\tau \perp \nabla f$ (τ est la tangente).
- Preuve :

$$
\begin{aligned}
f(x(t), y(t)) &= 1 \ \forall t \\
\frac{d}{dt} f(x(t), y(t)) &= 0 \\
\frac{d}{dt} f(x(t), y(t)) &= \frac{dx(t)}{dt} f_x(x(t), y(t)) + \frac{dy(t)}{dt} f_y(x(t), y(t)) \\
t_x f_x + t_y f_y &= 0 \\
\tau . \nabla f &= 0
\end{aligned}
$$

Généralisation de la transformée de Hough

Réduction de l'espace de recherche

- On a :

$$\tan\left(\widehat{\langle \nabla f, (0x)\rangle}\right) = \frac{f_y}{f_x}$$

et donc :

$$\tan\left(\widehat{\langle \tau, (0x)\rangle}\right) = -\frac{f_x}{f_y}$$

- On note ϕ l'angle entre τ et l'horizontale.
- Pour une image I de contours, $f \equiv I$: on mesure ϕ dans l'image, à partir du calcul de ses gradients.

Généralisation de la transformée de Hough

Réduction de l'espace de recherche

- Finalement, on doit résoudre :

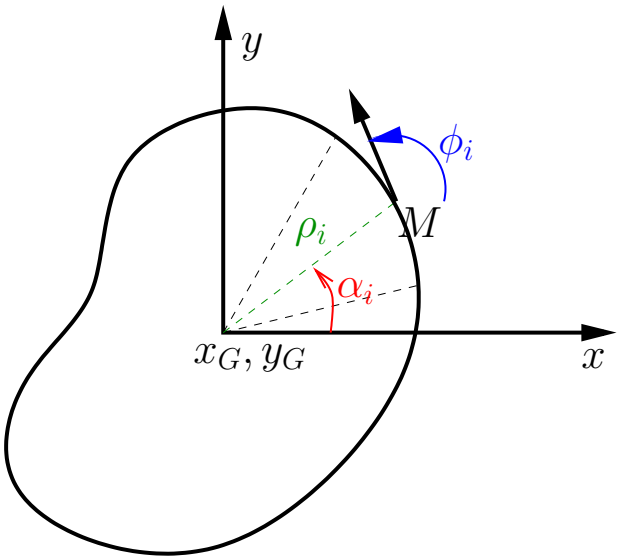
$$\begin{aligned} f(x, y, \theta_1, \cdots, \theta_n) &= 1 \\ -\frac{f_x(x, y, \theta_1, \cdots, \theta_n)}{f_y(x, y, \theta_1, \cdots, \theta_n)} &= \tan\phi \end{aligned}$$

- Deux équations, n paramètres : on peut réduire le système à $n-1$ paramètres.
- Exemple : le cercle.

$$\begin{aligned} f(x, y, x_0, y_0, r) &= (x-x_0)^2 + (y-y_0)^2 - r^2 + 1 = 1 \\ f_x &= 2(x-x_0) \quad f_y = 2(y-y_0) \\ -\frac{x-x_0}{y-y_0} &= \tan\phi \\ \Rightarrow \quad g(x, y, y_0, r) &= (y-y_0)^2(\tan^2\phi + 1) - r^2 = 0 \end{aligned}$$

Transformée de Hough Généralisée

- On peut utiliser le principe de l'orientation locale de la forme (ϕ) pour généraliser la transformée de Hough à des formes prédéfinies quelconques sans pour autant avoir une définition analytique de ces formes.
- Considérer une courbe fermée quelconque et son centre de gravité (x_G, y_G). On échantillonne la courbe avec des triplets $(\alpha_i, \rho_i, \phi_i)$:



- $\rho_i = CM$, $\phi_i = < \widehat{\tau_M, Cx} >$
- Et on forme la table F (forme à reconnaître) :

ϕ_1	$(\rho_1^1, \alpha_1^1), ..., (\rho_1^{n_1}, \alpha_1^{n_1})$
ϕ_2	$(\rho_2^1, \alpha_2^1), ..., (\rho_2^{n_2}, \alpha_2^{n_2})$
...	...
ϕ_k	$(\rho_k^1, \alpha_k^1), ..., (\rho_k^{n_k}, \alpha_k^{n_k})$

Transformée de Hough Généralisée

algorithme

```
algo hough_general(I:image, F:table, H:accumulateur);
    initialiser (H);
    pour chaque c point de contour dans I faire
       phi := - arctan(Ix(c)/Iy(c));
       pour chaque (r,a) dans F(phi) faire
          # (x,y) coordonnees de G
          G.x := c.x - r*cos(a);
          G.y := c.y - r*sin(a);
          H(G.x,G.y) ++;
       fin pour
    fin pour
fin algo
```

- Les maxima locaux de H indiquent des candidats centre de gravité de la forme F.

Transformée de Hough Généralisée

invariance par rotation et changement d'échelle

- En pratique, l'algorithme est restrictif car la forme F à rechercher doit avoir la même orientation et taille.
- On peut considérer un accumulateur à quatre dimensions en ajoutant un facteur d'échelle S et une orientation θ : l'accumulateur a donc 4 dimensions (x_G, y_G, S, θ)
- La formule de reconstruction devient alors :

$$\begin{aligned} x &= x_G + r \times S \times \cos(\alpha + \theta) \\ y &= y_G + r \times S \times \sin(\alpha + \theta) \end{aligned}$$

- En pratique : l'orientation est très sensible au bruit.