Master Info
Image

# Résolution de problèmes par minimisation d'une fonctionnelle et applications

Plis fòs ba pengwen là !

Vincent Pagé vpage@univ-ag.fr Université Antilles Guyane

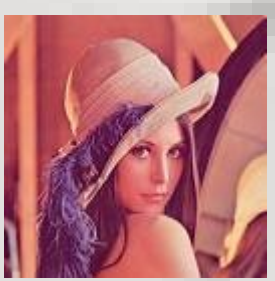
# Objet de ce cours ?

Lorsque vous interprétez une image, votre cerveau réalise énormément d'opérations. Par exemple, si vous regardez l'image suivante et que vous essayez de localiser les yeux du personnage :

- Vous faites des mesures (couleurs)
- Vous reliez ces mesures a des choses connues (corps humains)
- Votre connaissance du corps humain vous dit qu'il y a des yeux... et vous donne un modele pour ces yeux (blanc autour, plus ou moins rond...)
- Vous refaites des mesures dans l'image pour trouver les yeux.

# Objet de ce cours ? 2

Pour faire de l'analyse automatique d'image, on est obligé de simplifier :



- On va supposer que l'on sait que l'on cherche des yeux dans une image comportant des corps humains (c'est le contexte applicatif)

- On devra définir un modèle de correspondance entre un objet réel (des yeux) et les mesures que l'on peut faire dans l'image (niveaux de gris).

Si le programme cherche un oeil, et que l'on cherche une zone plus ou moins ronde, avec du blanc autour... on parle de traitement **bas niveau**.

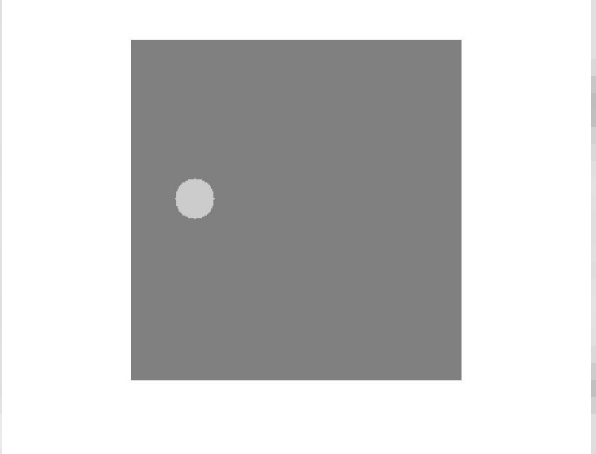
S'il cherche a interpréter l'image en termes de visages, et utiliser ces informations pour trouver les yeux, on parle de traitement **haut niveau**.

Pour nous : Bas niveau !

# Objet de ce cours ? 3

Même avec cette simplification, cela reste éventuellement difficile...

Ex : Trouver ou se trouve le rond clair (de 15 pixels de diametre) présent dans l'image :



Comment modéliser le problème pour en tirer un programme ?

Une solution consiste à :

- définir un critère (une valeur pour la « rond clair »-itude en chaque point)

- rechercher l'optimum du critère sur les solutions possibles (ici les différentes positions)

# Objet de ce cours ? 4

Cette démarche est souvent bien utile...Et c'est ce que l'on fera pour différents problèmes classiques d'image.

Reste a définir

- Un « bon critère » pour le problème
- Une méthode pour trouver l'optimum.

Dans le cas de l'exemple précédent,

- Le critère peut être : pour un pixel de l'image, calculer la somme des intensité sur le rond de largeur 15 centré sur le pixel considéré

- La méthode pour trouver le maximum : exhaustif, on regarde toutes les positions possibles.

# Exemples d'application

Dans ce cours, nous allons toujours procéder de la façon précédente... sur un grand nombre de problèmes.

Ceci nous permettra de voir :

- des critères utiles / classiques.

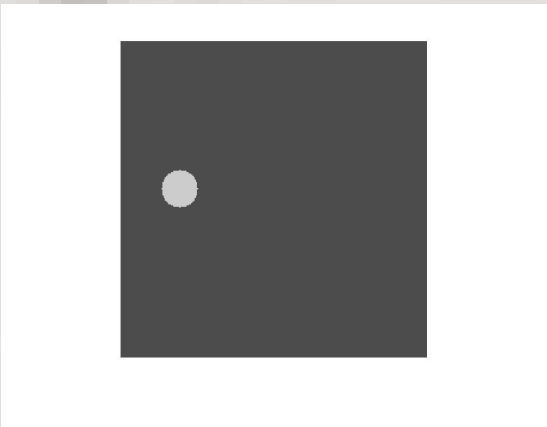
- des méthodes d'optimisations classiques aussi.

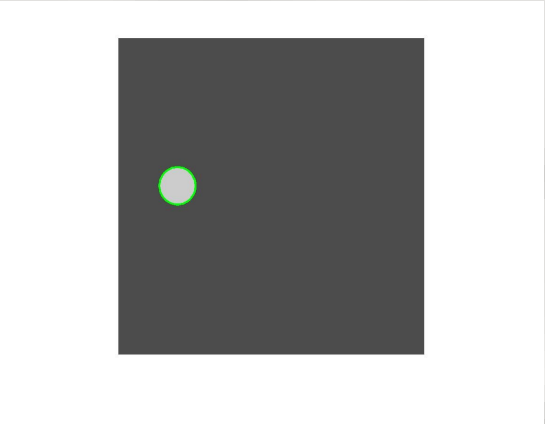
L'objectif étant de vous donner une certaine culture en traitement d'image... et la bonne habitude de modéliser correctement vos problèmes !

Problèmes concernés : Localisation/Détection, Recalage, Segmentation.

# Localisation

En appliquant la technique précédente, voici les résultats :



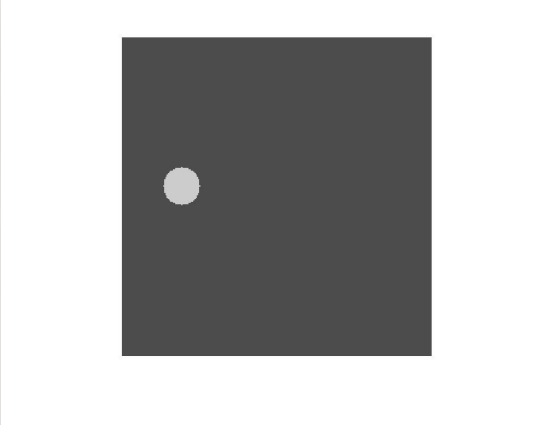


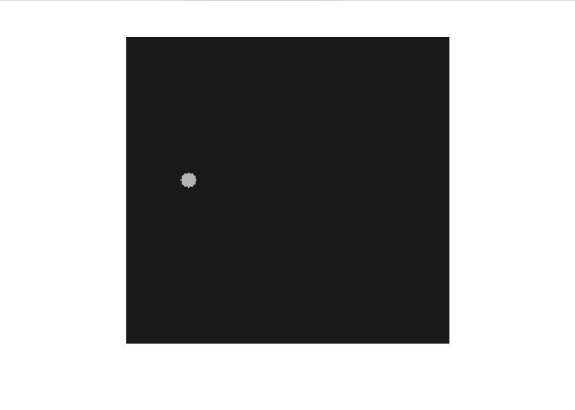
(Bien joué...)

En TP, vous essayerez cela ... En fait, on peut montrer que cette technique est la meilleure... sous certaines conditions... (lesquelles ?)

# Localisation (taille inconnue)

Cette fois, il s'agit de localiser le rond clair, de diamètre variable (disons entre 5 et 35 pixels de diamètre) dans des images comme celles qui suivent :





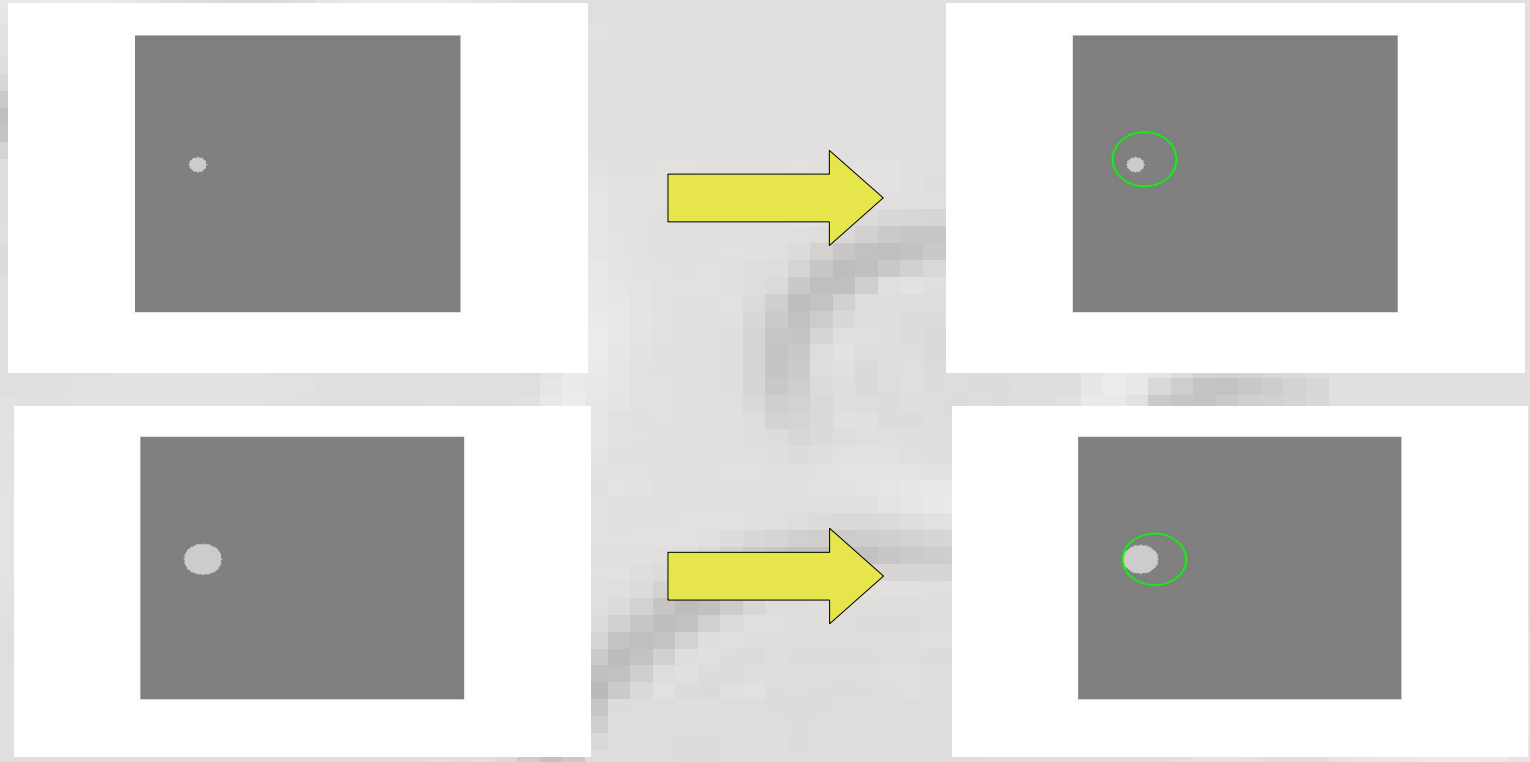
Conservons le critère précédent...

- pour un pixel (x,y) de l'image, calculer la somme des intensité sur le rond de largeur d centré sur le pixel considéré

- L'espace de recherche est l'ensemble des (x,y,d) possibles.

# Localisation (taille inconnue)

Résultats :



Ce qui ne va pas : les gros ronds ne « sortent » pas car la somme des intensités est toujours plus grande sur un gros rond que sur un petit ... notre critère n'est pas bon...

# Localisation (taille inconnue)

Deux méthodes (vues en détail plus loin) :

- On trouve un critère qui soit indépendant de la taille de la cible réelle. On parle alors de critère invariant pour le paramètre qui nous intéresse.
- On modifie le critère actuel pour qu'il pénalise les grandes tailles de cibles. Ainsi, une grosse cible aura moins de chance d'être selectionnée à moins d'etre « vraiment » présente.

La seconde méthode relève évidemment du « bricolage ». Elle est néanmoins très usitée lorsque trouver un critère invariant est difficile.

# Critère invariant par changement de taille

Concentrons nous sur la première méthode.

Notre premier critère etait : Somme des intensités sur la cible. Une variante qui soit invariante en fonction de la taille de la cible serait « Moyenne des intensités sur la cible » ...

Le problème est le suivant : a priori, toutes les cibles testées de taille inférieure à la taille vraie de la cible auront la même moyenne (avec des variations dues au bruit).

Notre critere simplement n'est pas bon : il ne prend pas en compte l'exterieur de la cible testée (celui ci doit etre plus sombre que l'intérieur si la cible testée a la taille correcte).

Pourriez vous définir un critère qui fonctionne ?

# Terme de Pénalité

Voyons donc la seconde solution : pénaliser les petites cibles...

Notre critère sera composé de deux termes :

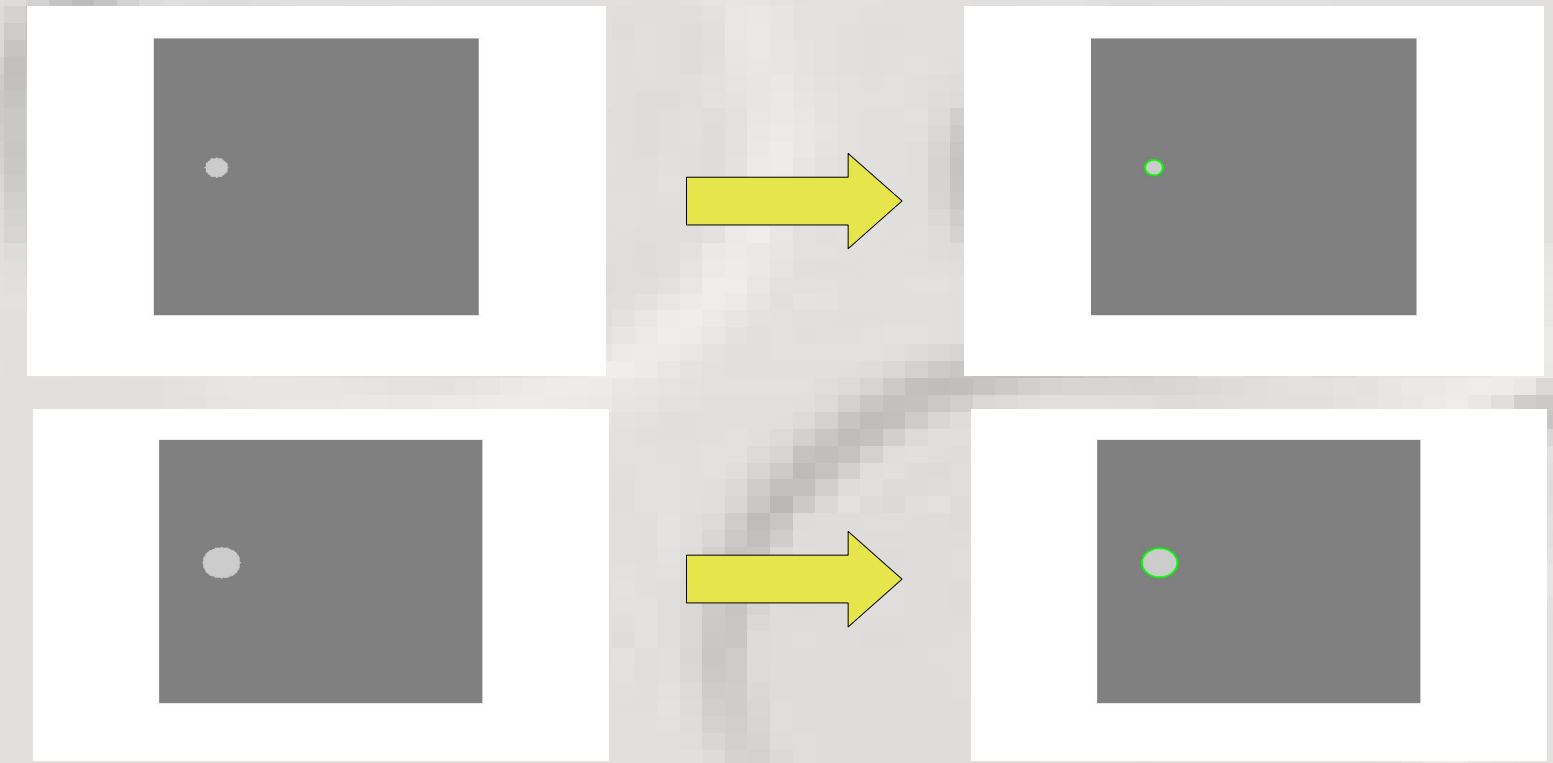
- Un terme d'attache aux données (la moyenne des intensités)
- Un terme d'attache au modèle (pénalisant les petites taille)
- Un poid différent pour chaque composante. En prenant un des deux a 1, on en supprime un...

Soit : pour Cr(x0,y0) un cercle de rayon r, centré en x0,y0 :

$$J(x0, y0, r) = \sum_{(x,y) \in Cr(x0,y0)} [s(x,y)] + \alpha * r$$

# Terme de Pénalité

Résultats : $\alpha = 0.0145$



Problème : choisir $\alpha$ (m'a pris 20mn et non généralisable...)

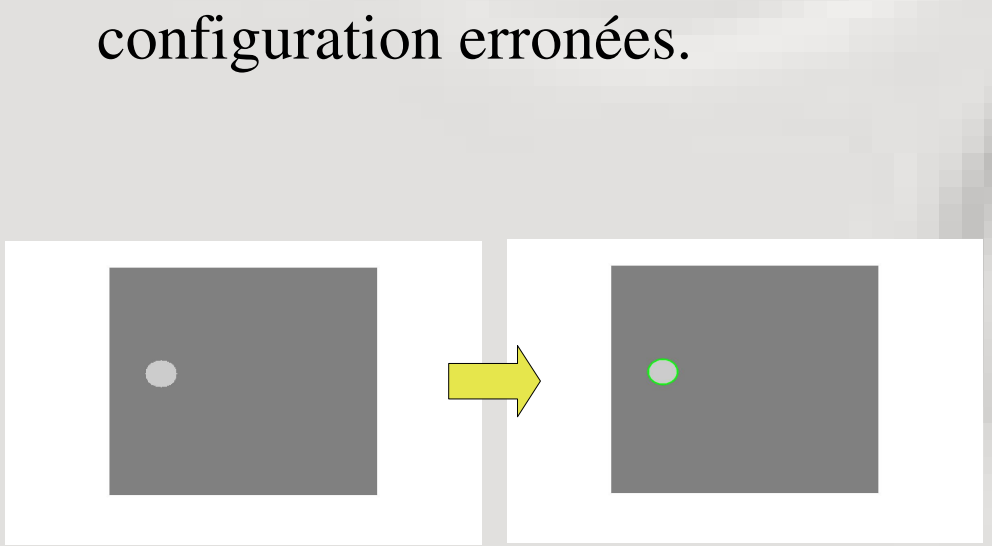
# Localisation v3.0

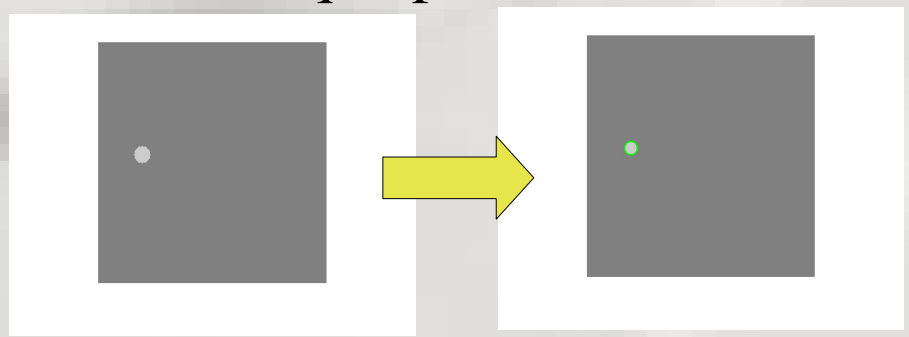
Si l'on reprend l'exemple précédent, il fallait prendre en compte la présence de l'extérieur de la cible.

Un critère qui fonctionne serait : m_I - m_O
avec

- m_I : moyenne des pixels a l'interieur de la cible testee,
- m_O : moyenne des pixels a l'exterieur de la cible testee...

Ici, pas de pénalisation, simplement un critère qui pénalise tout seul les configuration erronées.





# Améliorations ?

Peut on faire mieux ? Dis autrement :

- Pourquoi ne pas prendre m_I^2 - m_O^2 ???
- Ou pourquoi ne pas essayer de detecter les bords du disque (cf Desachy) pour trouver le cercle (ou encore appliquer la transformee de Hough) ???

En soi, cette question n'a aucune réponse valable : On ne peut améliorer un algorithme que lorsque l'on sait (au moins a peu près) à quels types d'images il va se frotter pour pouvoir le tester (et éventuellement concevoir l'algorithme le plus adapté a ces cas...)

Les critères précédents etaient empiriques... on ne peut les différencier que sur des exemples précis... et on serait bien en peine de définir un algorithme qui marche le mieux possible pour tous les cas !

# Ajout de bruit

Par exemple, il est rare que

- L'intensite sur la cible soit constante..
- L'intensité sur le fond soit constante...

Testons notre dernier algorithme en « saupoudrant » l'image d'un léger bruit :

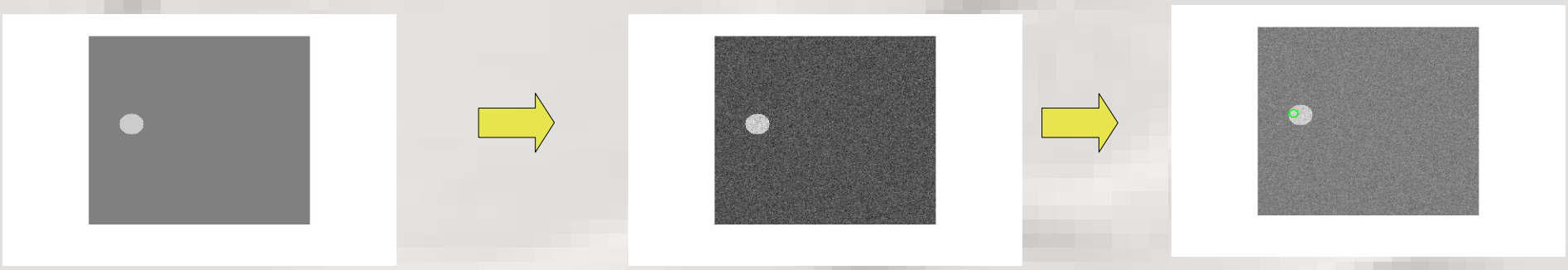


Image parfaite | Image observée | résultat

Raté....

# Notion de Modèle

Ici, nous quittons le domaine du bricolage pour définir un algorithme adapté a nos images : On se donne le modèle d'image suivant :

L'image est constitué de deux parties :

- une cible dans laquelle chaque pixel est une réalisation d'un bruit blanc gaussien, de moyenne mc, et d'ecart type s. La cible est toujours un rond.
- Le fond dans lequel chaque pixel est une réalisation d'un bruit blanc gaussien, de moyenne mf, et d'ecart type s (le meme que pour la cible)

Nous allons maintenant nous interesser à la localisation optimale de la cible pour ce modèle...

La probabilité d'observer l'image que l'on a sous les yeux si une cible de rayon r se trouve a une position i,j est :

$$P(s/x,y,r)=\prod_{(x,y)\in d(x,y,r)}\frac{1}{\sqrt{2\pi s}}e^{\frac{(x-mc)^2}{2s^2}}\cdot\prod_{(x,y)\notin d(x,y,r)}\frac{1}{\sqrt{2\pi s}}e^{\frac{(x-mf)^2}{2s^2}}$$

# Notion de Modèle

Ne reste plus qu'a chercher les x,y,r qui rendent cette probabilité la plus grande possible.
Pour cela :

- Pour simplifier les calculs, on prend le logarithme de cette quantité.
- On arrange un peu tout ca.... (Vu en cours)
- On remplace les moyennes par leurs estimées

Ce qui nous donne le critère suivant :

$$J(x, y, r) = N_I . m_I^2 + N_O . m_O^2$$

Ce critère a l'avantage d'être justifié théoriquement et nous assure la meilleure probabilité de localisation correcte tant qu'on reste dans le modèle (ce qui ne veut pas dire que notre algorithme ne se trompera jamais...)

# Autres problèmes de localisation

Dans ce qui précédait, on cherchait une cible uniforme, sur un fond uniforme, en présence d'un bruit blanc gaussien...Problème pour lequel nous avons vu différents critères avant d'en dégager un qui corresponde au modèle.

Voici quelques autres problèmes classiques de localisation. Pourriez vous trouver un couple critère – optimisation pour traiter ces problèmes ?

- Les niveaux de gris de la cible sont connus, le fond est nul. L'image est bruitée avec du bruit blanc gaussien.
- Les niveaux de bruit de la cible sont connus. L'illumination générale de l'image peut changer.
- Les niveaux de bruit de la cible sont connus, l'orientation de la cible est inconnue. L'illumination générale de l'image peut changer.

# Localisation et détection

Dans la pratique, il s'agit souvent non pas de localiser une cible, mais de détecter sa présence (et de la localiser par la même occasion).

Les deux problèmes sont donc relativement liés.

Si l'on peut parfois trouver une approche optimale du problème en utilisant un modèle (et des tests statistiques), le plus souvent ce problème est réglé de la façon suivante :

- On calcule la valeur du critère en chaque point.
- Si cette valeur est supérieur à un seuil, on décide de la présence d'une cible.

Que se passe-t-il le plus souvent pour les pixels voisins de la cible ?

# Recalage

Le problème du recalage d'image est également un problème classique : une « même » scène est capturée sous deux formes différentes. Il s'agit de trouver la fonction qui transforme au mieux une image en l'autre.

Exemples :

- Deux appareils photo prennent chacun une photo de la même scène sous deux angles différents au même moment (stéréoscopie)
- Deux satellites prennent une photo du même endroit sous des résolutions différentes et sous des angles différents (Google Earth ...)
- On observe l'évolution d'un cancer chez un patient a deux instant différents.
- On observe simultanément une radio et un scanner d'un patient pour recouper ces informations.

# Modélisation du problème

Il s'agit de trouver la transformation qui transforme chaque pixel de l'image en son correspondant dans l'autre image :

- Soit x,y les coordonnées d'un pixel dans l'image 1.
- Soit x',y' les coordonnées du pixel correspondant dans l'image 2.
- On cherche la fonction h qui a (x,y) associe (x',y').

h est bijective. On peut donc calculer la transformée de I1 en chaque pixel grâce à la formule suivante :

$$\forall (x,y)\ \ I'_1(h(x,y)) = I_1(x,y)$$

Ou encore :

$$\forall (x',y')\ \ I'_1(x',y') = I_1(h^{-1}(x,y))$$

# Critère de recalage

Supposons que lorsque les images sont parfaitement superposées, les intensités sont égales.

Dans ce cas, on pourraite chercher la transformation h telle que :

$$J(I_1, I_2, h)=0 \quad avec \quad J(I_1, I2, h)=h(I_1)-I_2=\sum_{(x',y')} I_1(h^{-1}(x',y'))-I_2(x',y')$$

Mais notre solution ne correspond pas à un minimum de J.

Un critère classique est celui de l'**erreur quadratique** :

$$J(I_1, I_2, h) = \sum_{(x',y')} (I_1(h(x',y'))-I_2(x',y'))^2$$

Ou encore :

$$J(I_1, I_2, h) = \sum_{(x,y)} (I_1(x,y)-I_2(h(x,y)))^2$$

# Transformations géométriques classiques

Admettons que nous cherchions a minimiser l'erreur quadratique... Quel est l'ensemble des solutions possibles ?

Il s'agit de l'ensemble des transformations applicables à l'image 1.
Dans l'absolu, cet espace est extremement grand.

Nous allons donc voir quelques transformations géometriques classiques, de façon a ne chercher les solutions qu'au sein de ces transformations.

Ceci réduira l'espace de recherche, ce qui permettra parfois (rarement) de faire une recherche exhaustive de la solution.

# Translation

Il s'agit de l'ensemble des fonctions de $R^2$ dans $R^2$ telles que :

$$X=(x_{0,}x_1)\rightarrow X'=(x'_{0,}x'_1)$$
$$X'=X+A \quad avec \quad A=(a_{0,}a_1)$$

Il faudrait donc tester tous les A possibles (en restant dans le cadre de l'image)

# Rotation

Il s'agit de l'ensemble des fonctions de $R^2$ dans $R^2$ telles que :

$$X=(x_1,x_2)\rightarrow X'=(x'_1,x'_2)$$

$$X'=R.X \quad avec \quad R=\begin{bmatrix}\cos(\theta) & \sin(\theta)\\ -\sin(\theta) & \cos(\theta)\end{bmatrix}$$

Il faudrait donc tester tous les angles possibles (en restant dans le cadre de l'image)

Ceci est une rotation centrée sur le point 0,0.

Une rotation centrée sur le point C s'écrit : $X'=R.[X-C]+C$

# Zoom

Il s'agit de l'ensemble des fonctions de $R^2$ dans $R^2$ telles que :

$$X=(x_{0,}x_1)\rightarrow X'=(x'_{0,}x'_1)$$

$$X'=\lambda.X$$

Ceci définit un zoom centré en 0,0.

Pour obtenir un zoom différent sur chaque axe, on aurait :

$$X'=\left[x'_i\ ,\ i\in\{1..N\}\right]$$

$$x'_i=\lambda_i.x_i$$

Ou encore

$$X'=\Lambda.X$$

$$avec\quad \Lambda=\begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix}$$

Pour un zoom centré sur C... changez de repère !

Pour tester les différents zoom...il faut faire varier $\lambda$

# Transformations affines

Il s'agit de l'ensemble des fonctions de $R^2$ dans $R^2$ telles que :

$$X=(x_0, x_1) \rightarrow X'=(x'_0, x'_1)$$

$$X'=A.X+B$$

Ces fonctions généralisent toutes les précédentes...Et laisse des parallèles, ...parallèles...

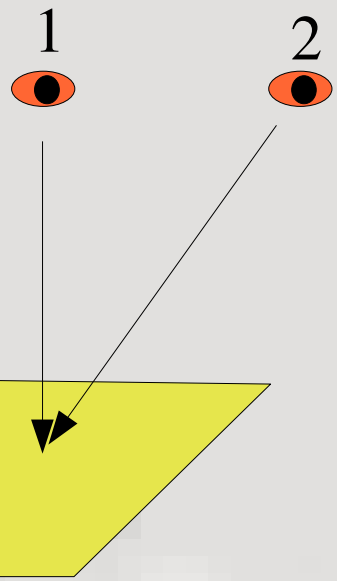
Il y a ici (en dimension N) : $N^2+N$ paramètres

(si la transformation est centrée en 0. En ajoutant les coordonnées du centre, cela en fait : $N^2+2N$ )

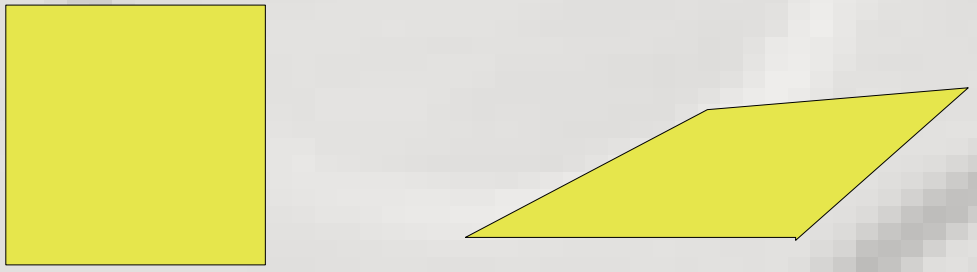
Soit 6 paramètres ou 8 pour une image en 2D.

# Transformations projectives

Imaginons une scène plane, vue sous deux angles différents.




Vu des observateurs 1 et 2, le même carré aura les formes suivantes :

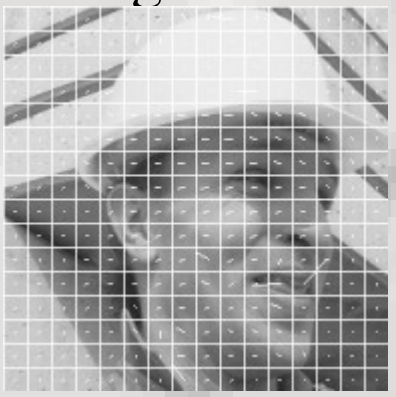


Les transformations affines n'incluent pas ce type de transformations (comment le démontrer ?). Il faudra avoir recours aux transformations projectives ou homographies...
Ce qui nous emmènerait un peu trop loin.

# Transformations non rigides

Enfin, dans le cadre du suivi de transformations temporelles, il est courant que les images ne puissent pas etre considérées comme des transformations « en bloc » l'une vers l'autre :

De fait, chaque pixel (ou bloc de pixels) se déplace à une autre position, comme illustré sur la figure suivante.



L'espace des solutions est énorme (N! Si N est le nombre de pixels des images). De plus, il faut régulariser les solutions pour trouver un champ de mouvement cohérent...

# Algorithmes d'optimisation classiques

Dans les exemples précédents (localisation), il était possible de faire une recherche exhaustive de la solution.
Dans ce problème ci, pour peu que l'espace des paramètres soit de grande dimension, cette solution est inapplicable.
On a alors recours a des algorithmes d'optimisation approchés.

- Descente du gradient (trouve un minimum local en partant d'un jeu de paramètres initiaux)... et ses nombreuses variantes...
- Recuit simulé et variante à température nulle.
- Algorithmes génétiques.
- .....

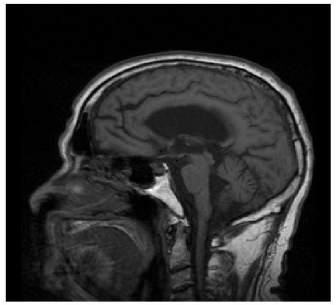
Dès lors, si la méthode ne fonctionne pas dans la pratique, ce peut être parce que :

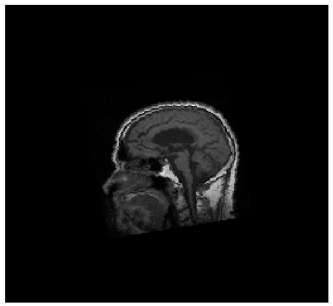
- Le critère est inadapté.
- L'algorithme d'optimisation trouve un minimum local.

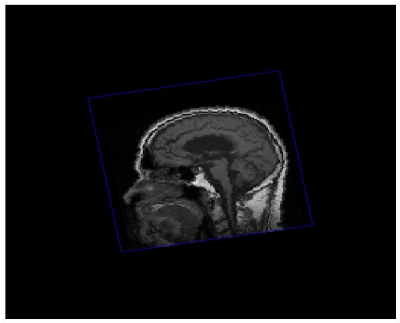
# Exemples d'application

Ici : on cherche la rotation transformant l'image 1 en l'image2

- Le centre de rotation est connu.
- Le facteur de zoom est connu.





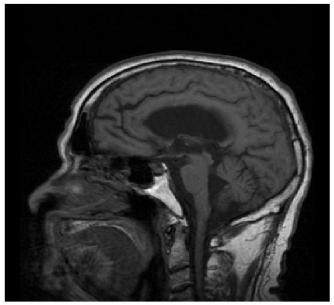


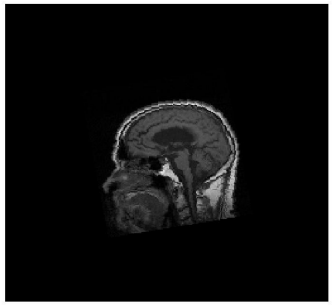
L'optimisation teste tous les angles...

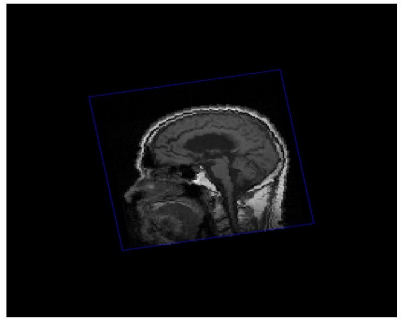
# Exemples d'application

Ici : on cherche la rotation transformant l'image 1 en l'image2

- Le centre de rotation est connu.
- Le facteur de zoom est INCONNU.





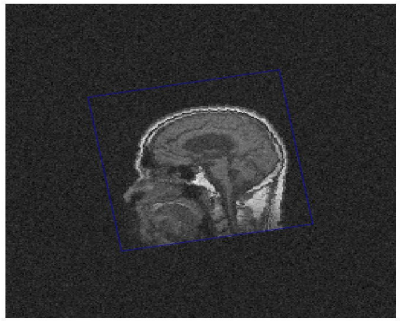


On teste tous les angles et tous les facteurs de zoom...

# Exemples d'application

Ici : on cherche la rotation transformant l'image 1 en l'image2

- Le centre de rotation est connu.
- Le facteur de zoom est INCONNU.
- Les deux images sont entachées de bruit gaussien moyenne nulle.

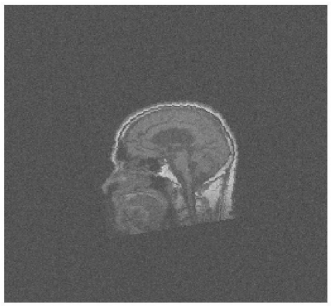


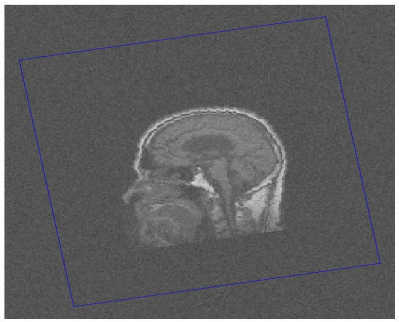
Pourquoi cela fonctionne-t-il ?

# Exemples d'application

Ici : on cherche la rotation transformant l'image 1 en l'image2

- Le centre de rotation est connu.
- Le facteur de zoom est INCONNU.
- Les deux images sont entachées de bruit...
- L'image 2 a une moyenne differente...



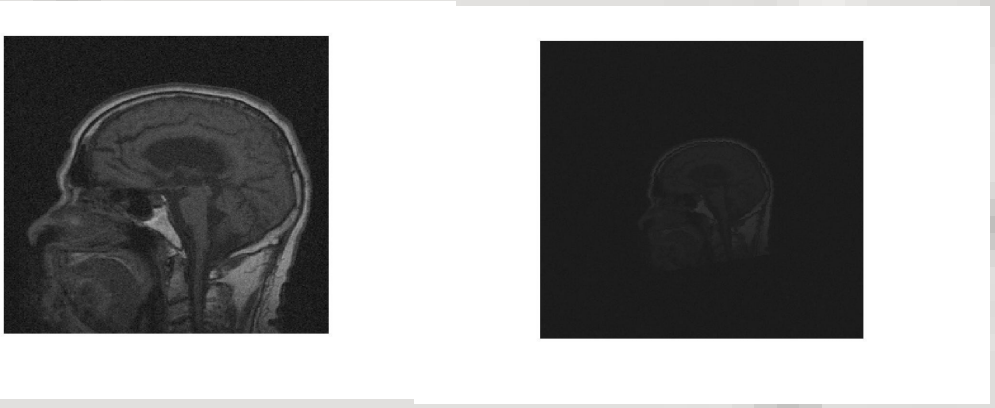


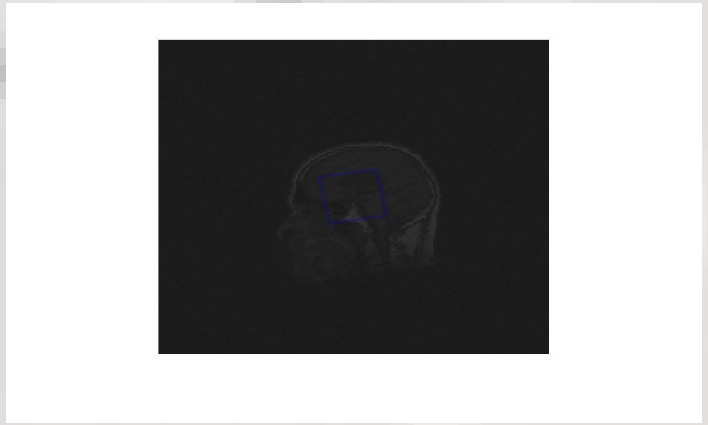
Pourquoi cela ne fonctionne-t-il pas ?

# Exemples d'application

Ici : on cherche la rotation transformant l'image 1 en l'image2

- Le centre de rotation est connu.
- Le facteur de zoom est INCONNU.
- Les deux images sont entachées de bruit...
- Le gain est different sur les deux images (...)





Pourquoi cela ne fonctionne-t-il pas ?

# Exemples d'application

Ici : on cherche la rotation transformant l'image 1 en l'image2

- Le centre de rotation est connu.
- Le facteur de zoom est INCONNU.
- Le facteur de gain est estimé (...)
- Les deux images sont entachées de bruit...

Comment faire ?

# Autres méthodes de recalage

Le problème, on l'a vu est de faire l'exploration de l'espace de recherche pour trouver le minimum.

Il existe d'autres méthodes pour faire l'économie de cette exploration (ou l'accélerer. Elles sortent de notre contexte (critère et optimisation) et ne sont citées içi que pour votre culture.

Ces méthodes, pour la plupart, sélectionnent des points d'intérêt dans chacune des deux images (par exemple, des points de fort gradient ou des coins ou....) puis cherchent les paramètres des transformation qui permettent d'apparier au mieux ces points.

# Segmentation

Dans ce cas, il s'agit de découper l'image en zones sensées représenter des objets. Comme nous sommes dans le cas de traitement bas niveau, il s'agira exclusivement de découper l'image en zones homogènes.

Une zone sera considéree homogène si un paramètre mesuré en différents pixels de la zone ne varie que peu.

Voici quelques exemples de paramètres (non exhaustifs !)

- L'intensité du pixel
- Son gradient
- La variance sur un voisinage du pixel
- La valeur de la transformée en ondelette de l'image en ce point à différentes échelles
- ...

# Segmentation à deux classes

En considérant le problème comme un problème de minimisation, il nous faut un critère a minimiser. Pour simplifier, interessons nous au problème d'une cible unique placée sur un fond.

Supposons par exemple que l'image suive le modèle posé dans la partie localisation (du bruit blanc gaussien de moyenne différente sur l'objet et sur le fond)

Cette fois ci, il nous faut trouver sa forme (W).

=> Explorer l'ensemble des formes de cibles possibles et choisir celle qui minimise le critère.

Un critère adapté a notre problème serait : $J(W_{test}) = N_I . m_I^2 + N_O . m_O^2$

# Contours actifs...

L'espace des formes possibles étant énorme, nous allons adopter une méthode d'optimisation de type descente du gradient.

Pour cela, il nous faut :

- une situation initiale (une forme initiale)
- définir ce qu'est une variation de cette forme.

On pourra par exemple utiliser une approximation polygonale de notre forme. Une déformation de la forme pourrait ainsi consister a déplacer un des sommets du polygone.

Un algorithme simple a implémenter est le suivant :

poly <- polygone initial;

while (N< seuil ou poly ne change plus depuis longtemps)

tirer un sommet k au hasard

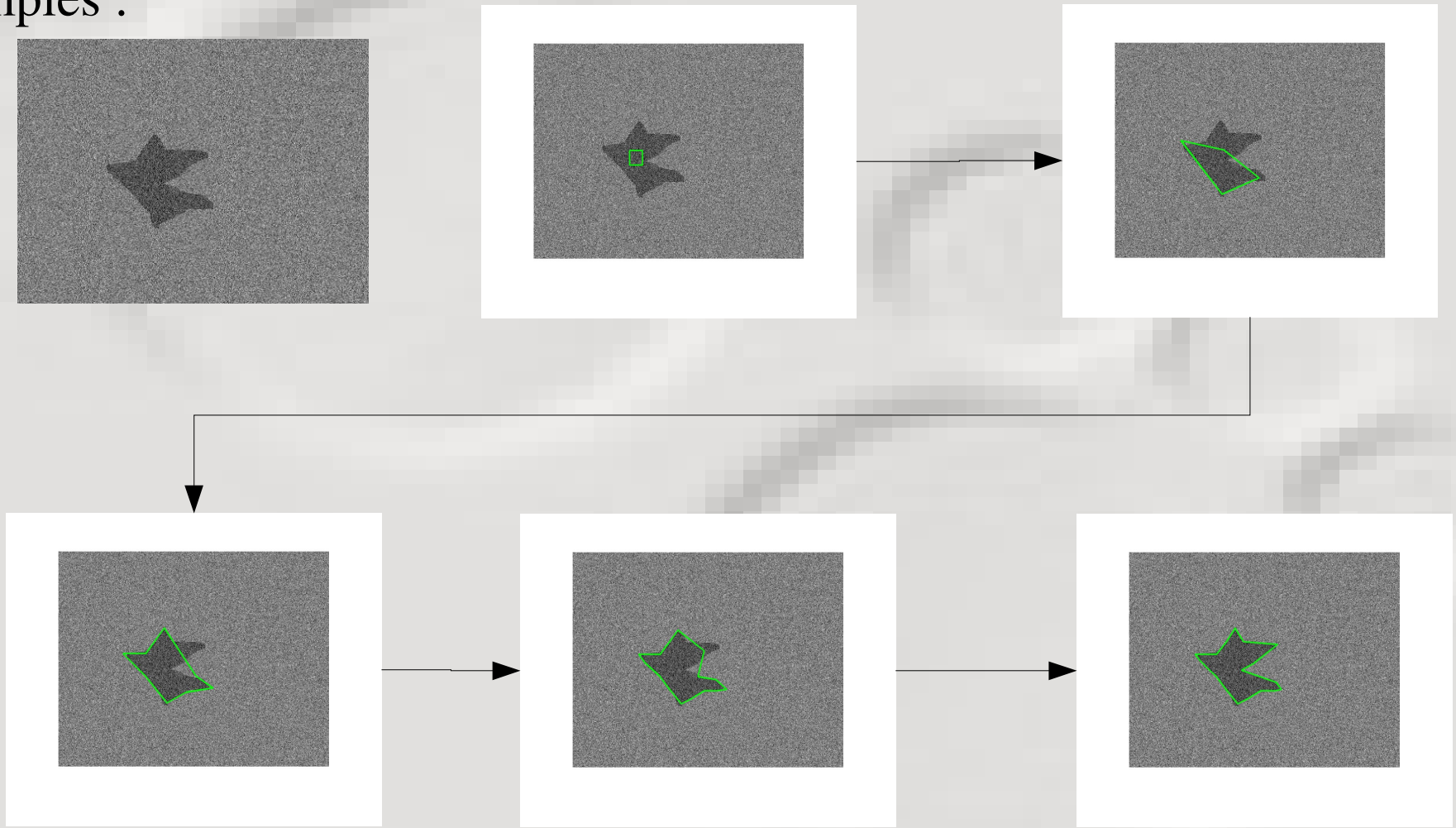
tirer un déplacement de ce sommet au hasard et construire ainsi polytest.

Si J(polytest) < J(poly) alors poly <- polytest

# Contours actifs...

Ce nom de contours actifs est due au fait que notre courbe (le polygone) va se déformer petit à petit pour venir segmenter l'objet...

Exemples :



# Contours actifs originaux

De facon générale, les contours actifs cherchent a minimiser un critère qui prend la forme suivante pour une forme testée C et une image s :

$$J(C) = \alpha . E_{interne}(C) + \beta . E_{Externe}(C, s)$$

- L'énergie externe représente l'attache aux données : l'adéquation du snake avec ce que l'on recherche dans l'image. Dans la version originale des snakes, le contour actif devait se positionner sur les contours présents dans l'image. Pouvez vous trouver une energie externe adaptée ?
- L'énergie interne représente l'attache au modèle : c'est un terme pénalisant les formes non désirées (points non regulierements répartis, courbure trop grande, force ballon....)

Dans le modèle de snake proposé dans les diapos précédentes, que sont les énergies internes et externes ?
Definissez les autres forces proposées ...