Introduction à la vision artificielle

Traitement des images

Patrick Hébert

Génie électrique et génie informatique

Université Laval

**collaboration Éric Samson*

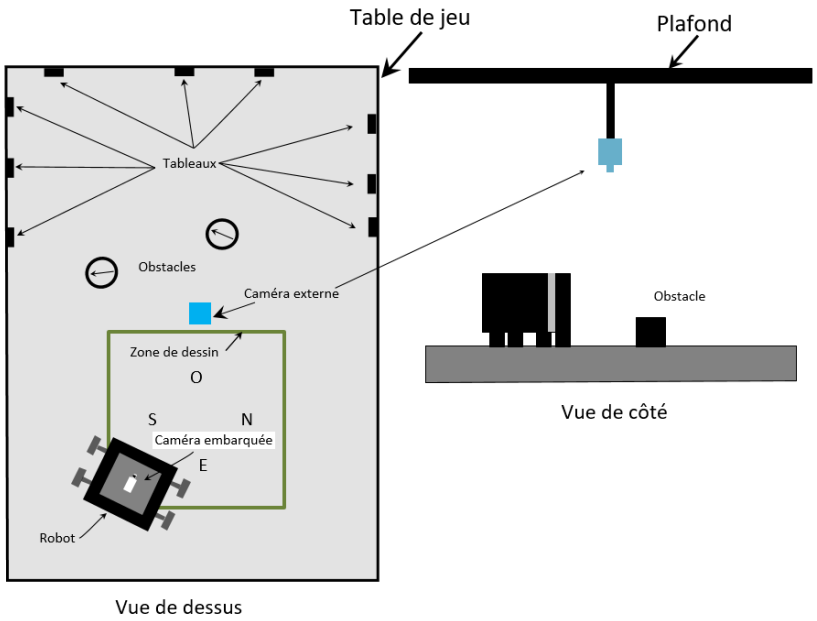
(Revisité par A. Bendada en 2013 et D. Laurendeau en 2017

Introduction à la vision artificielle

Traitement des images

Mise en contexte dans le cadre du cours de Design III

Mise en contexte…



➢ **Qu'est-ce que la caméra *embarquée* permet au robot de faire ?**

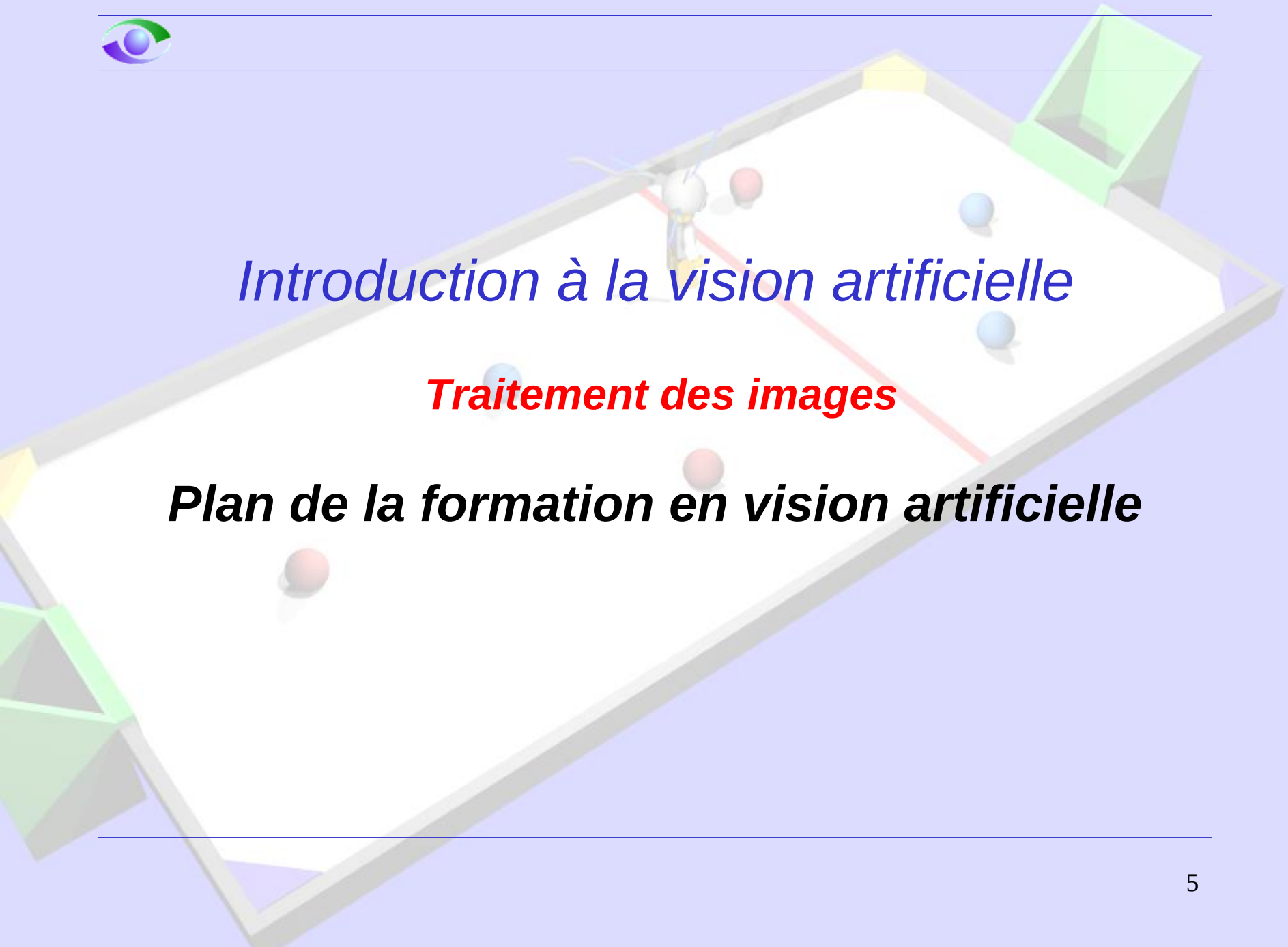
- Reconnaître les objets sur la table. *(Tableaux, obstacles, zone de dessin, etc.)*
- Raffiner l'estimation de la position des objets sur la table pour l'évitement de collision et la planification de trajectoire.

➢ **Qu'est-ce que la caméra *externe* permet de faire ?**

- Estimer la position et l'orientation du robot sur la table et planifier la trajectoire
- Estimer la position (et l'orientation) des objets sur le terrain (obstacles, zone de dessin, etc.)

- Ces deux étapes s'inscrivent dans le cycle global "perception/action" du processus de vision

1. **Perception initiale de l'environnement**
   Formation des images/acquisition
2. **Traitement des images et reconnaissance**
   Repérer les objets de l'aire de travail dans l'image
3. **Analyse des données**
   Calculer la position des objets et du robot
4. **Planification et prise de décision**
   Déterminer la prochaine commande à transmettre au robot
5. **Exécution**
   Transmission de la commande au robot



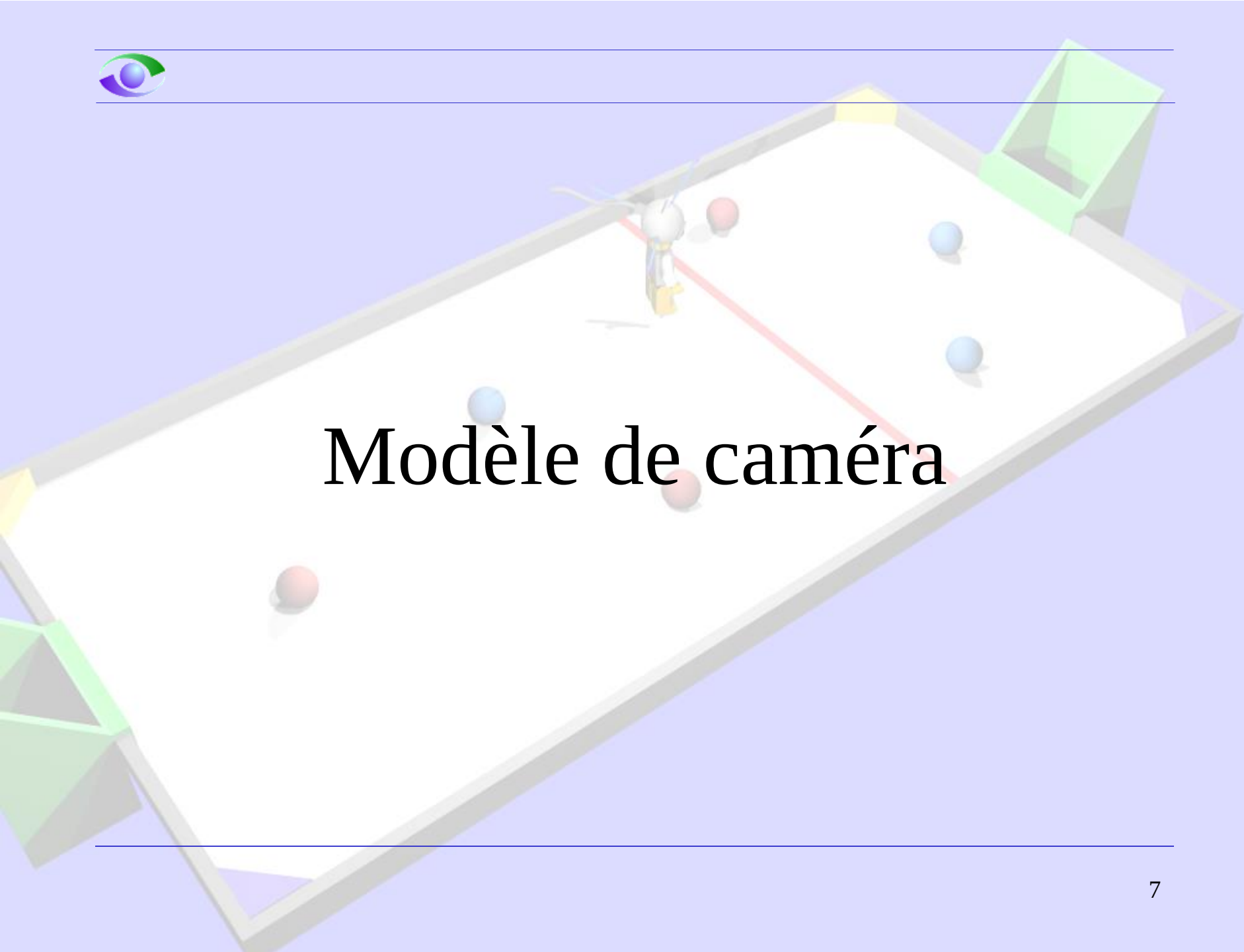
Introduction à la vision artificielle

Traitement des images

Plan de la formation en vision artificielle

Plan de la présentation

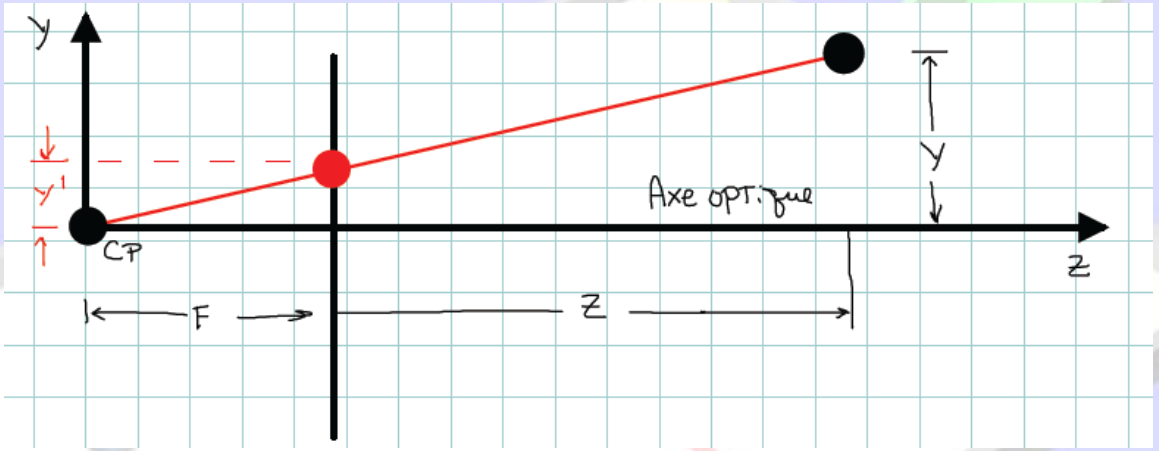
- Modèle de caméra
- Espaces des couleurs
- Traitement des images : Introduction
- Filtrage
- Segmentation
  - Extraction de régions
  - Extraction de contours
  - Détection de coins
- Conclusion



Modèle de caméra

À 1 référentiel

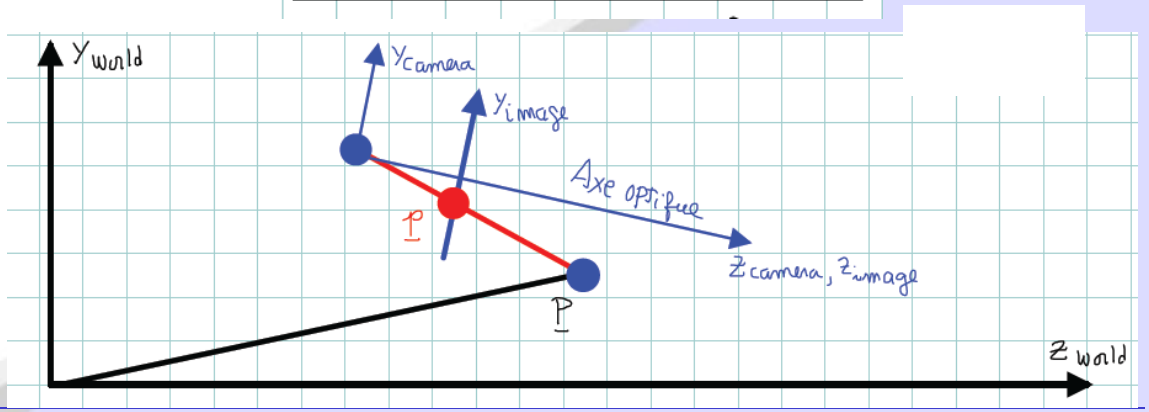
Formation d'image: sténopé (pinhole) et projection de perspective



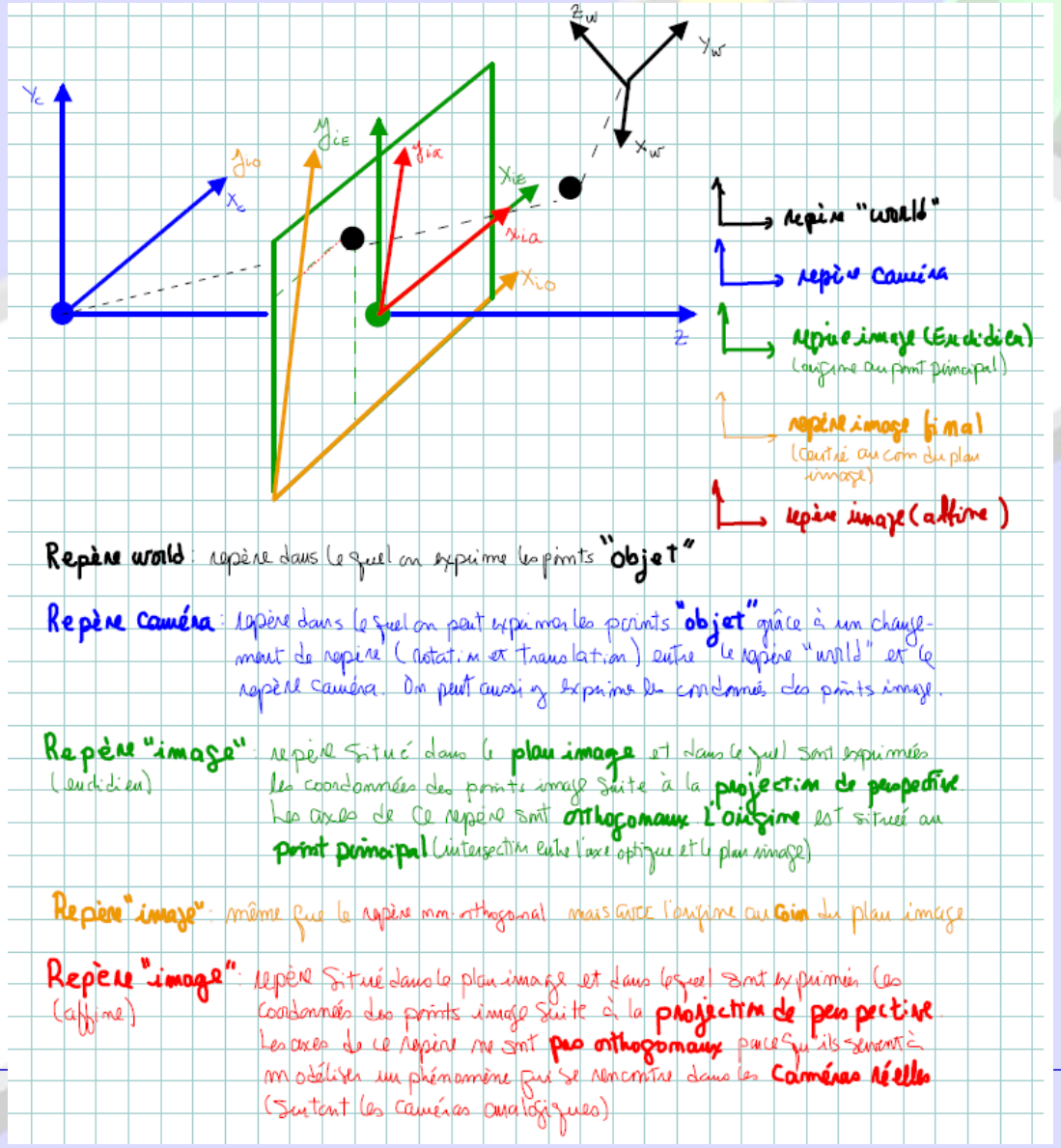
$$\frac{y'}{F} = \frac{y}{z} \quad , \quad \frac{x'}{F} = \frac{x}{z}$$

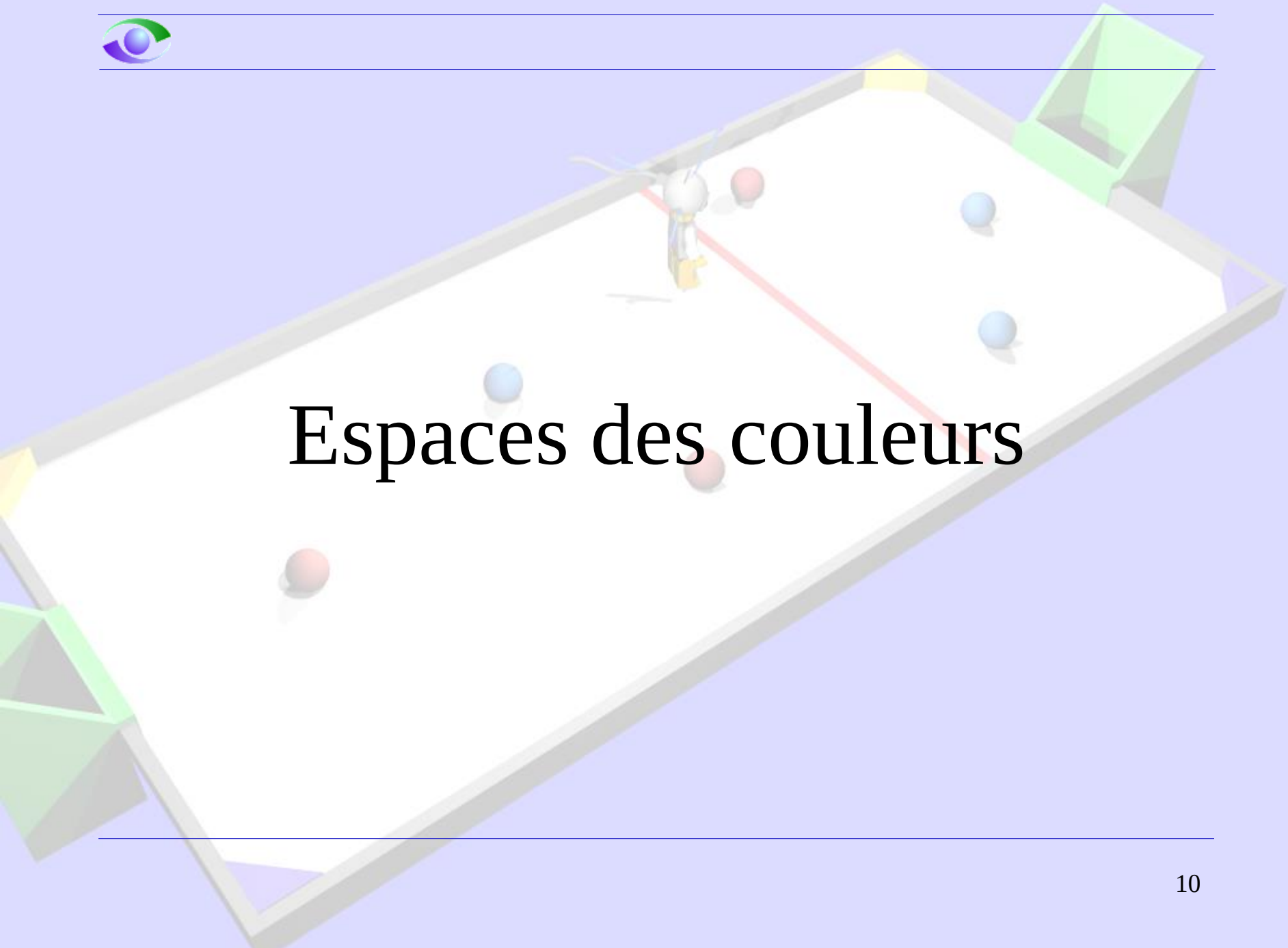
$$x' = \frac{F}{z} x \quad \text{et} \quad y' = \frac{F}{z} y$$

À 2 référentiels



Modèle complet avec tous les référentiels

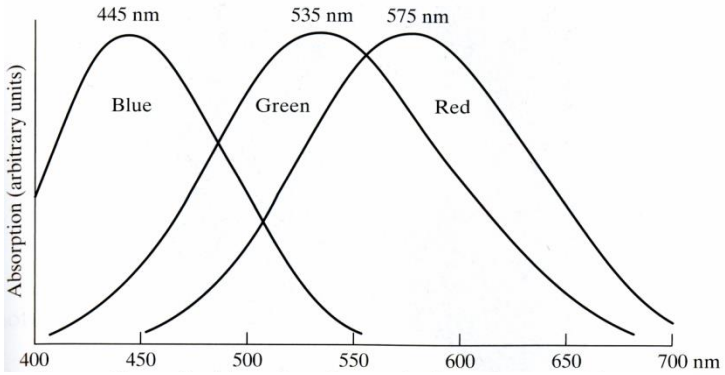




Espaces des couleurs

Espaces des couleurs…

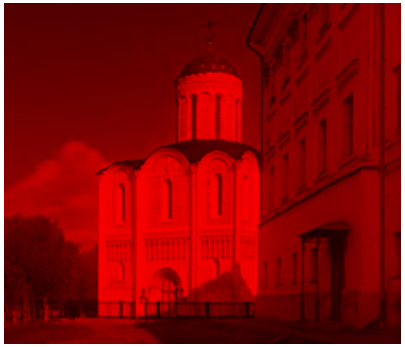
- En général, on représente la couleur à l'aide de 3 valeurs → Espace en 3 dimensions
- L'espace le plus connu est RGB (Red, Green, Blue)
  - Lié au système visuel humain
    - L'œil perçoit la couleur à l'aide de trois types de capteurs (cônes) sensibles à trois plages de longueurs d'onde.



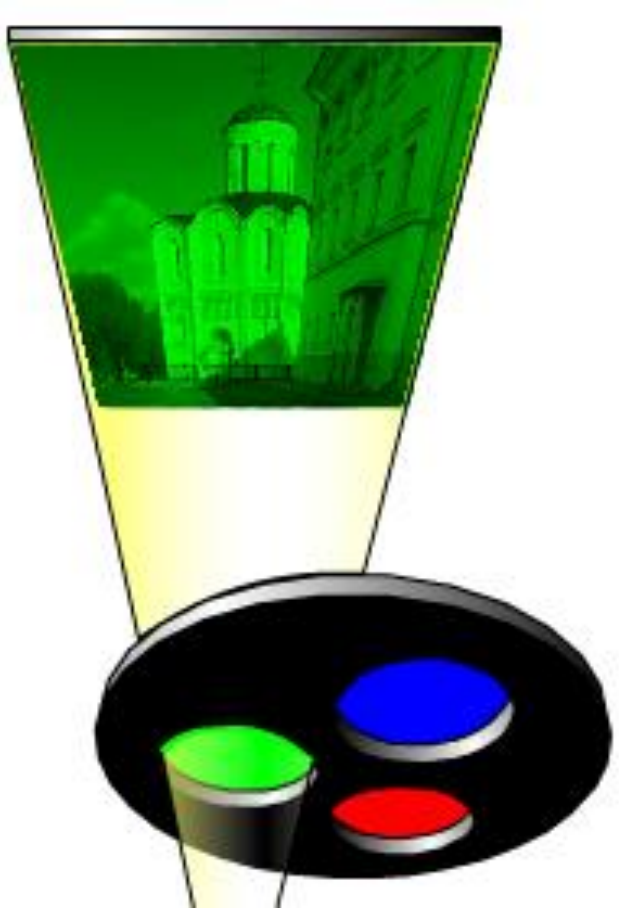
Absorption de la lumière par les cônes de l'œil humain

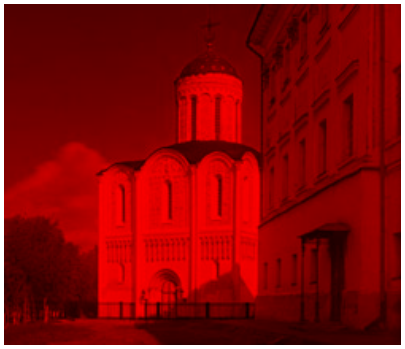
*tirée de Gonzales & Woods

Principe de la caméra couleur…

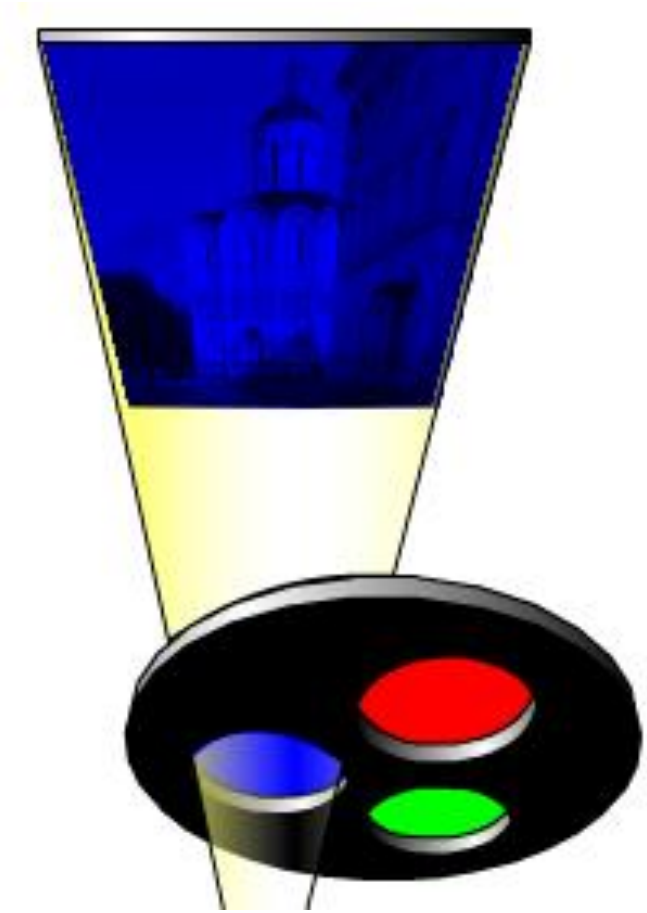


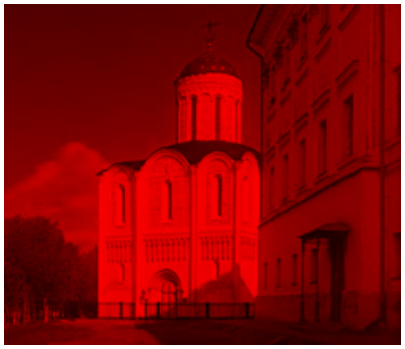
Principe de la caméra couleur…

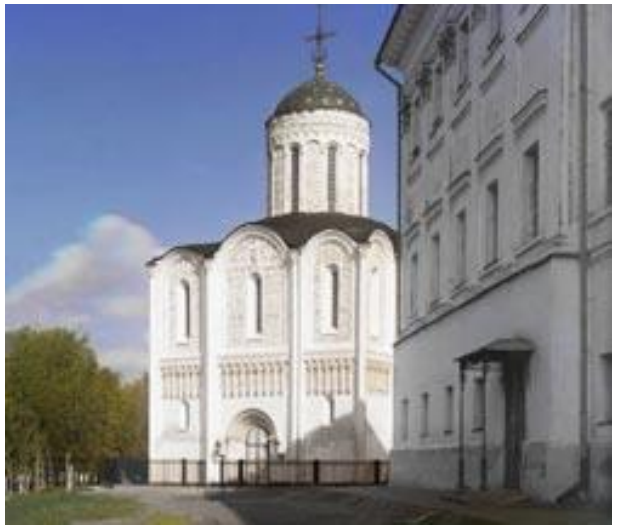




Principe de la caméra couleur…







Sortie d'une caméra couleur

Mosaïque à filtres de Bayer

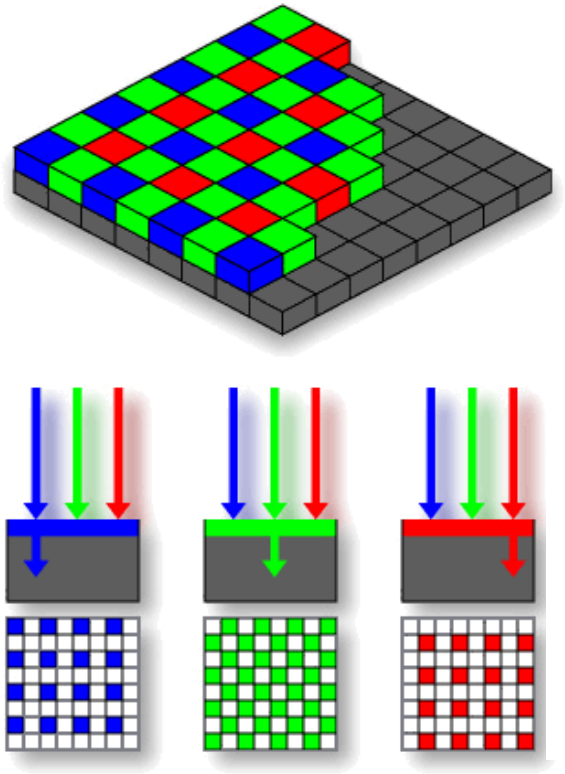
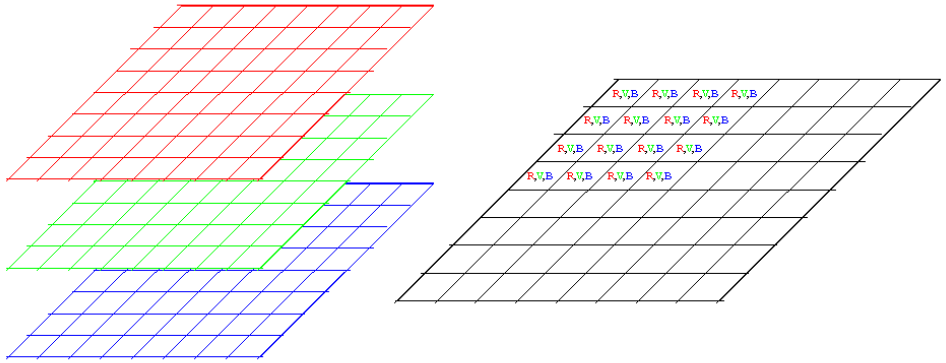


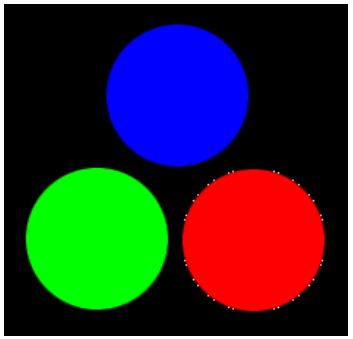
Image numérique couleur

Matrice 3D

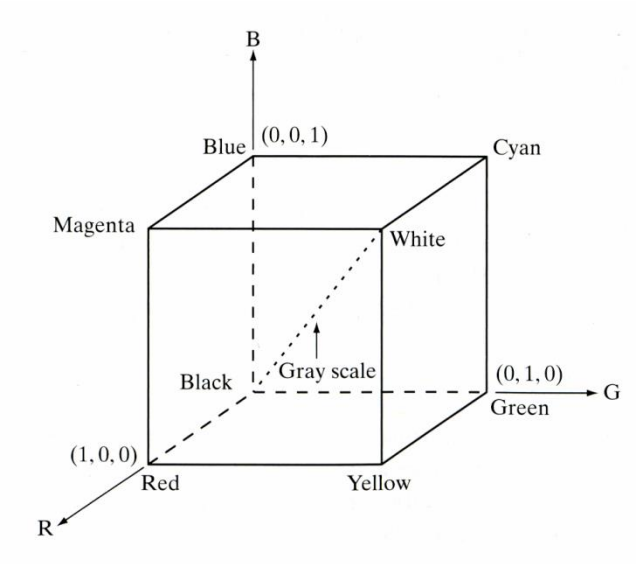


Espaces des couleurs…

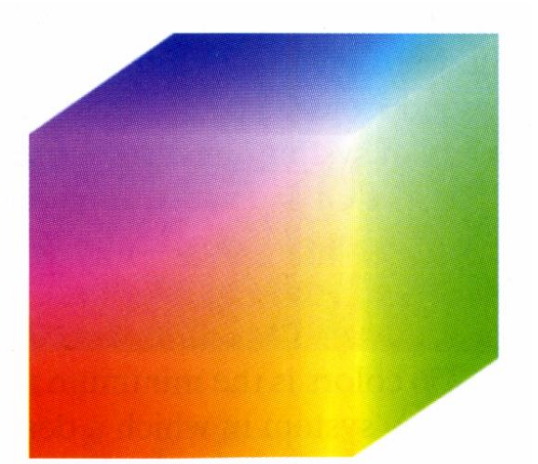
- L'espace RGB (suite)
  - Largement utilisé en informatique (moniteurs)
  - Teintes de gris → R=G=B
    le long de la diagonale



Génération de couleurs



Cube de couleurs RGB



Cube RGB 24-bits

- Problème:

  L'espace RGB n'est pas intuitif pour l'utilisateur (e.g. Photoshop)

Exemple: On veut baisser la saturation du disque orange de 50%

Ces valeurs semblent aléatoires

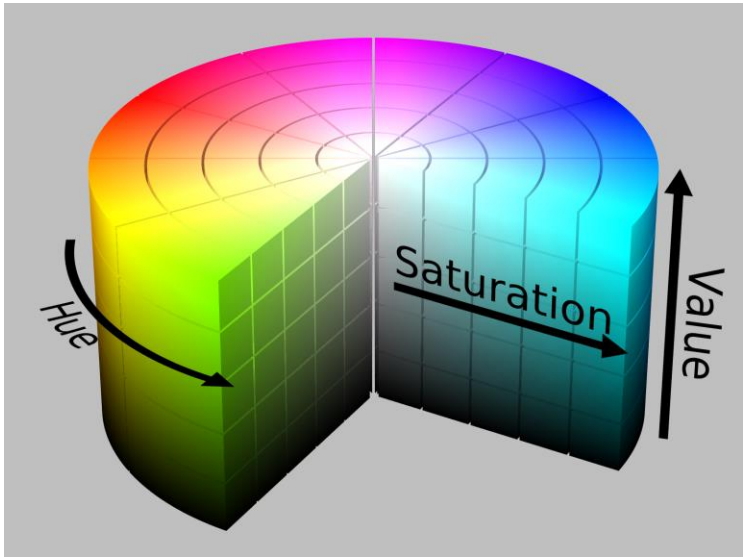
- HSV → Hue (H), Saturation (S), Value (V)
  - Séparation de la teinte, de la saturation et de l'intensité
    - Plus intuitif pour identifier et spécifier les couleurs
    - Traitement des ombrages plus facile lors de la segmentation

      Ombrage → même teinte mais intensité différente.

- V (intensité): une hauteur entre 0 (noir) et 1 (blanc)

- La teinte est contenue dans H (Hue) : un angle (0 → 360°)

- S (Saturation = pureté): un rayon entre 0 et 1:
  - S = 0 ➡ niveau de gris
  - S = 1 ➡ couleur pure

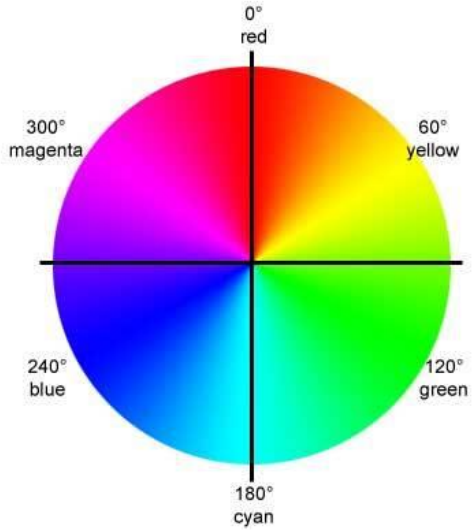


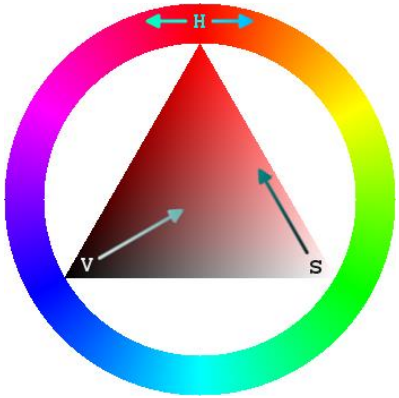
Espace HSV

- HSV

  - Le modèle HSV est particulièrement utilisé dans les applications graphiques.

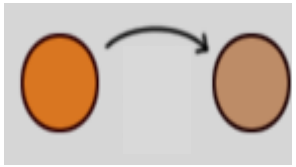
  - La teinte est sélectionnée sur la région circulaire; la saturation et l'intensité sont sélectionnées *a posteriori* sur un triangle.





Roue de couleurs HSV

Exemple: On veut baisser la saturation du disque orange de 50%



- Passage RGB → HSV : Formules

Pré-calculs

$$M = max(R,G,B)$$
$$m = min(R,G,B)$$
$$C = M - m$$

3

$$H' = \begin{cases} undefined, if\ C = 0 \\ \frac{G-B}{C} \bmod 6, if\ V = R \\ \frac{B-R}{C} + 2, if\ V = G \\ \frac{R-G}{C} + 4, if\ V = B \end{cases}$$
$$H = 60^o\ x\ H',\ with\ H \in [0^\circ, 360^\circ[$$

2

$$S = \begin{cases} 0, if\ C = 0 \\ \frac{C}{V}, otherwise \end{cases}$$

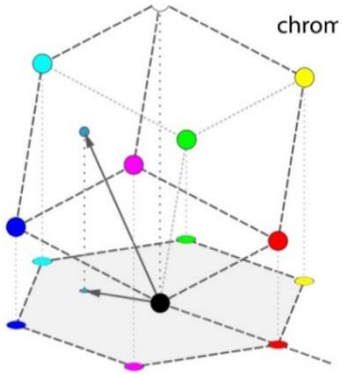
1

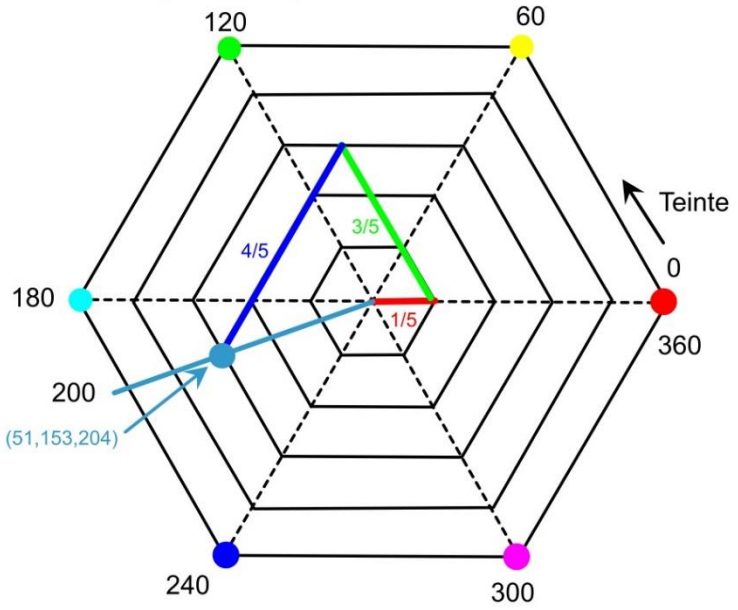
$$V = M$$

Référence sur les conversions: http://en.wikipedia.org/wiki/HSL_and_HSV

Espaces des couleurs…

Exemple avec R = 1/5, G = 3/5, B = 4/5





M = Max(1/5,3/5,4/5)= 4/5

m = Min(1/5,3/5,4/5)= 1/5

C = M - m = 4/5 - 1/5 = 3/5

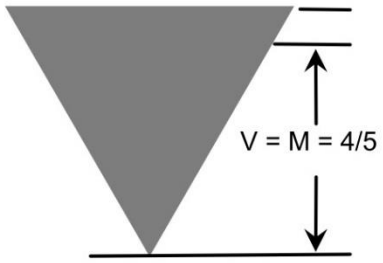
V = M = 4/5

Comme V = B = 4/5 on a:

H' = [(R - G) / C] + 4 = [(1/5 - 3/5) / 3/5] + 4 = 4 - 2 / 3 = 10 / 3

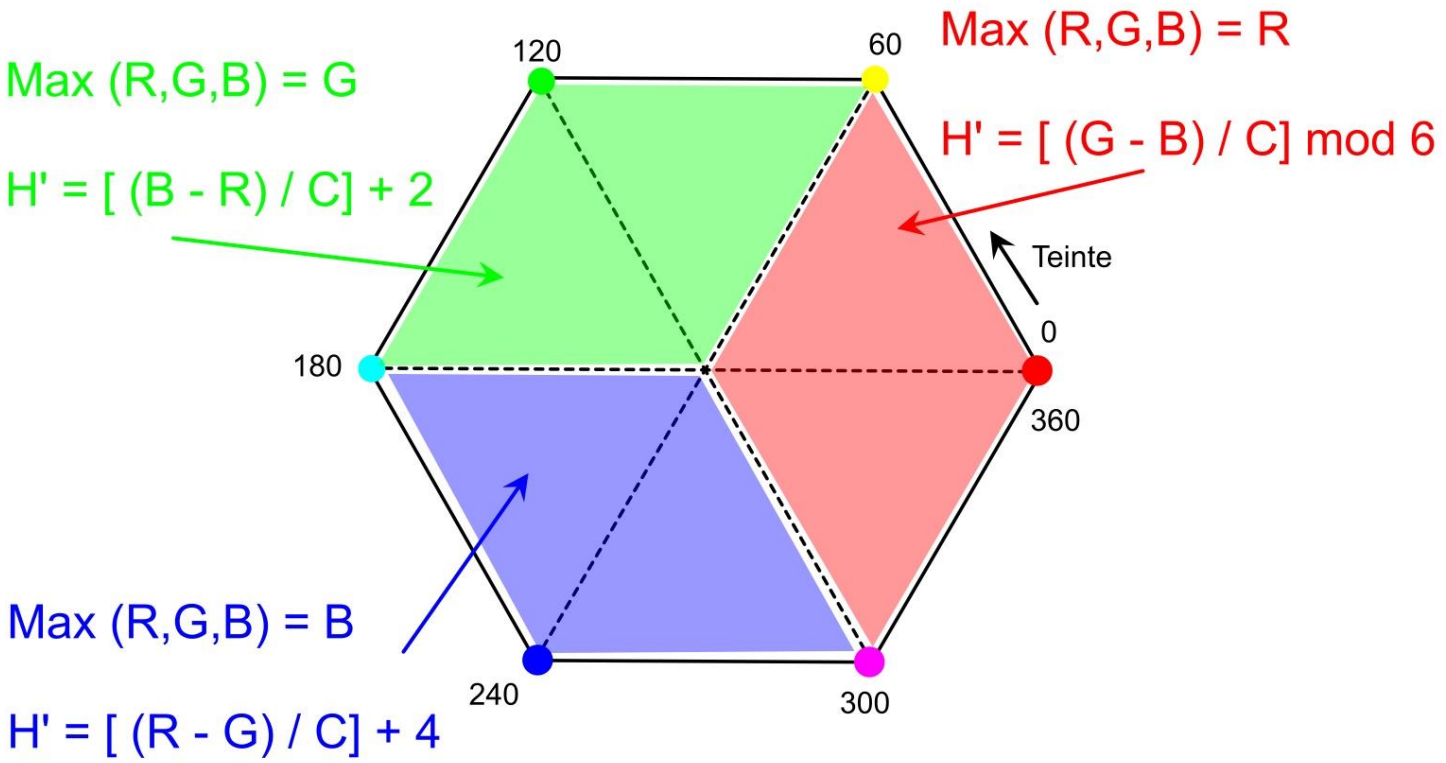
et donc :

H = 60 x 10 / 3 = 200

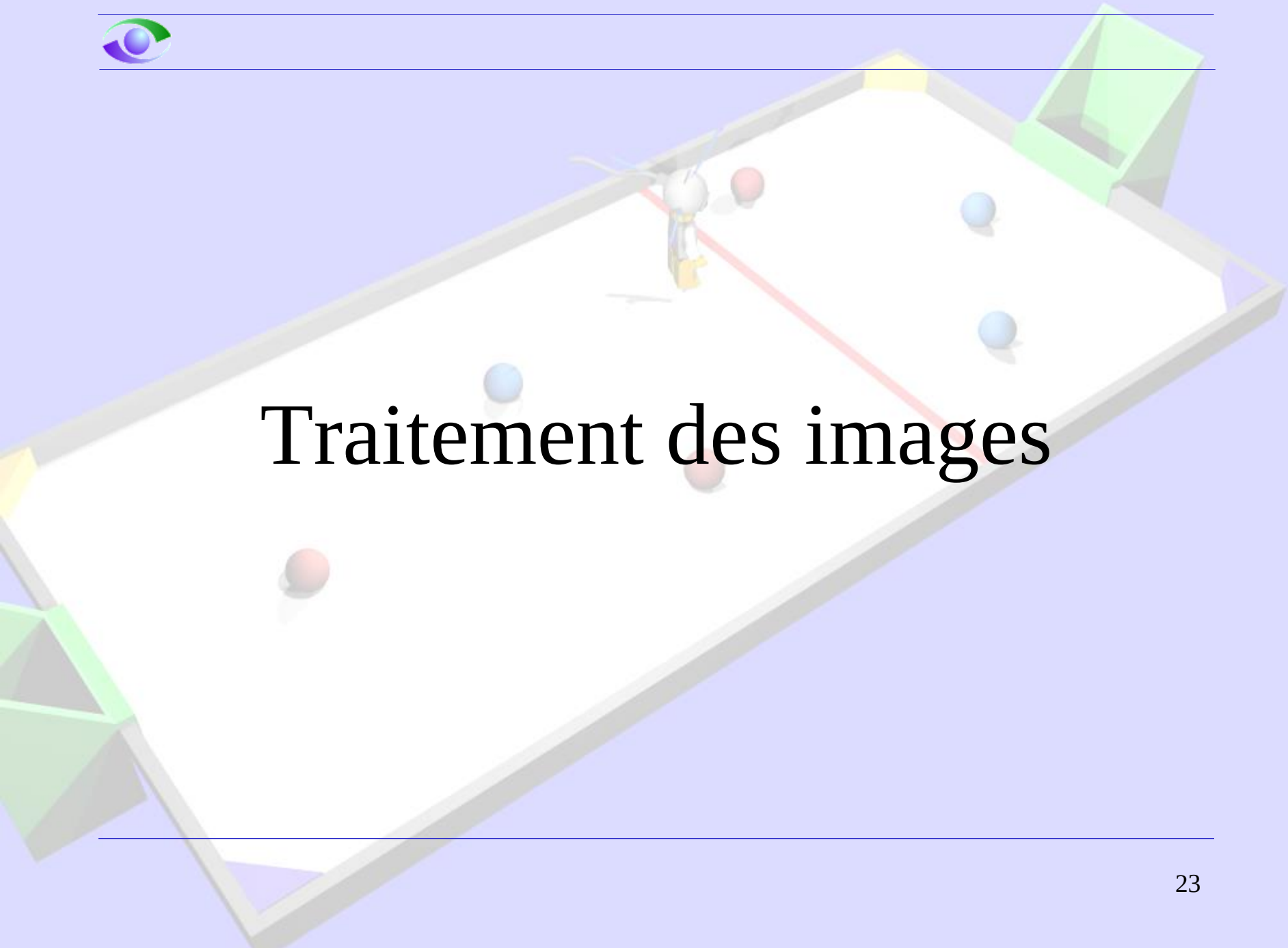


Espaces des couleurs…

Régions de calcul de la teinte (hue)



H = 60 x H'



Traitement des images

Plan de la présentation

- ✓ Espaces des couleurs
- ➢ Traitement des images : Introduction
- Filtrage
- Segmentation
  - Extraction de régions
  - Extraction de contours
  - Détection de coins
- Conclusion

Traitement des images : introduction

- Le traitement simple des images peut se faire en 2 étapes:

1. **Traitement sur les pixels**
   - *Filtrage*
   - *Regroupement en régions*
   - *Détection de contours*
   - *Détection de points d'intérêt (coins)*

2. **Interprétation**

   Reconnaissance basée sur les modèles

   Est-ce que ce groupe de pixels "jaunes" peut correspondre à une balle sphérique ou à une cible rectangulaire?

Traitement des images : introduction

- Plusieurs facteurs influencent le traitement et l'interprétation des images
  - Éclairage variable

    Heure du jour, éclairage naturel ou artificiel
  - Arrière-plan (circulation autour de la table!)
  - Points de vue différents

    Un objet peut présenter des formes très différentes selon le point de vue
    - *La simple projection d'un espace 3D à 2D pour un objet opaque*
    - *La réflectance des matériaux*
  - Bruit du capteur
  - ...

- Pour éviter des résultats indésirables (*e.g. fausse détection, manque*), il faut que le traitement et l'interprétation soient robustes aux variations des facteurs hors de votre contrôle

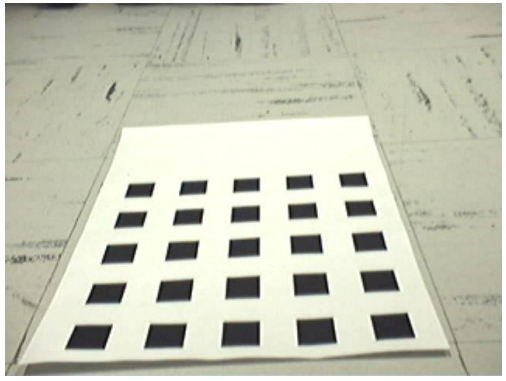
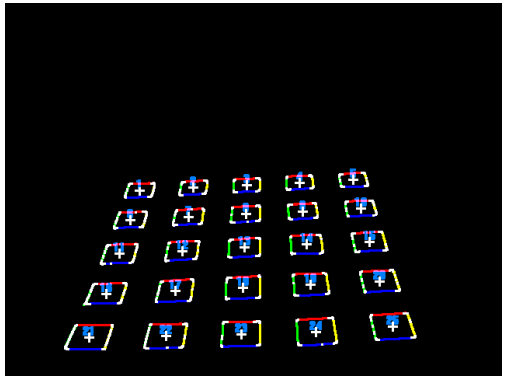
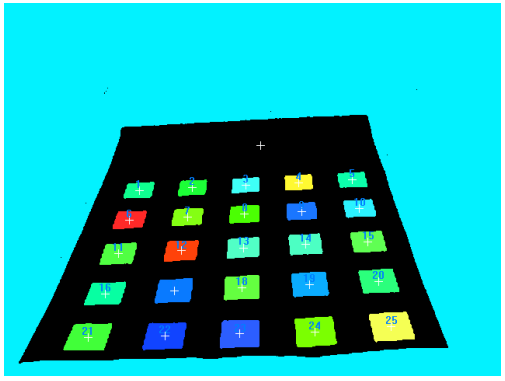
  - Robustesse au niveau du traitement

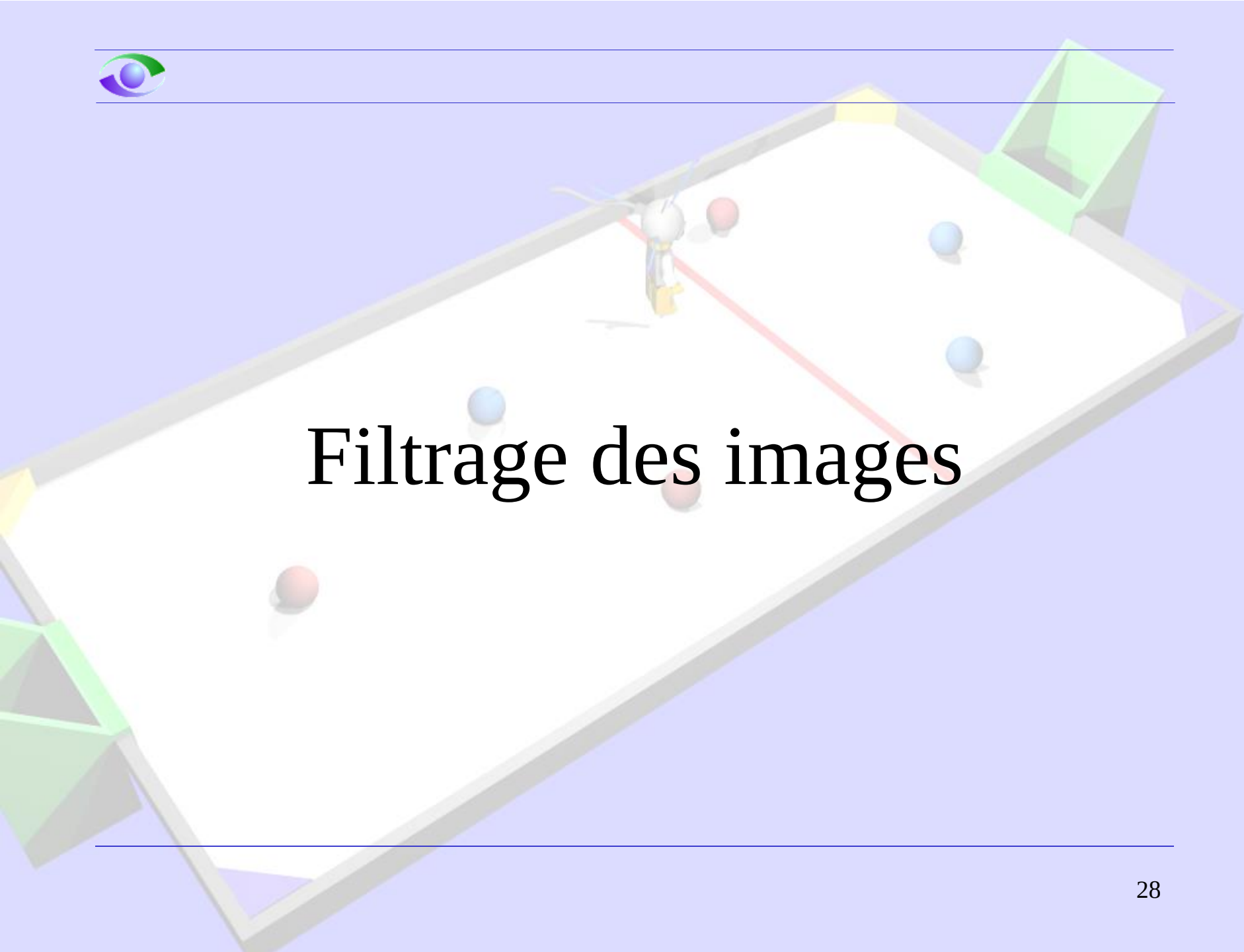
    (ex. Seuillage adaptatif)

"et/ou"

  - Robustesse au niveau de l'interprétation

    (ex. Confirmation de la reconnaissance en utilisant un 2e traitement (différent)).



Filtrage des images

Plan de la présentation

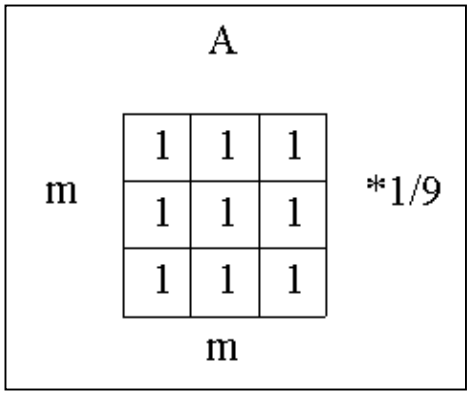
- ✓ Espaces des couleurs
- ✓ Traitement des images : Introduction
- ➢ Filtrage
- Segmentation
  - Extraction de régions
  - Extraction de contours
  - Détection de coins
- Conclusion

- Convolution de l'image avec un masque (noyau) fixe
  - La convolution est une opération linéaire:
    - commutative **A** * E = E * **A**
    - associative **A** * (B * E) = (**A** * B) * E

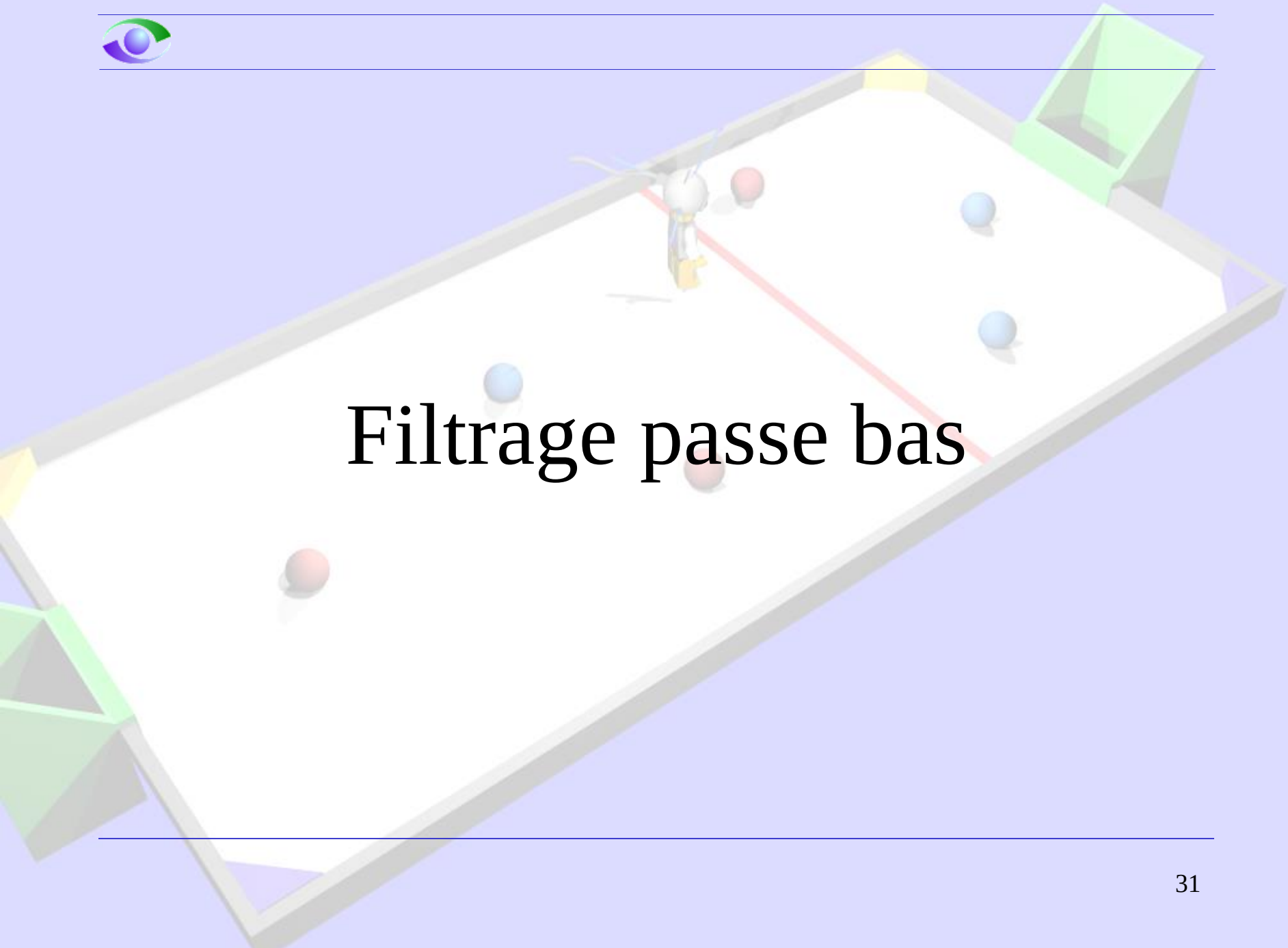
Équation de convolution

$$E_f(i,j) = A * E = \sum_{h=-\frac{m-1}{2}}^{\frac{m-1}{2}} \sum_{k=-\frac{m-1}{2}}^{\frac{m-1}{2}} A(h,k)\, E(i-h,\ j-k)$$

Exemple de masque



(voir cvSmooth dans OpenCV)

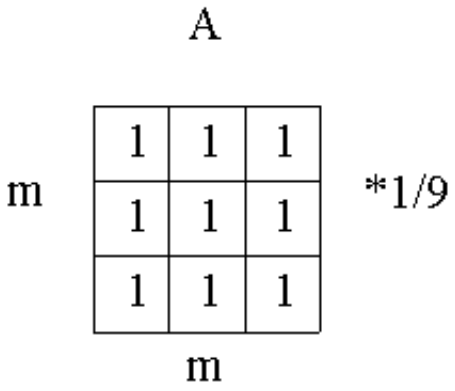


Filtrage passe bas

- Exemple de filtre : Le filtre de moyennage
  - Chaque pixel de l'image résultat prend comme valeur la somme des pixels voisins dans le masque.

90	100	120	125	110	90	100	
120	110	95	130	100	110	110	
115	110	100	120	90	105	110	
90	110	110	95	130	110	110	
120	125	100	110	105	110	125	
105	125	110	90	100	90	125	
110	130	100	110	90	100	130	

Résultat = 107



- Filtrage linéaire : Attention aux effets de bord !
  - les ignorer → image résultat plus petite
  - on peut ajouter des valeurs constantes sur les bords mais …
  - on peut considérer l'image comme étant périodique mais …
  - on peut considérer le bord de l'image comme un miroir mais …

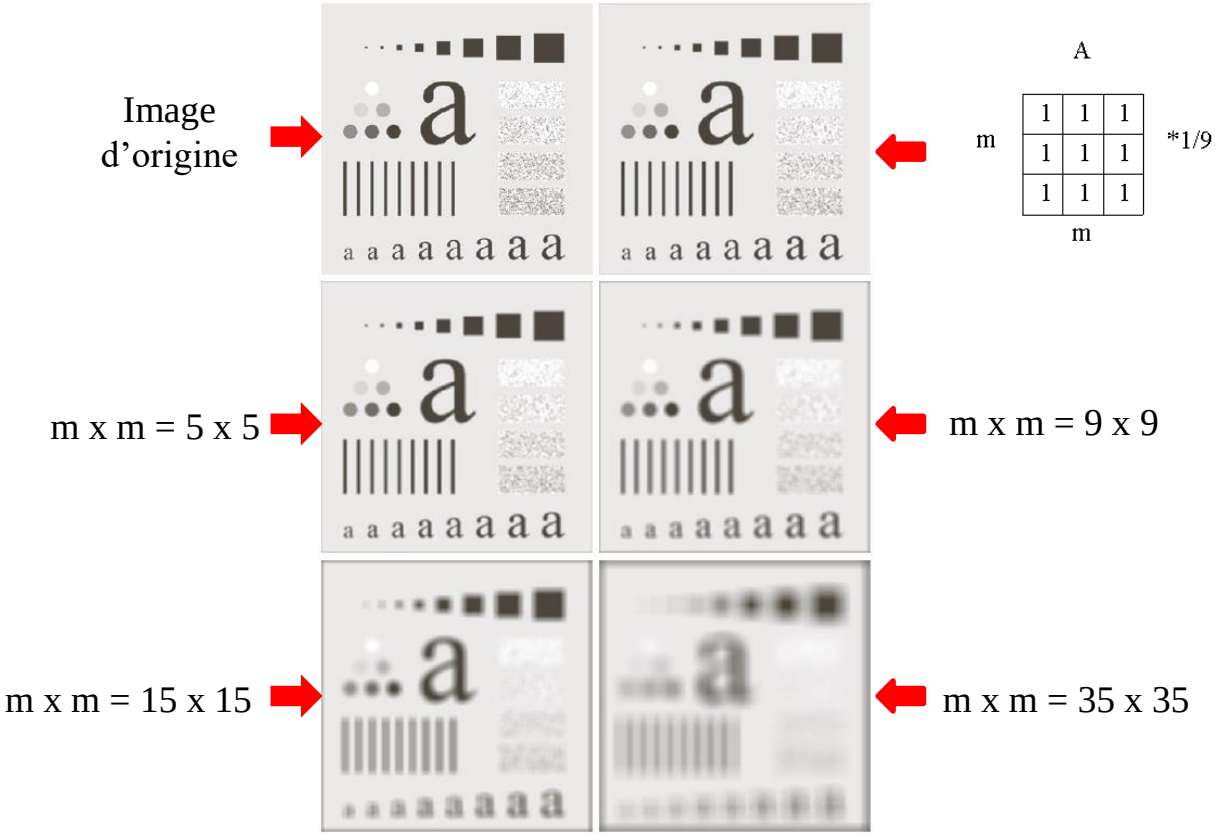
?	?	?						
?	95	130	120	125	110	90	100	
?	110	105	95	130	100	110	110	
	115	110	100	110	125	105	110	
	90	110	120	110	90	110	110	
	120	125	90	100	110	110	125	
	105	125	110	90	100	90	125	
	110	130	100	110	90	100	130	

Filtrage linéaire passe bas

1. Filtre moyenne

$$\frac{1}{5}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

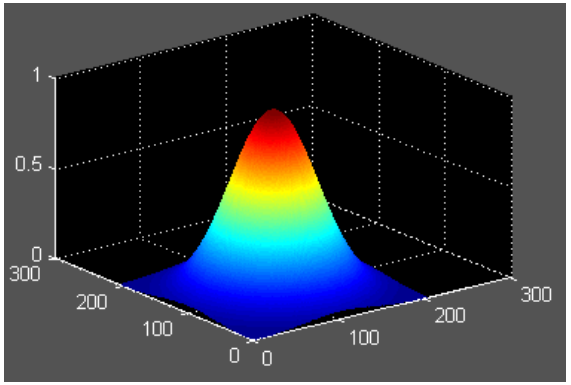
Carré *Octogonal*



Filtrage linéaire passe bas

2. Filtre Gaussien…

* Meilleur comportement fréquentiel.
  Préserve mieux les arêtes.



$$A(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-\left(x^2+y^2\right)}{2\sigma^2}}$$

– On choisit la taille du filtre w (en pixel) tel que $w \geq 5\,\sigma$

Exemple

$w = 3\ pixels \rightarrow \sigma = 3/5$

$$A = \begin{bmatrix} 0.0277 & 0.1110 & 0.0277 \\ 0.1110 & 0.4452 & 0.1110 \\ 0.0277 & 0.1110 & 0.0277 \end{bmatrix}$$

2. Filtre Gaussien…

Attention: Plus la gaussienne est large, plus le flou est marqué.
En général, un sigma $\sigma \leq 1$ est utilisé pour réduire le bruit.

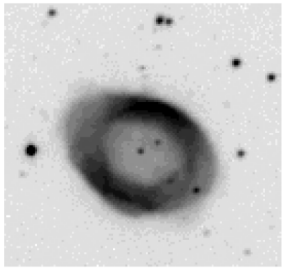
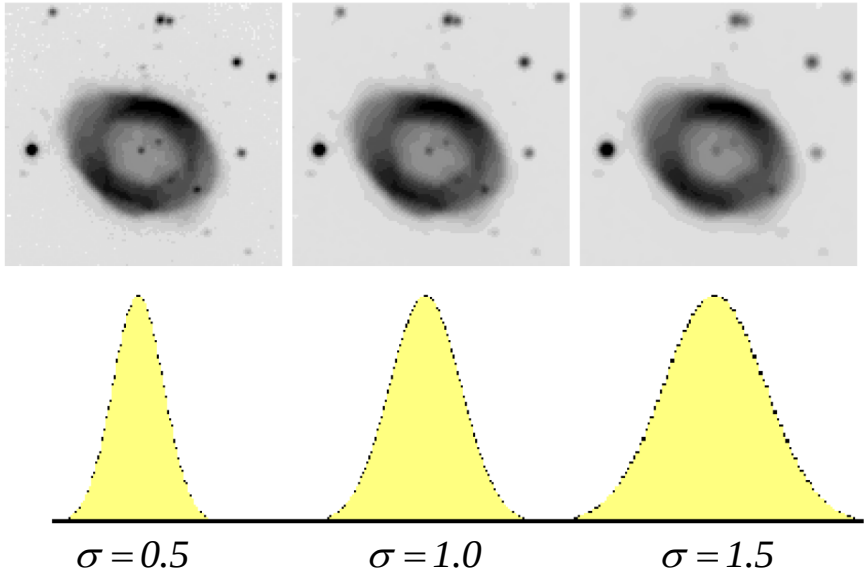


Image non filtrée



2. Filtre Gaussien

- Propriété du filtre gaussien ➔ Filtre séparable

Exemple

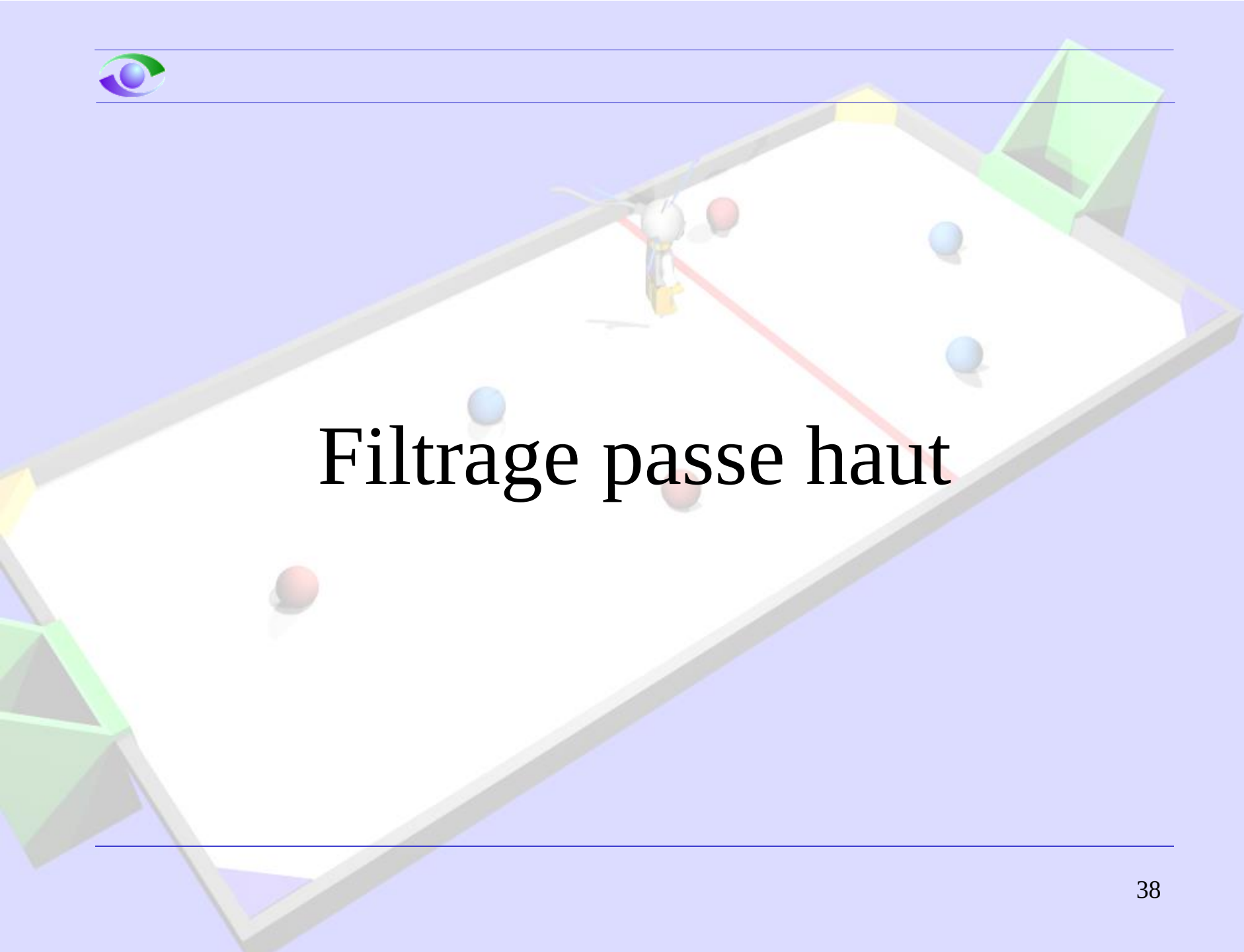
$w = 3\ pixels \rightarrow \sigma = 3/5$

1. Filtrer l'image avec le filtre gaussien 1D vertical

$$\text{Filtre 1D vertical : } \begin{bmatrix} 0.1664 \\ 0.6672 \\ 0.1664 \end{bmatrix}$$

2. Filtrer le résultat avec le filtre gaussien 1D horizontal

$$\text{Filtre 1D horizontal : } \begin{bmatrix} 0.1664 & 0.6672 & 0.1664 \end{bmatrix}$$



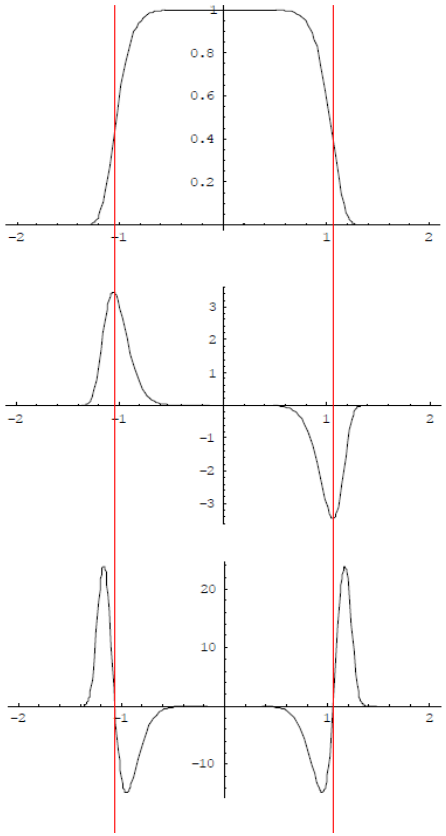
Filtrage passe haut

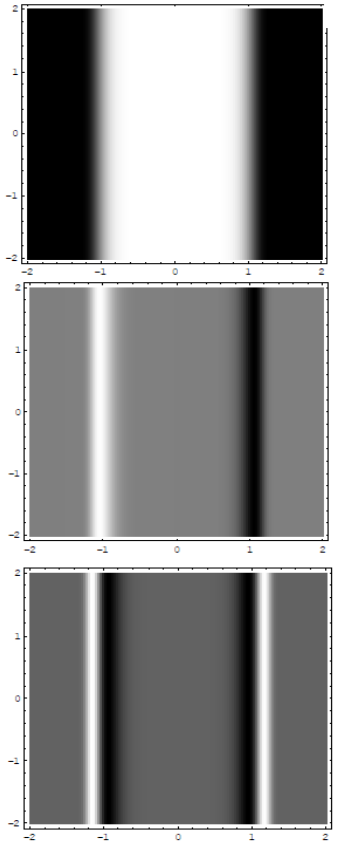
Principe des filtres de rehaussement d'arêtes

$f(x)$

$\frac{\partial f}{\partial x}$

$\frac{\partial^2 f}{\partial x^2}$





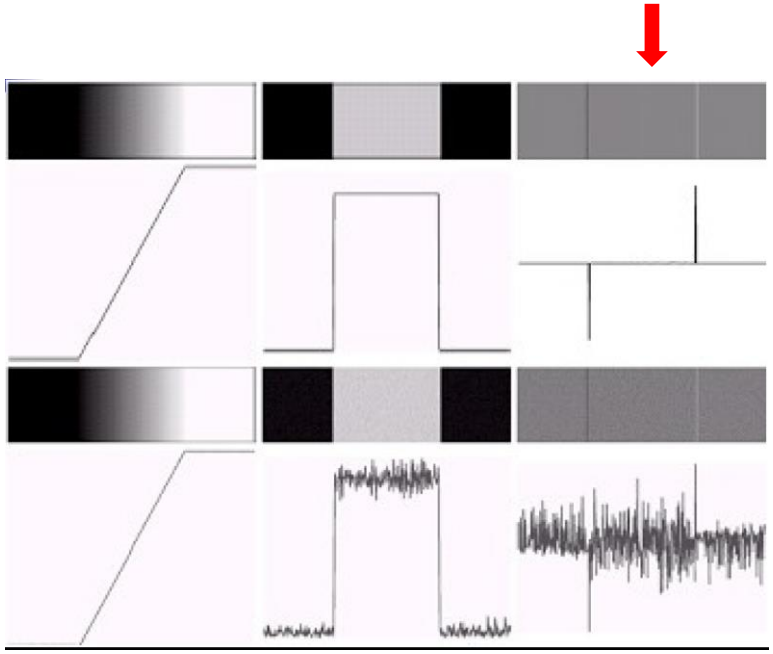
Image

Première dérivée

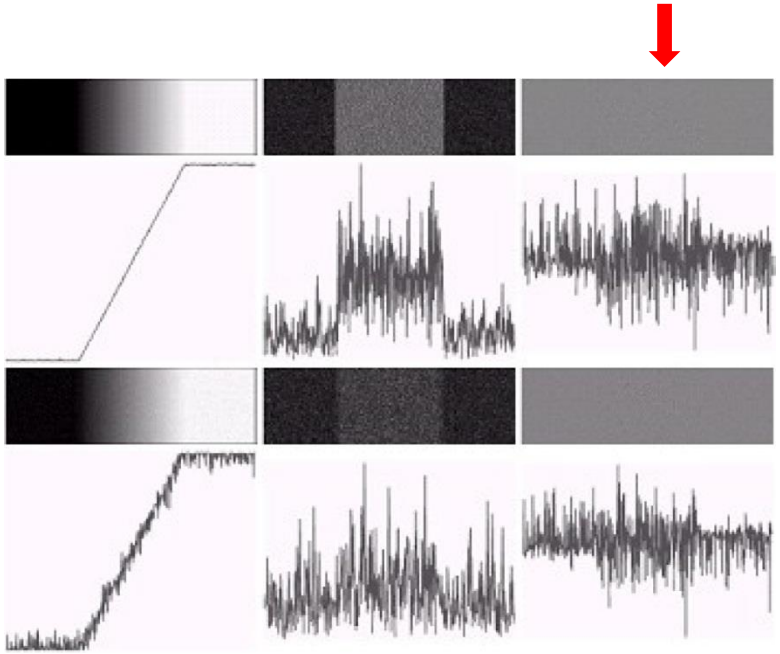
Deuxième dérivée

Effet du bruit sur les filtres de rehaussement d'arêtes

Filtre Gradient



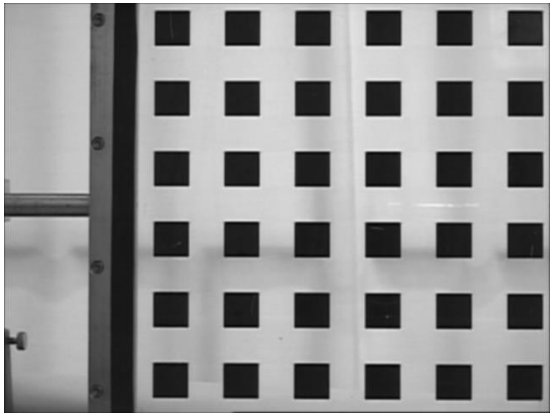
Filtre Gradient



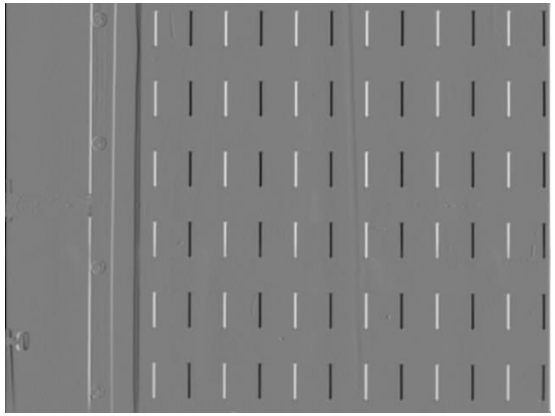
1. Filtre Gradient (Masque de Prewitt)

Gradient horizontal : $D_X = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ Gradient vertical : $D_Y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

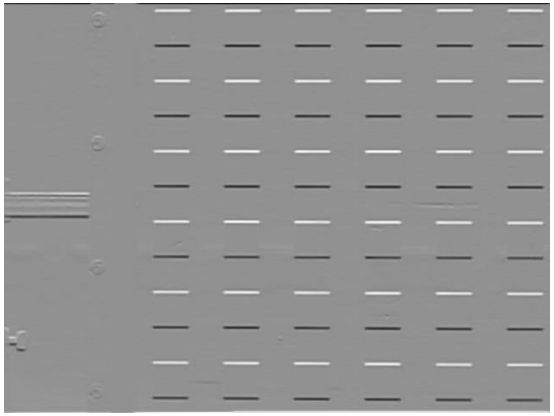
- Attention : la dérivée est sensible au bruit.
  Faire un filtrage gaussien avant de l'utiliser ou bien appliquer un masque de Sobel.
- À noter : $D \otimes (G \otimes I) = (D \otimes G) \otimes I$



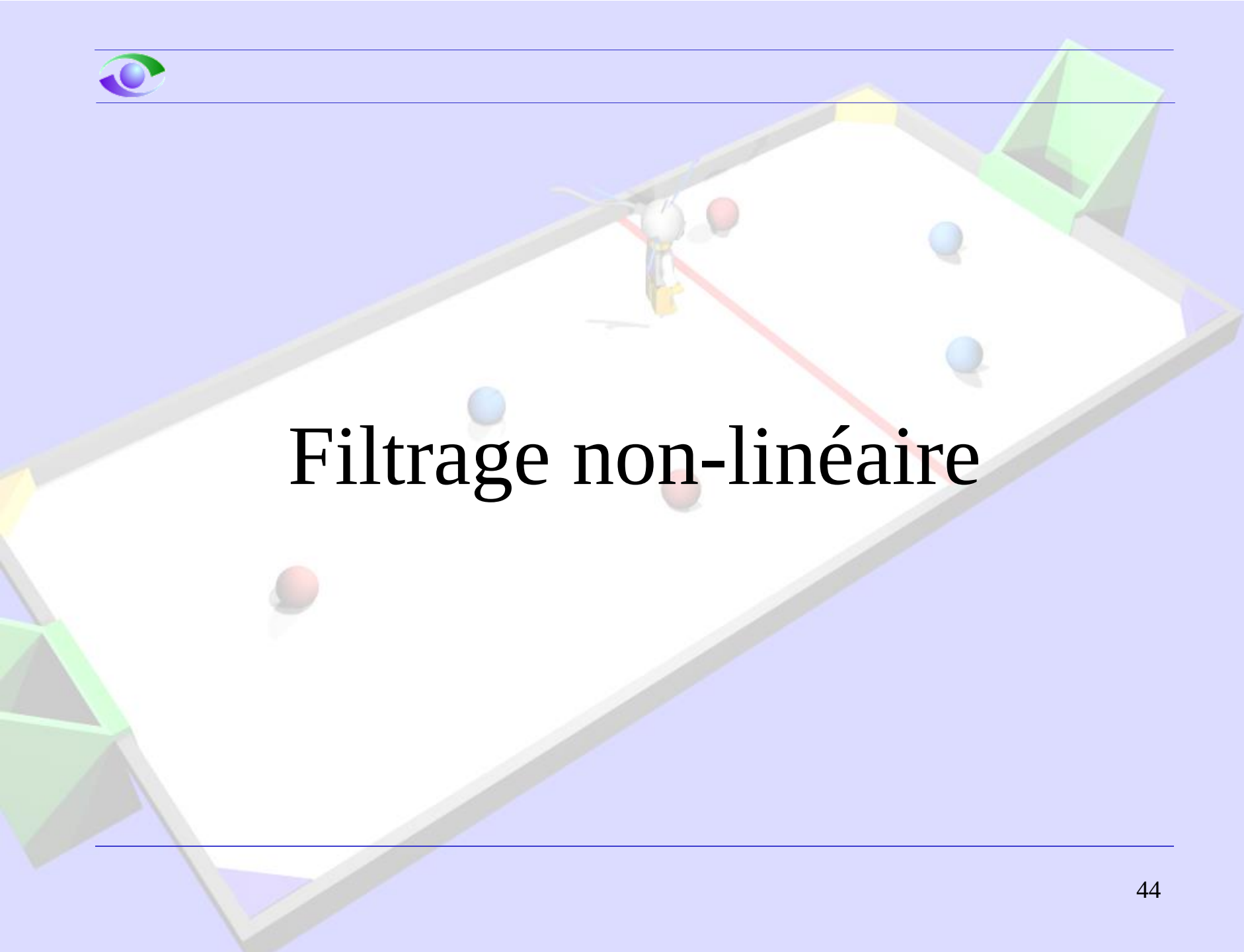
$G \otimes I$



$E_X = D_X \otimes (G \otimes I)$



$E_Y = D_Y \otimes (G \otimes I)$



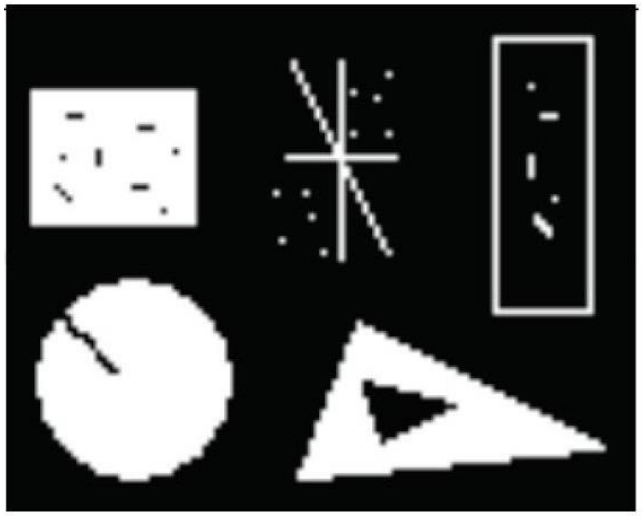
Filtrage non-linéaire

Note sur le filtrage non-linéaire

- Le résultat n'est pas une combinaison linéaire des pixels de l'image à traiter mais plutôt une fonction non-linéaire telle qu'un **min**, un **max** ou la **médiane**.

Opérateurs morphologiques

- Très utilisés sur les images binaires (images de masques)
  - *mais aussi sur les images en niveaux de gris*
- Permettent de modifier la morphologie des objets
  - Pour nettoyer le résultat de la segmentation
    - Remplir les trous, éliminer le bruit
  - Pour lisser le résultat de la segmentation



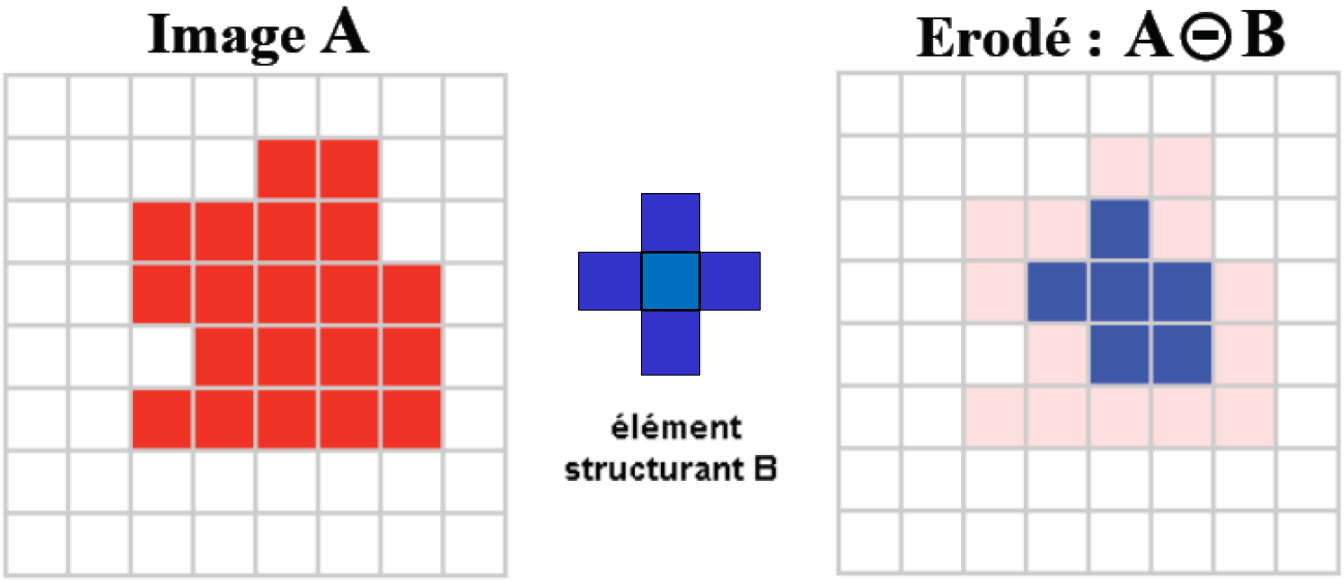
Utilisé en post-segmentation

- Caractérisés par
  - un élément structurant
  - des transformations
    - erosion, dilatation,
    - ouverture (érosion & dilatation), fermeture (dilatation & érosion)

Tiré de Alain Boucher - IFI

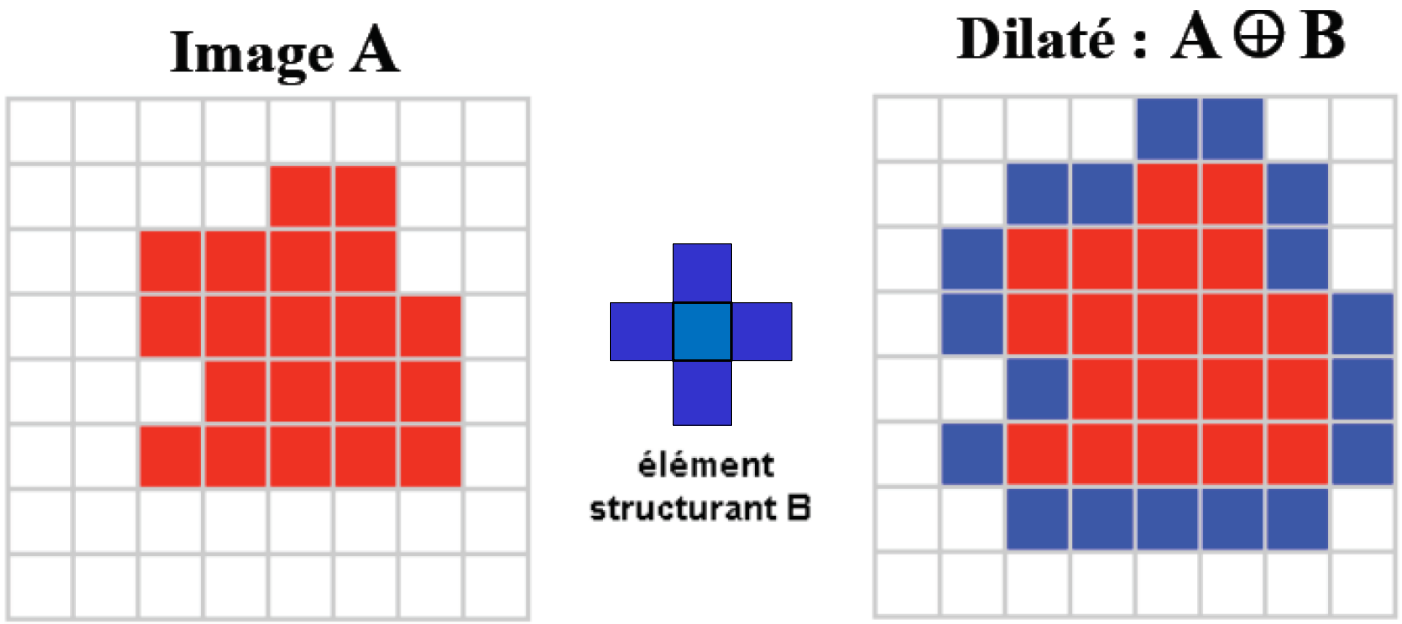
Érosion

- **Erosion**
  - Si un des pixels du masque est *fond* (valeur 0) alors le pixel central devient *fond*



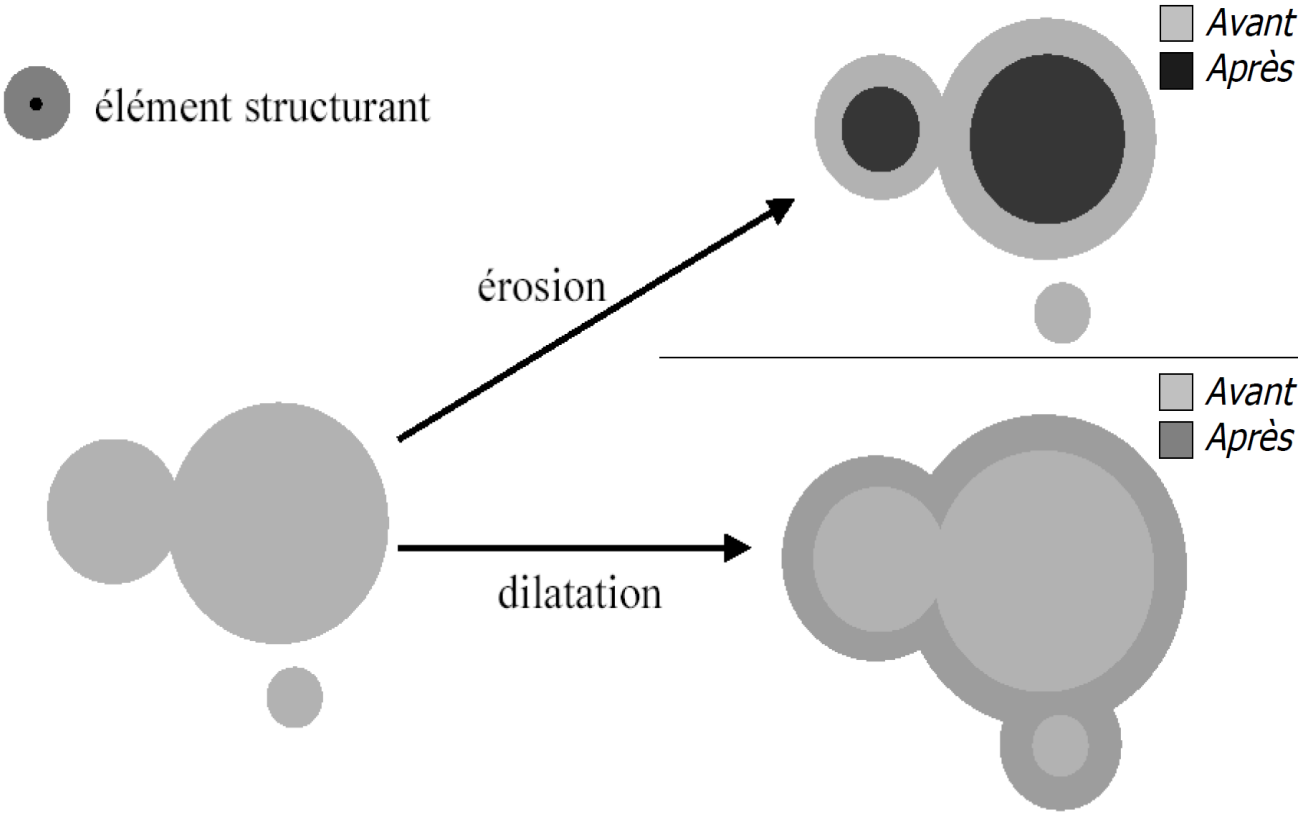
Fonction dans OpenCV : ***erode****.*

- **Dilatation**
  - Si un des pixels du masque fait partie de l'*objet*
    alors le pixel central devient *objet*



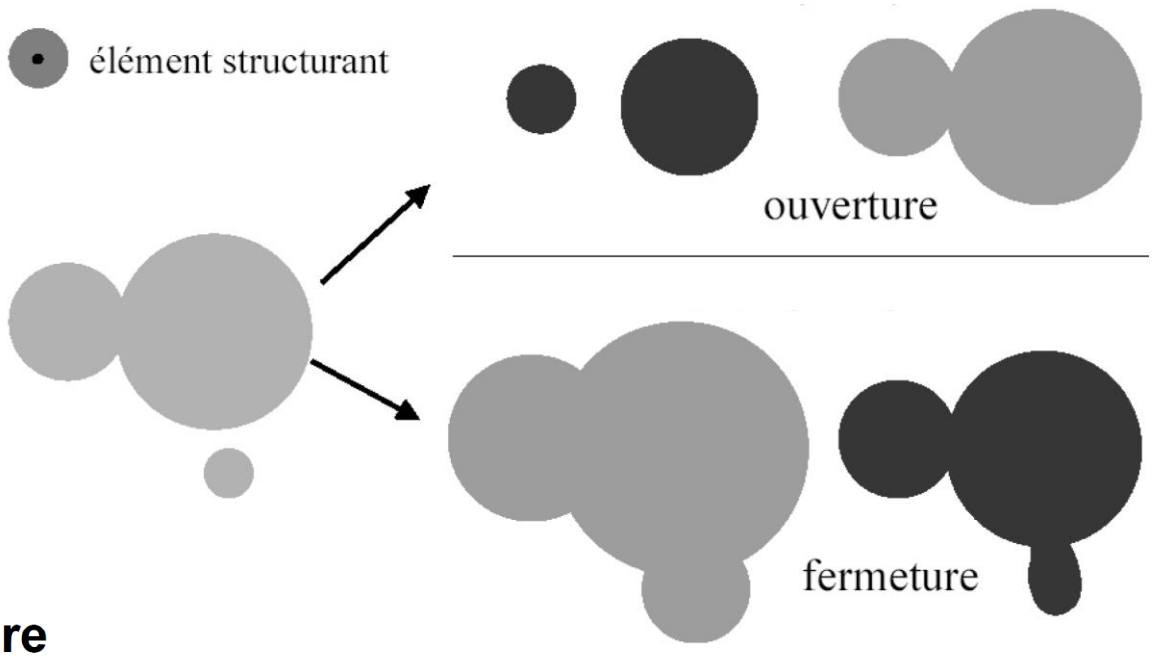
Fonction dans OpenCV : ***dilate****.*

Exemples : Érosion, Dilatation



Ouverture – Fermeture...

- **Ouverture**
  - Erosion puis dilatation



- **Fermeture**
  - Dilatation puis érosion

*(Fonctions dans OpenCV : **Open, Close**)*

Ouverture – Fermeture

Ouverture: - Les petits détails sont supprimés.
- Les objets « massifs » retrouvent leurs formes.

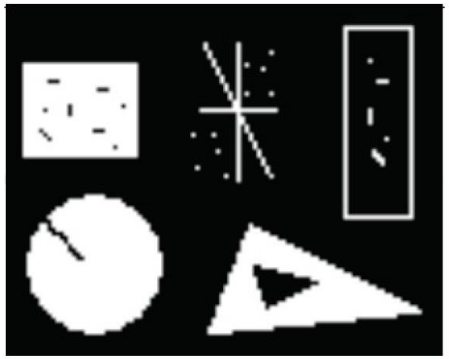
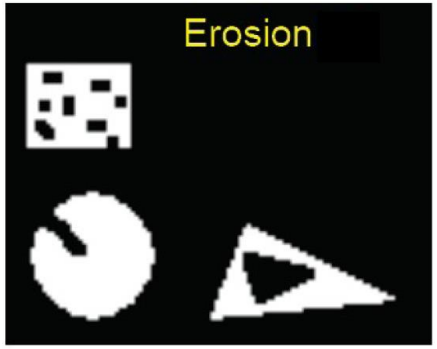
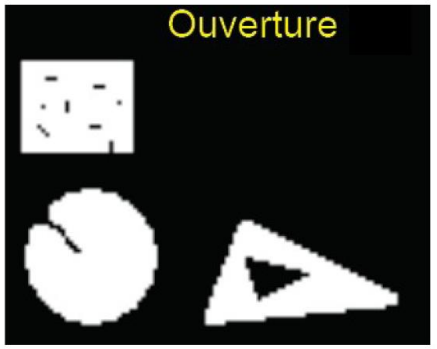
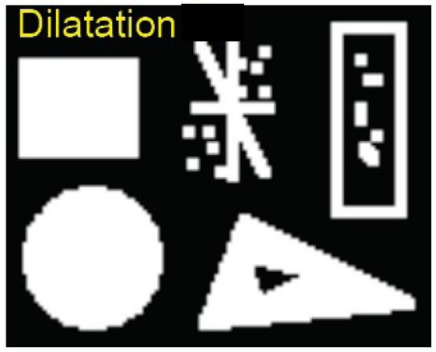
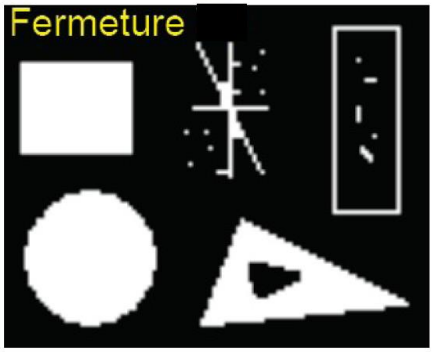


Image originale









Fermeture: - Les petits trous ou les fentes fines sont bouchés.
- Les objets « massifs » retrouvent leurs formes.

Filtre médian

- élimine le bruit **impulsionnel**
- préserve mieux les discontinuités
- plus coûteux en calculs**

1- trier les valeurs

64 64 64 64 64 64 255 255 255

2- extraire la médiane

64	64	64	64	64	64	64	64
64	64	64	64	64	64	64	64
64	64	64	64	64	255	255	64
64	64	64	255	255	64	64	64
64	64	64	64	255	64	64	255
64	64	64	64	64	255	64	128
64	64	64	64	64	64	128	128
64	64	64	64	255	128	128	128

*tirée de Efford

**S. Perreault et P. Hébert, *IEEE Trans. On Image Proc., Vol. 16, No. 9, 2007* proposent une implantation en temps constant. (utilisée dans OpenCV)

Filtre médian

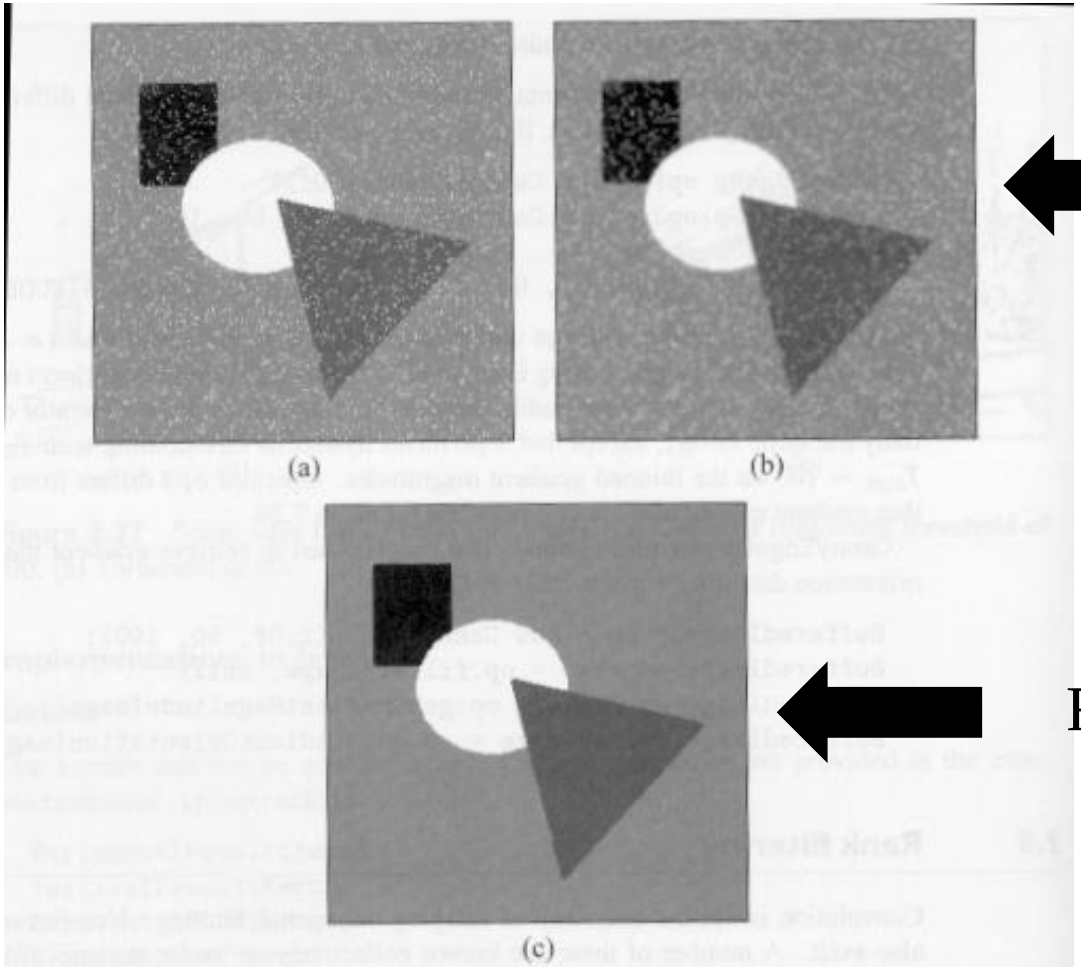


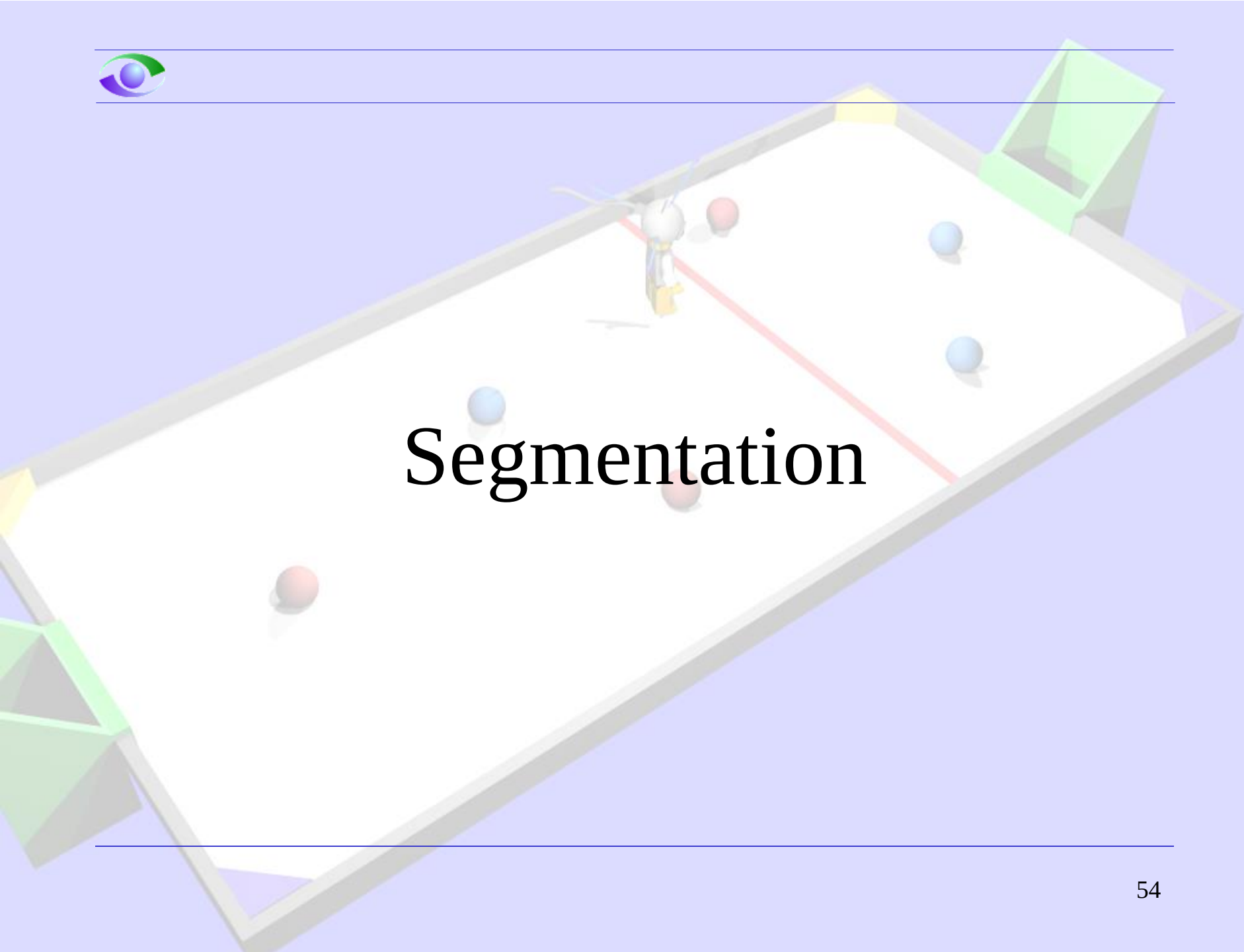
Figure 7.28 Noise suppression by mean and median filtering. (a) Image corrupted by five per cent impulse noise. (b) Result of 3 × 3 mean filtering. (c) Result of 3 × 3 median filtering.

Filtre passe bas

Remarquez l'effet sur le fond

Filtre médian

*tirée de Efford



Segmentation

Plan de la présentation

- ✓ Traitement des images : Introduction
- ✓ Espaces des couleurs
- ✓ Filtrage
- ➢ Segmentation
  - Contours
  - Points d'intérêt : coins
  - Régions

Exploitation des contours

Retour au filtrage passe haut

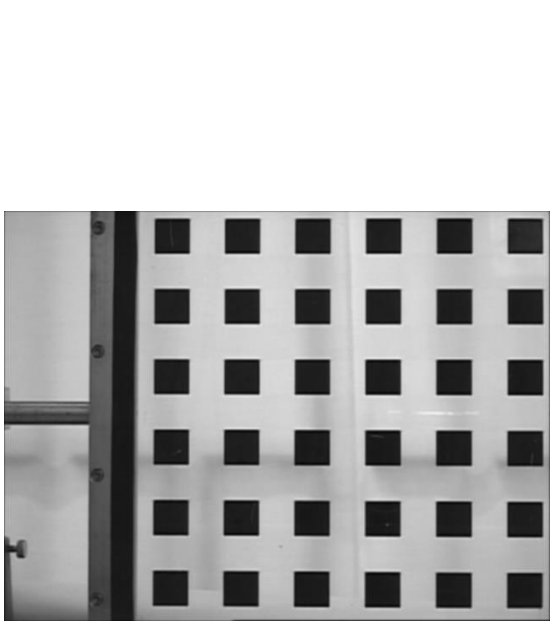
- Intérêt
  - Moins sensibles que les régions aux variations d'éclairage

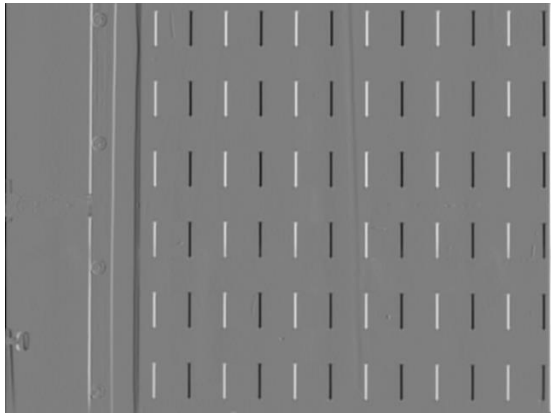
- Difficultés
  - Contours incomplets (par parties)
  - Il faut regrouper les contours d'un même objet

Filtre très répandu pour la segmentation des arêtes: filtre de Canny

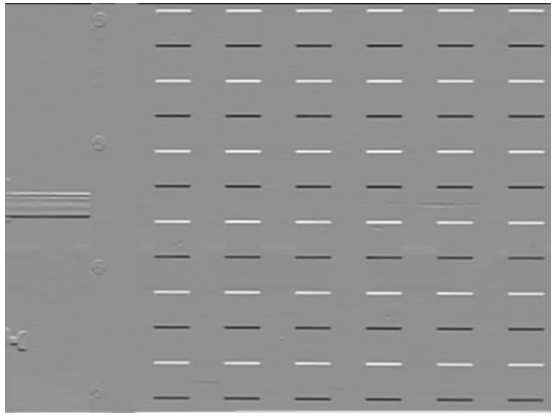
Extraction de contours (Filtre de Canny)

1. Calculer les gradients horizontal et vertical (E_X et E_Y)





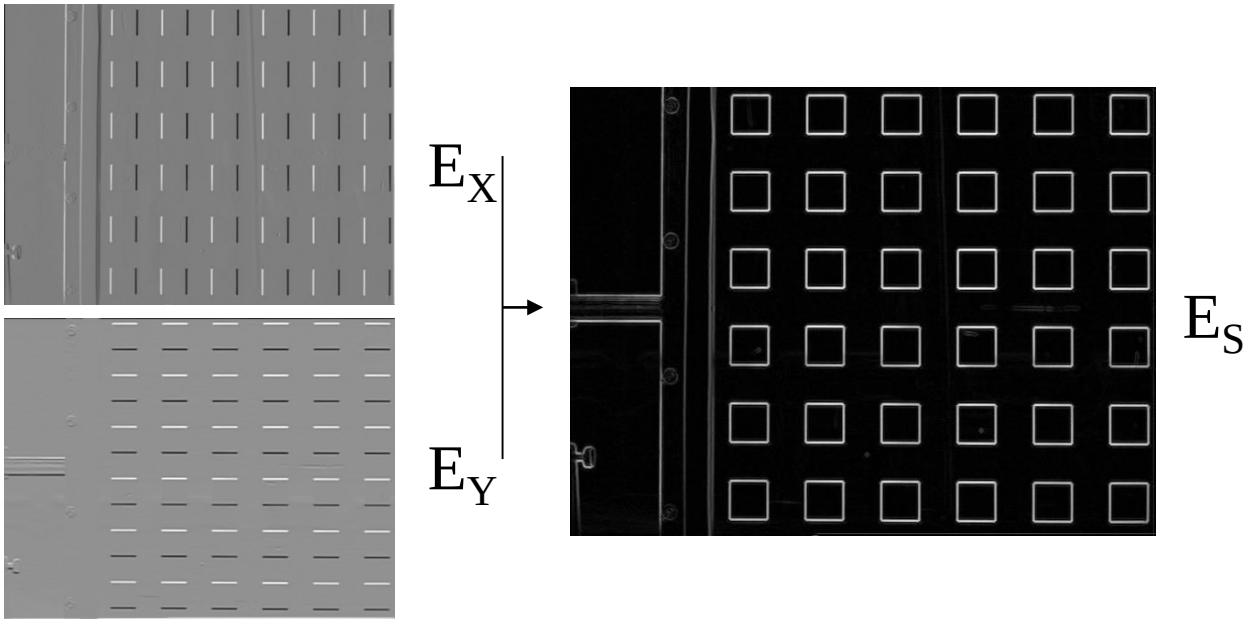
E_X



E_Y

Extraction de contours (Canny)

1. Calculer les gradients horizontal et vertical (E_X et E_Y)
2. Calculer les images : $E_s = \sqrt{E_X^2 + E_Y^2}$ $E_\theta = \tan^{-1}\left(\frac{E_Y}{E_X}\right)$



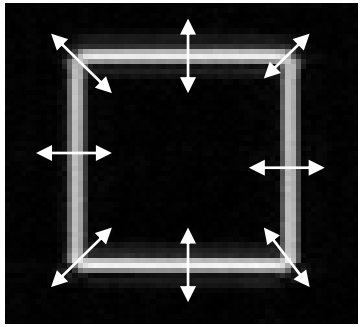
Extraction de contours (Canny)

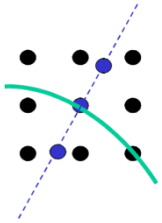
1. Calculer les gradients horizontal et vertical (E_X et E_Y)

2. Calculer les images : $E_s = \sqrt{E_X^2 + E_Y^2}$ $\quad E_\theta = \tan^{-1}\left(\frac{E_Y}{E_X}\right)$

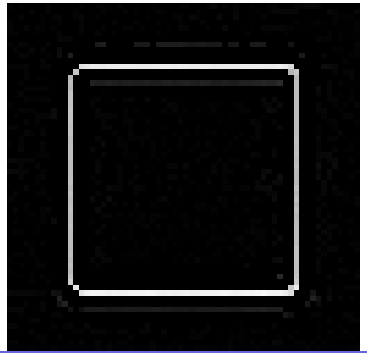
3. Éliminer les réponses non-maximales

   Un pixel a une réponse maximale si ses deux voisins, situés dans l'axe de sa normale, ont une réponse inférieure.





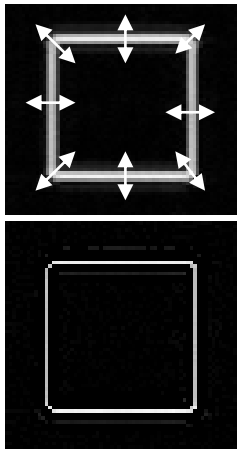
Un pixel non max est mis à zéro



Extraction de contours (Canny)

1. Calculer les gradients horizontal et vertical (E_X et E_Y)

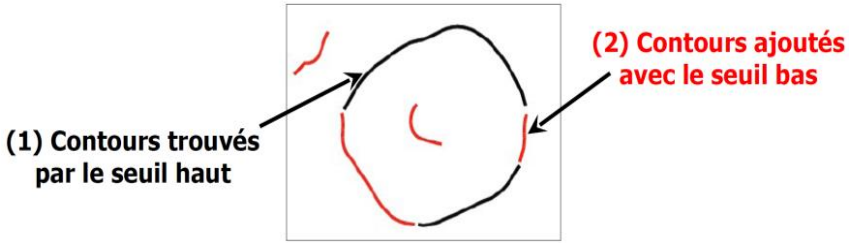
2. Calculer les images : $E_s = \sqrt{E_X^2 + E_Y^2}$ $\quad E_\theta = \tan^{-1}\left(\frac{E_Y}{E_X}\right)$



3. Éliminer les réponses non-maximales

   Un pixel a une réponse maximale si ses deux voisins, situés dans l'axe de sa normale, ont une réponse inférieure. ***Un pixel non-max est mis à zéro.***

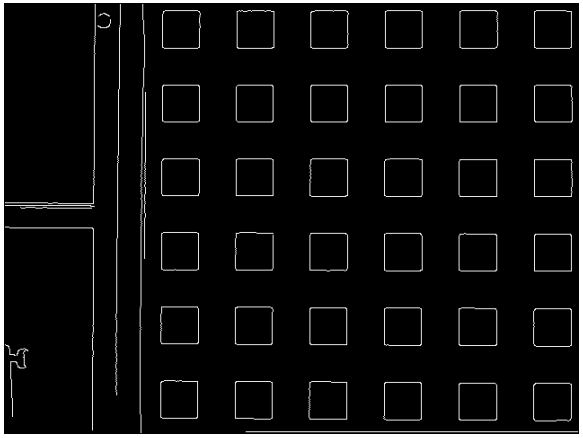
4. Seuillage par hystérésis (2 seuils)
   1. *Si réponse du pixel > Seuil 1* ➔ *Contour*
   2. *Suivre le contour tant que la réponse des pixels rencontrés soit > Seuil 2*



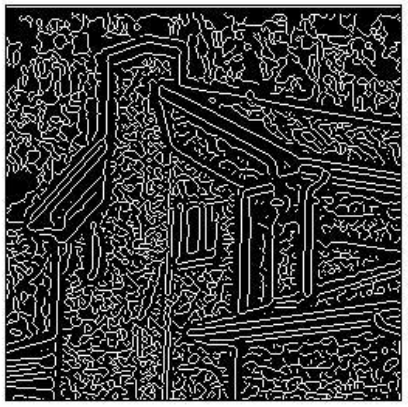
Extraction de contours (Canny)

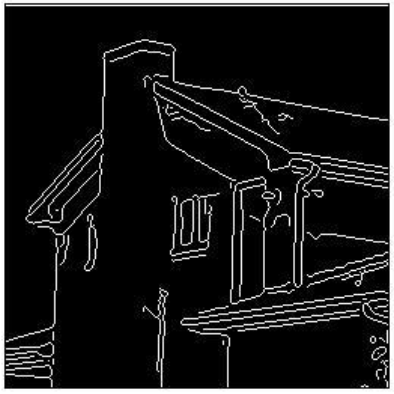
5. Sortie du détecteur de contours de Canny :

→Image de contours (binaire)

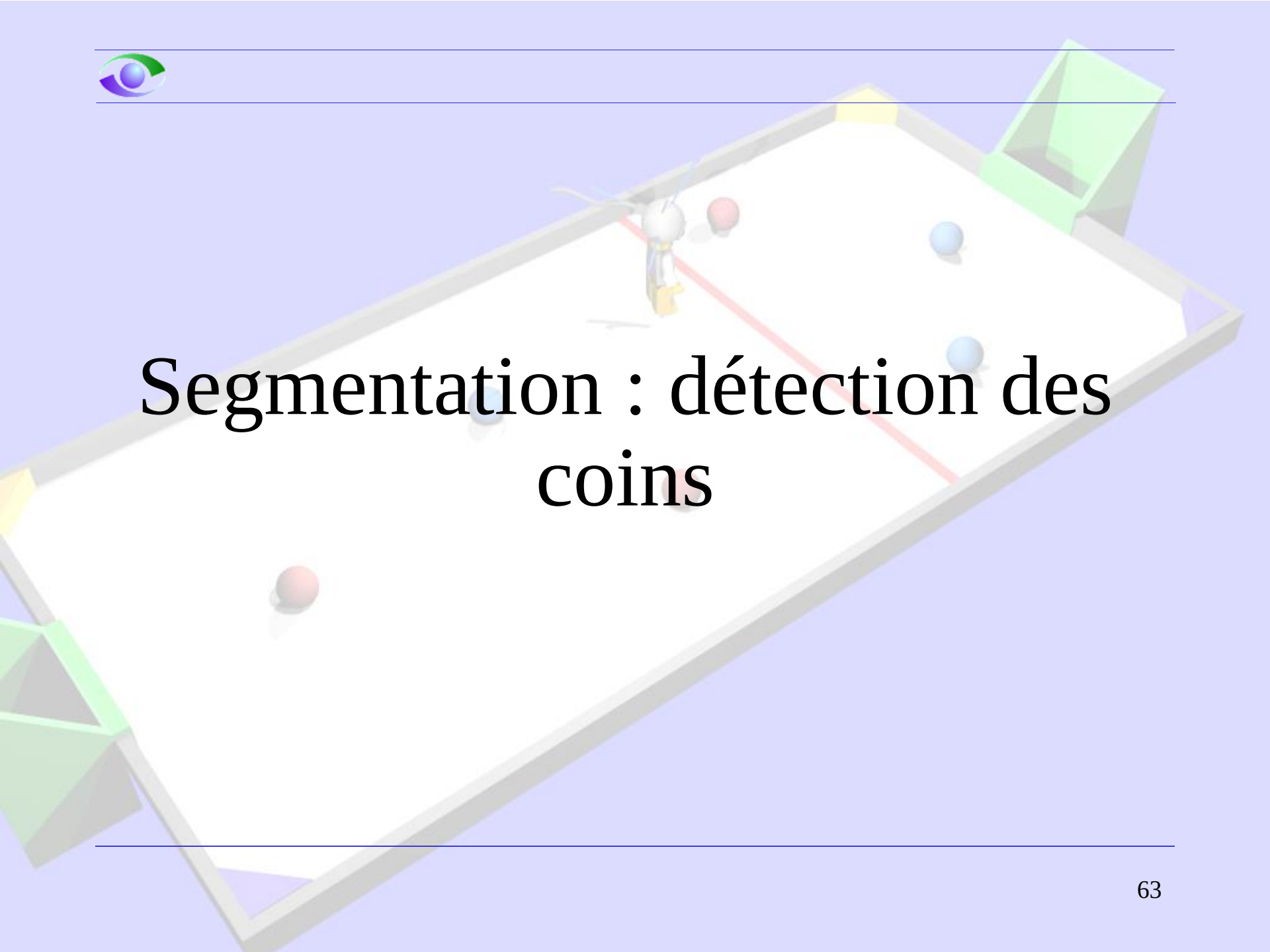


Exemple sur l'impact du seuillage et l'effet de l'ombrage





(voir aussi : OpenCV : les fonctions **canny** et **findContours**)



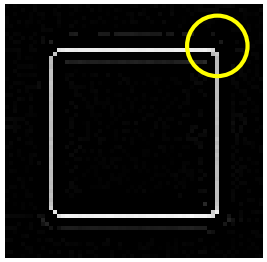
Segmentation : détection des coins

Plan de la présentation

- ✓ Traitement des images : Introduction
- ✓ Espaces des couleurs
- ✓ Filtrage linéaire
- ➢ Segmentation
  - ✓ Contours
  - ▪ Points d'intérêt : coins
  - ▪ Régions

❑ Détecteur de Harris

- Constat: les détecteurs d'arêtes se comportent moins bien près des coins



- Proposition: détecter les endroits où le gradient est fort dans plus d'une direction

- Réalisation: **détecteur de Harris**

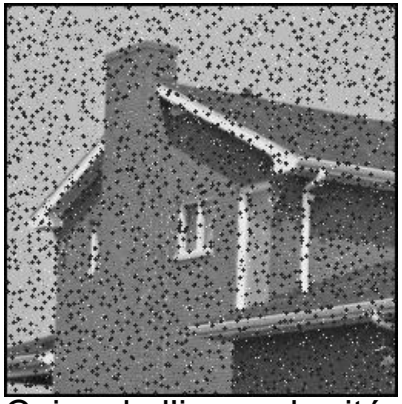
- Voir aussi dans Open CV la fonction : *Good features to track*

❑ Détecteur de Harris

Image originale

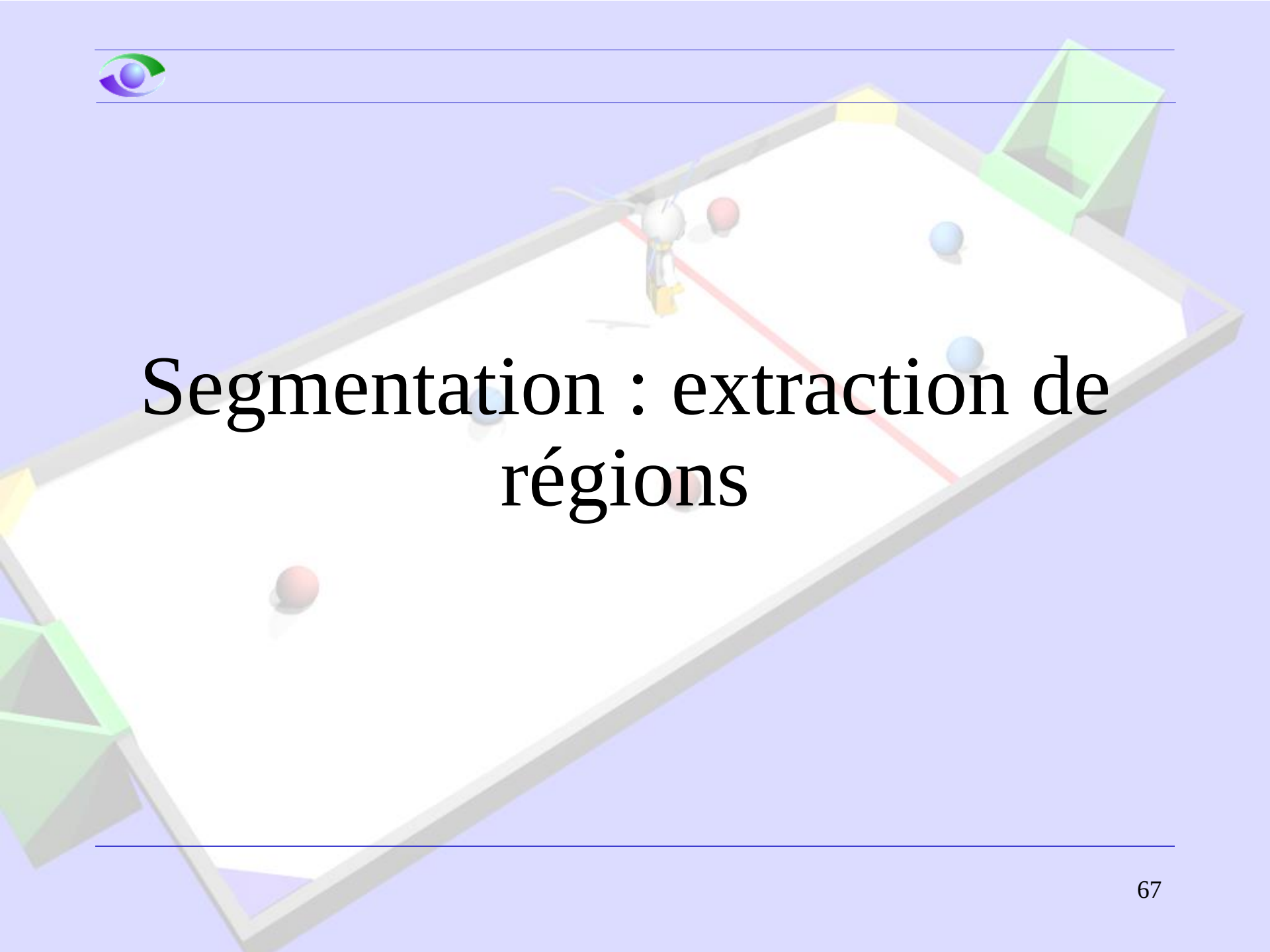
Coins de l'image originale

Image bruitée



Coins de l'image bruitée

Extrait de : Patrick Labatut (http://www.tsi.telecom-paristech.fr/)



Segmentation : extraction de régions

Plan de la présentation

- ✓ Traitement des images : Introduction
- ✓ Espaces des couleurs
- ✓ Filtrage
- ➢ Segmentation
  - ✓ Extraction de contours
  - ✓ Détection de coins
  - ▪ Extraction de régions

Segmentation en régions

- Découper l'image en régions de mêmes caractéristiques
  - Exemples de caractéristiques sur lesquelles on peut segmenter
    - Couleur (caractéristique d'un pixel)
    - Texture (caractéristique de voisinage)
    - Mouvement inter-images (caractéristique de voisinage)

- Segmentation simple (basée sur les caractéristiques des pixels) ➔ 3 étapes
  1. Classifier chaque pixel

     Ex. : 0 → Noir
     1 → Blanc
     2 → Jaune
     3 → Rouge
     4 → Autre
  2. Regrouper par régions

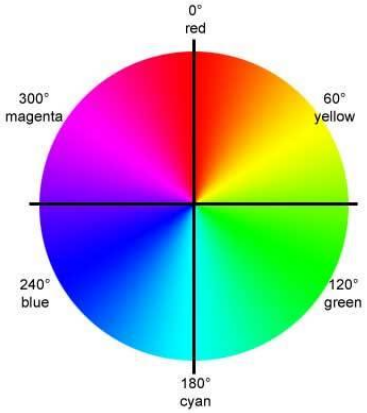
     Regrouper les pixels connexes de même classe.
  3. Étiqueter les régions

Segmentation en régions

- Étape 1 – Classifier chaque pixel…
  - Chaque pixel est caractérisé par la valeur de ses paramètres

    Paramètres : H, S et V si on désire segmenter sur la couleur
  - Modèle : Intervalle sur les paramètres ⇔1 classe

    Ex. : Jaune ➔ $55^o \leq H \leq 65^o$



➢ Comment choisir les intervalles pour que les régions soient représentatives ?

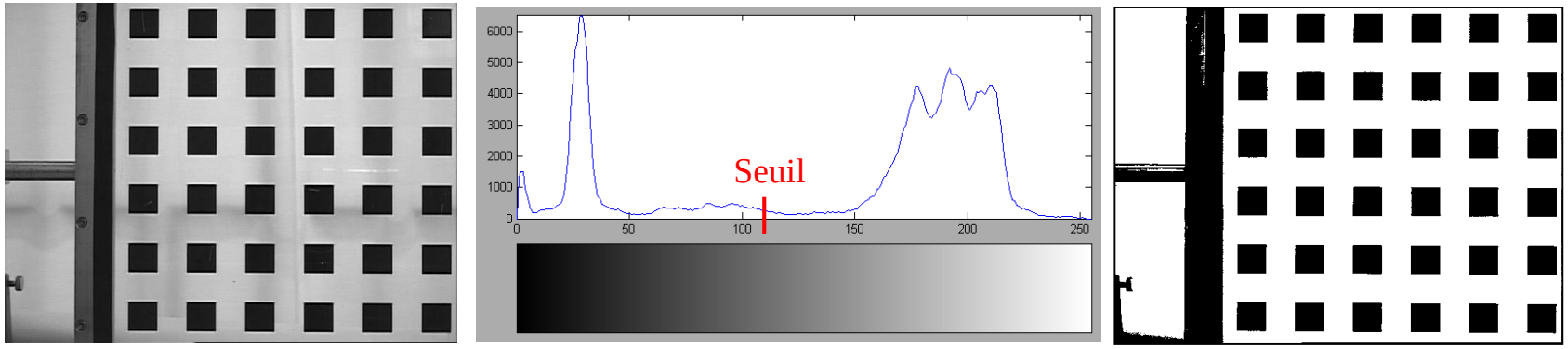
Ex. : On désire mettre en évidence des objets d'une couleur distincte

- Mesure directe (un modèle basé sur un ensemble d'images)
- Analyse d'histogramme (pour adaptation automatique d'intervalles)

Étape 1 – Classifier chaque pixel…

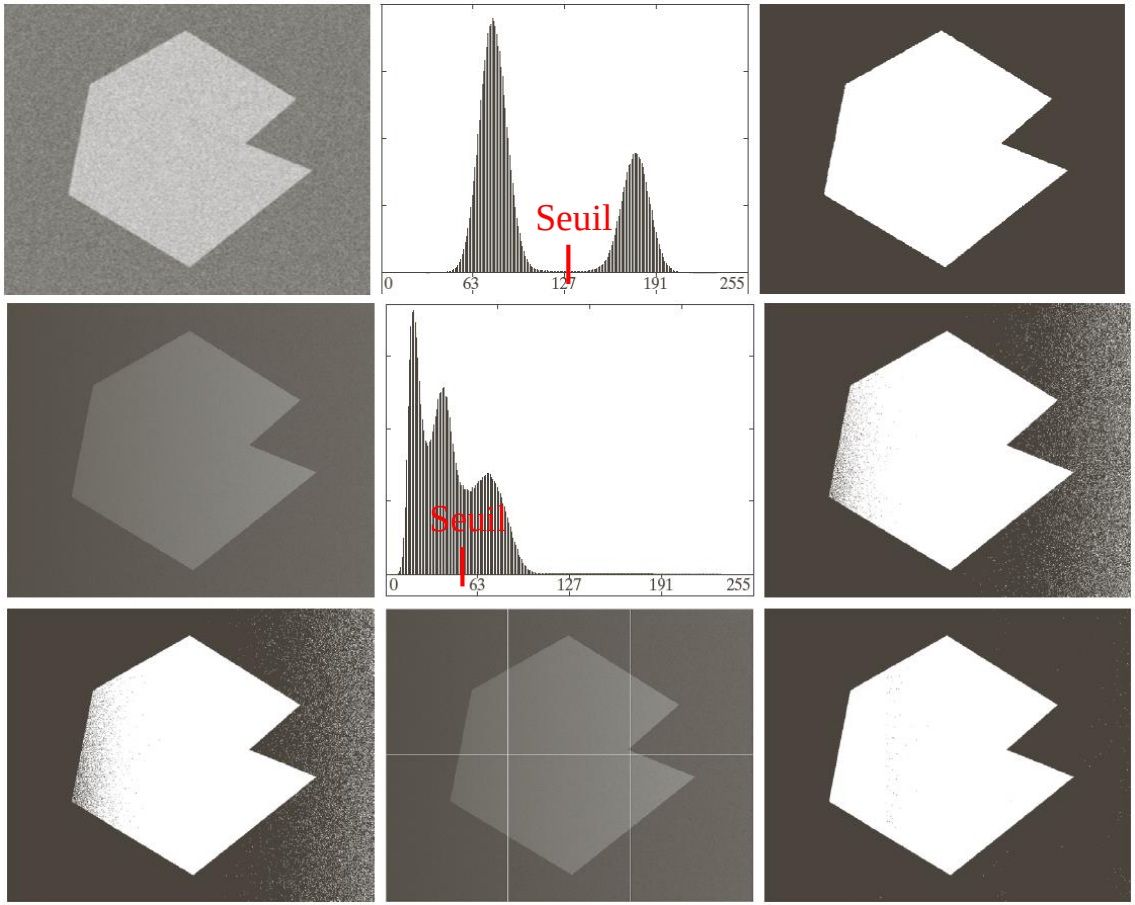
- Analyse d'histogramme
  - On tente de séparer les différents modes de l'histogramme
  - Cette procédure peut être automatisée: Des commandes existent pour calculer automatiquement le seuil (e.g. Matlab : Graythreshold)

Histogramme d'une image en niveaux de gris (Ex: canal V dans HSV)



Étape 1 – Classifier chaque pixel…

- Analyse d'histogramme
  - Seuillage adaptatif (Efficace pour ombrage ou éclairage non uniforme)



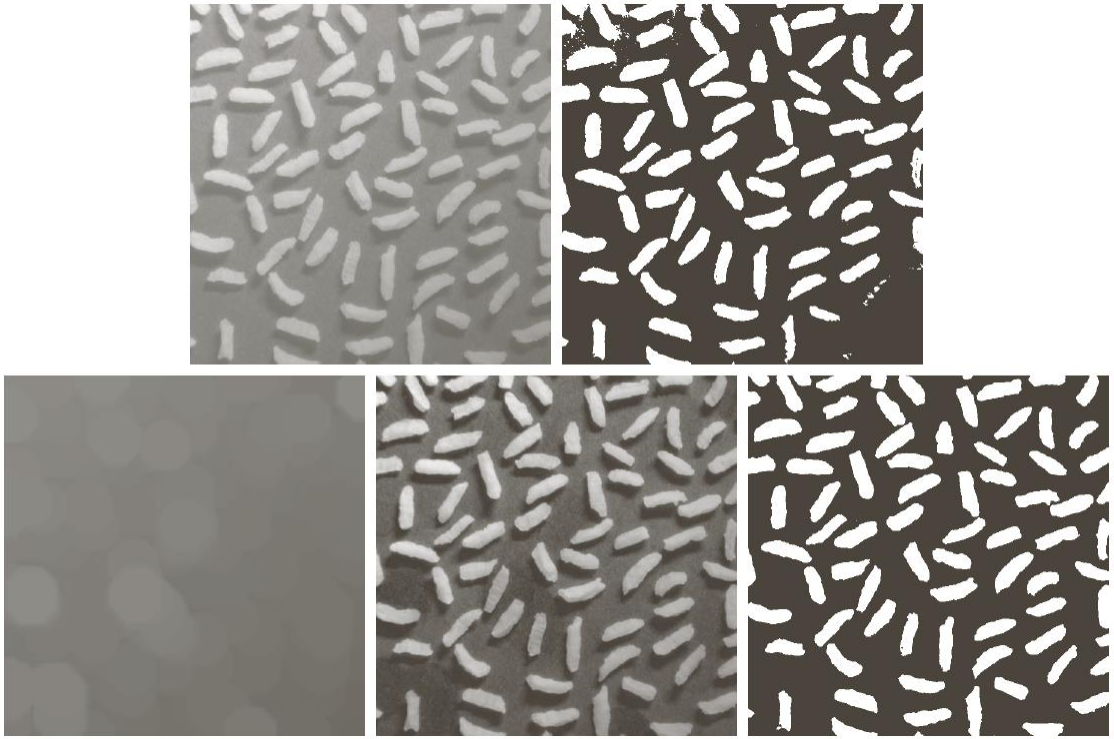
*Tirée de Pratt

Dans OpenCV, voir les fonctions ***threshold et adaptive threshold****.*

Étape 1 – Classifier chaque pixel…

Autre option efficace pour ombrage ou éclairage non uniforme:
Fonction morphologique TOP HAT (Image d'origine – Son ouverture)

Éclairage non uniforme



*Tirée de Gonzales and Woods

a b
c d e

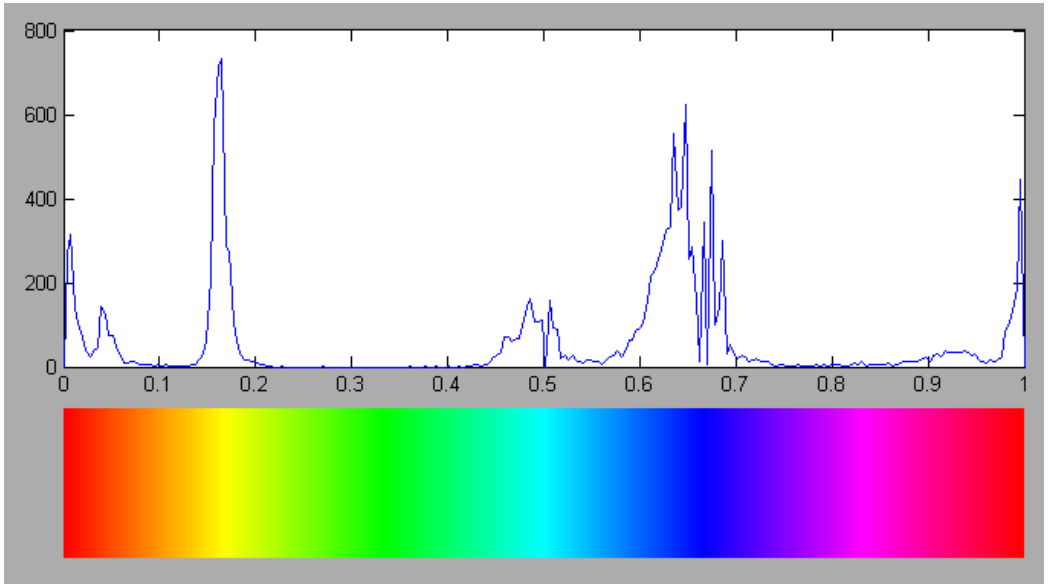
FIGURE 9.40 Using the top-hat transformation for *shading correction*. (a) Original image of size 600 × 600 pixels. (b) Thresholded image. (c) Image opened using a disk SE of radius 40. (d) Top-hat transformation (the image minus its opening). (e) Thresholded top-hat image.

*Dans OpenCV, voir la fonction **tophat**.*

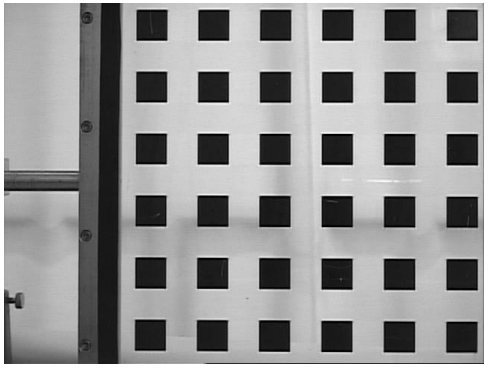
Étape 1 – Classifier chaque pixel…

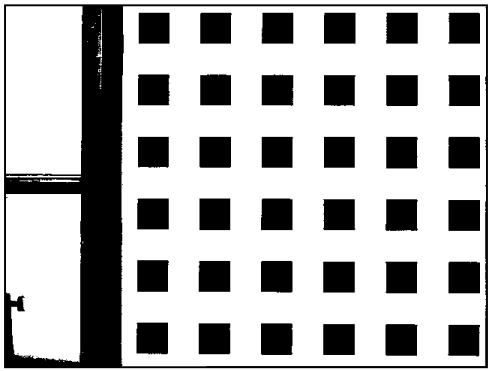
- Analyse d'histogramme
  - On tente de séparer les différents modes de l'histogramme
  - Cette procédure peut être automatisée

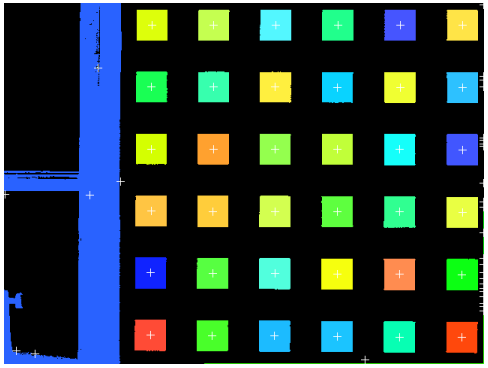
Histogramme des couleurs (Canal H dans HSV)



- Étape 2 – Regrouper par régions...
  - Regrouper les pixels connexes de la même classe pour former des régions.

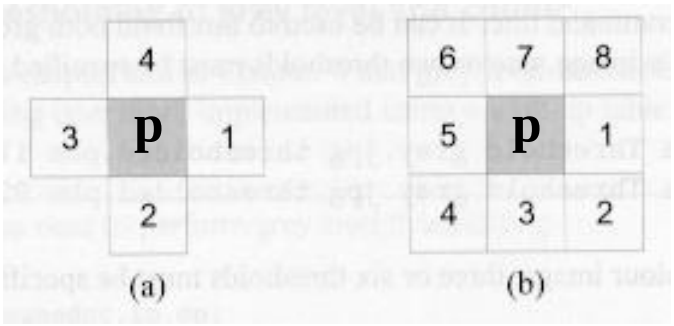






- Étape 2 – Regrouper par régions
  - Regrouper les pixels connexes de la même classe pour former des régions.

➢Définition de la connexité (voisinage)



Connexité à 4 voisins

Connexité à 8 voisins

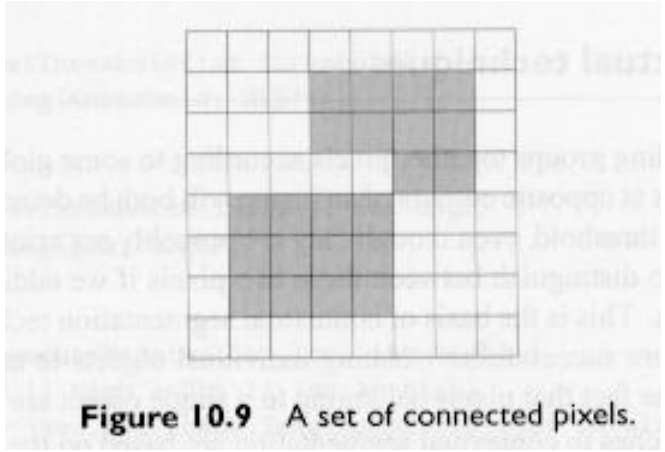
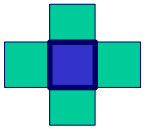
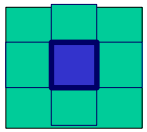


Figure 10.9 A set of connected pixels.





*Tirée de Efford

p et q sont connexes-4 si q est dans $N_4(p)$

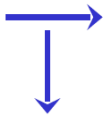
p et q sont connexes-8 si q est dans $N_8(p)$

- Étape 3 – Étiqueter les régions...
  - On désire donner une valeur commune (étiquette) pour les pixels d'une région.
  - On désire avoir une valeur différente pour chaque région.

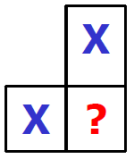
Premier parcours de l'image

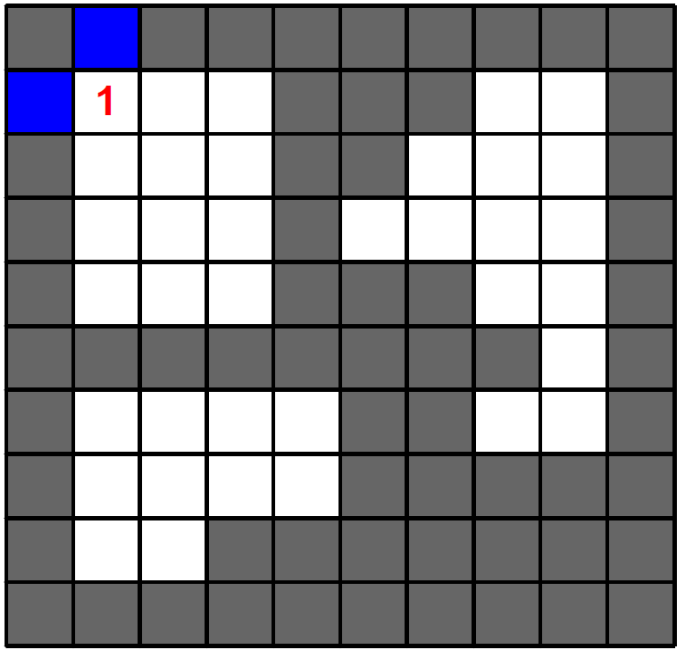
- Pour chaque pixel d'une région, on lui affecte
  - soit la plus petite étiquette parmi ses voisins haut et gauche
  - soit une nouvelle étiquette.

Parcours



Voisinage



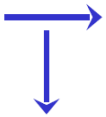


- Étape 3 – Étiqueter les régions...

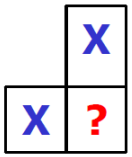
Premier parcours de l'image

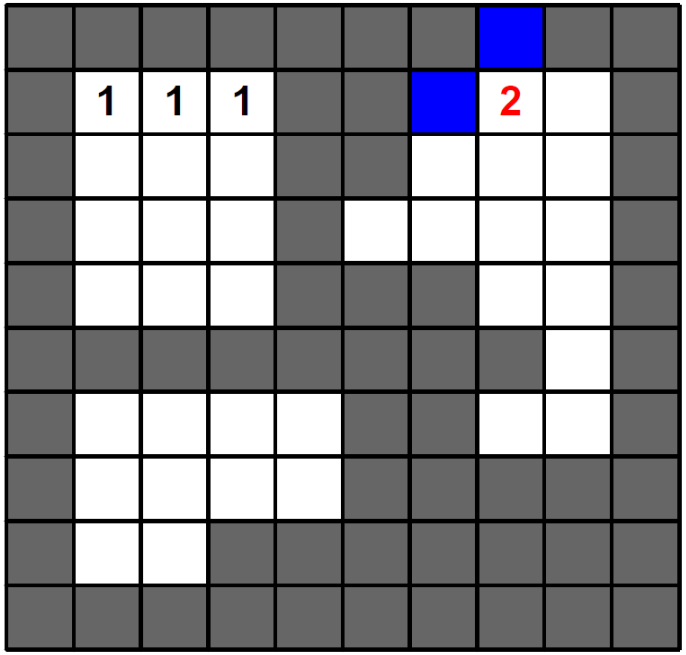
- Pour chaque pixel d'une région, on lui affecte
  - soit la plus petite étiquette parmi ses voisins haut et gauche
  - soit une nouvelle étiquette.

Parcours



Voisinage



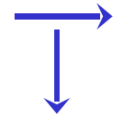


- Étape 3 – Étiqueter les régions...

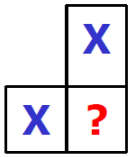
Premier parcours de l'image

- Pour chaque pixel d'une région, on lui affecte
  - soit la plus petite étiquette parmi ses voisins haut et gauche
  - soit une nouvelle étiquette.

Parcours



Voisinage



	1	1	1				2	2	
	1	1	1			3	2	2	
	1	1	1		4	3	2	2	
	1	1	1				2	2	
								2	
	5	5	5	5			6	2	
	5	5	5	5					
	5	5							

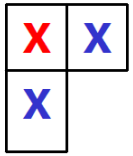
- Étape 3 – Étiqueter les régions...

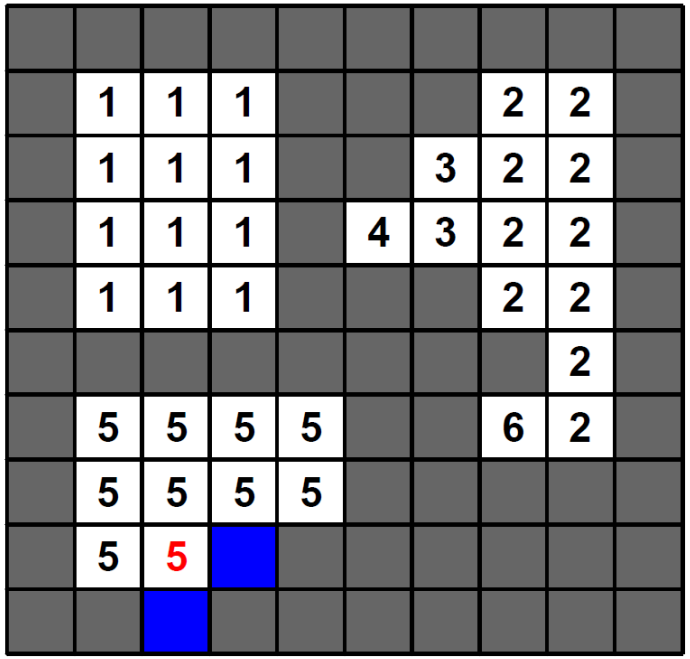
Deuxième parcours de l'image

- Pour chaque pixel d'une région, on lui affecte
  - la plus petite étiquette parmi la sienne et celles ses voisins bas et droite

Parcours

Voisinage





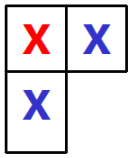
- Étape 3 – Étiqueter les régions

Deuxième parcours de l'image

- Pour chaque pixel d'une région, on lui affecte
  - la plus petite étiquette parmi la sienne et celles ses voisins bas et droite

Parcours

Voisinage



	1	1	1				2	2	
	1	1	1			2	2	2	
	1	1	1		2	2	2	2	
	1	1	1				2	2	
								2	
	5	5	5	5			2	2	
	5	5	5	5					
	5	5							

❑ Exploitation du résultat de la segmentation

- Pour une étiquette donnée, récolter diverses informations :
  Nombre de pixels, largeur, hauteur, centre, point de contact avec le sol, etc.
- Définir des critères pour éliminer les régions sans intérêt

<u>Par exemple</u> : On cherche la ligne rouge. On n'est pas intéressé par le rouge sur le chandail en arrière-plan.

➢Exemples de critères

- Rapport Hauteur/Largeur
- Un cercle couvre (π/4)*100% de la surface de son "bounding box"; Un rectangle, près de 100%
- Position du centre de gravité
- etc. (Il s'agit souvent des critères empiriques, soyez inventifs !)

Conclusion

- Considérations calculatoires: le traitement d'images sur les pixels est très coûteux.
- Quel est le temps raisonnable de calcul ?
- Est-il acceptable de travailler à mi-résolution ?
- Références intéressantes sur la vision:
  - OpenCV : http://opencv.itseez.com
  - Digital image processing (Gonzales and Woods)