Introduction à OpenCV

Design III : Intégration

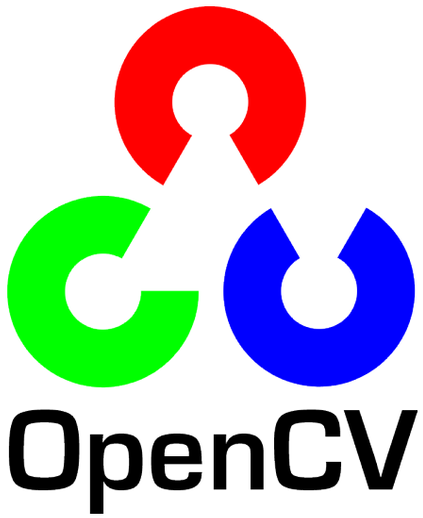
Marc-André Gardner
Yannick Hold-Geoffroy

Département de génie électrique, génie informatique
Faculté des sciences et de génie
Université Laval

Hiver 2017

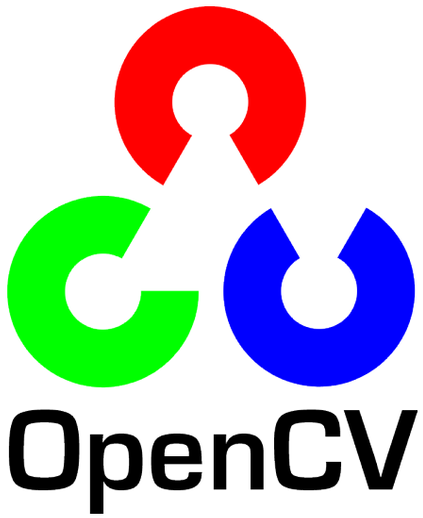
OpenCV, c'est...

- un cadriciel très puissant pour la vision numérique



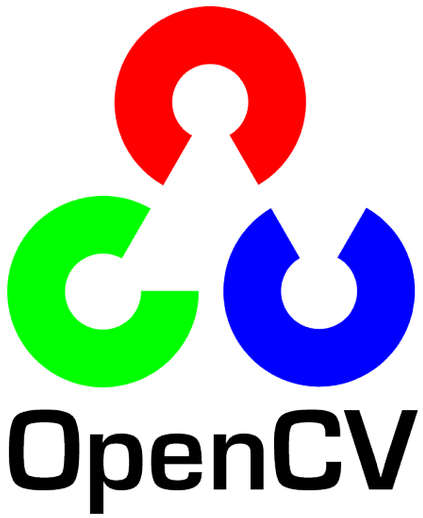
OpenCV, c'est...

- un cadriciel très puissant pour la vision numérique
- une compilation d'algorithmes utiles



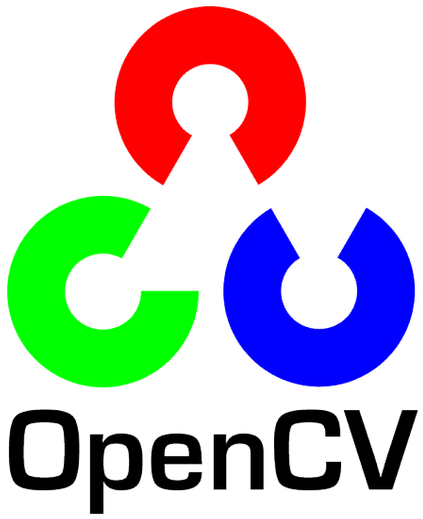
OpenCV, c'est...

- un cadriciel très puissant pour la vision numérique
- une compilation d'algorithmes utiles
- une implémentation optimisée pour un maximum de performance



OpenCV, c'est...

- un cadriciel très puissant pour la vision numérique
- une compilation d'algorithmes utiles
- une implémentation optimisée pour un maximum de performance
- une documentation bien écrite avec de nombreux exemples



OpenCV, ce n'est pas...

Un magicien !

- si vos images sont mauvaises, le résultat le sera aussi (garbage in, garbage out)

OpenCV, ce n'est pas...

Un magicien !

- si vos images sont mauvaises, le résultat le sera aussi (garbage in, garbage out)
- peu utile en dehors de la vision (asservissement des moteurs, intelligence artificielle...)

OpenCV, ce n'est pas...

Un magicien !

- si vos images sont mauvaises, le résultat le sera aussi (garbage in, garbage out)
- peu utile en dehors de la vision (asservissement des moteurs, intelligence artificielle...)
- traiter des images en 1600x1200 donnera un superbe résultat, mais au prix d'une surcharge de calcul importante

OpenCV, ce n'est pas...

Un magicien !

- si vos images sont mauvaises, le résultat le sera aussi (garbage in, garbage out)
- peu utile en dehors de la vision (asservissement des moteurs, intelligence artificielle...)
- traiter des images en 1600x1200 donnera un superbe résultat, mais au prix d'une surcharge de calcul importante
- si les mathématiques derrière vos algorithmes sont incorrectes, le résultat ne le sera pas non plus. Utiliser des algorithmes préimplémentés est intéressant, mais il faut tout de même comprendre ce qu'ils font...

Installation d'une version optimisée

- **OpenCV est préinstallé** en version 2.4.7 sur les images Fedora Linux qui vous sont fournies.

Installation d'une version optimisée

- **OpenCV est préinstallé** en version 2.4.7 sur les images Fedora Linux qui vous sont fournies.
- Il peut être intéressant d'utiliser une version personnalisée plus récente (gains en performances / fonctionnalités)

Installation d'une version optimisée

- **OpenCV est préinstallé** en version 2.4.7 sur les images Fedora Linux qui vous sont fournies.
- Il peut être intéressant d'utiliser une version personnalisée plus récente (gains en performances / fonctionnalités)
- Binaires disponibles pour Windows (Visual Studio)

Langages supportés

- OpenCV offre des bindings officiels pour les langages suivants : C, C++, Python et Java

Langages supportés

- OpenCV offre des bindings officiels pour les langages suivants : C, C++, Python et Java
- Les documentations C, C++ et Python sont communes, mais la documentation Java est séparée (`http://docs.opencv.org/java/`) (les interfaces sont légèrement différentes)

Langages supportés

- OpenCV offre des bindings officiels pour les langages suivants : C, C++, Python et Java
- Les documentations C, C++ et Python sont communes, mais la documentation Java est séparée (`http://docs.opencv.org/java/`) (les interfaces sont légèrement différentes)
- Il existe également le projet non officiel JavaCV (`https://github.com/bytedeco/javacv`)

Langages supportés

- OpenCV offre des bindings officiels pour les langages suivants : C, C++, Python et Java
- Les documentations C, C++ et Python sont communes, mais la documentation Java est séparée (`http://docs.opencv.org/java/`) (les interfaces sont légèrement différentes)
- Il existe également le projet non officiel JavaCV (`https://github.com/bytedeco/javacv`)
- Pour pouvoir avoir les bindings java, il faut **compiler une version personnalisée d'OpenCV**, en ayant préalablement installé *ant* et en ayant correctement décrit le chemin d'installation de Java avec la variable d'environnement $JAVA_HOME (voir document d'installation sur le site du cours)

USB Video-device Class et paramétrage

- UVC est un standard pour la vidéo sur USB ; il supporte un certain nombre de contrôles utiles au projet, en particulier *Exposure*, *Gain*, *White balance*, *Backlight compensation*, *Contrast* et *Brightness*.

USB Video-device Class et paramétrage

- UVC est un standard pour la vidéo sur USB ; il supporte un certain nombre de contrôles utiles au projet, en particulier *Exposure*, *Gain*, *White balance*, *Backlight compensation*, *Contrast* et *Brightness*.
- La Logitech C905 est configurée par défaut avec des réglages automatiques.

USB Video-device Class et paramétrage

- UVC est un standard pour la vidéo sur USB ; il supporte un certain nombre de contrôles utiles au projet, en particulier *Exposure*, *Gain*, *White balance*, *Backlight compensation*, *Contrast* et *Brightness*.
- La Logitech C905 est configurée par défaut avec des réglages automatiques.
- Ces paramètres peuvent être ajustés directement dans OpenCV (*VideoCapture.set* en C++ par exemple), ou avec un utilitaire en ligne de commande, **uvcdynctrl** (*yum install uvcdynctrl*). Le logiciel **guvcview** peut être utile pour constater l'impact des réglages.

Acquisition et lecture vidéo

- OpenCV permet une acquisition facile à partir de la caméra (en temps réel) :

```
import cv2
captObj = cv2.VideoCapture(CAMERA_ID) # 0 pour la 1ere camera, 1 pour la
      seconde...
assert captObj.isOpened(), "Erreur lors de l'ouverture de la camera!"
isFrameReturned, img = captObj.read()
```

Acquisition et lecture vidéo

- OpenCV permet une acquisition facile à partir de la caméra (en temps réel) :

```
import cv2
captObj = cv2.VideoCapture(CAMERA_ID) # 0 pour la 1ere camera, 1 pour la
      seconde...
assert captObj.isOpened(), "Erreur lors de l'ouverture de la camera!"
isFrameReturned, img = captObj.read()
```

- La même interface peut être utilisée pour lire un *fichier* :

```
import cv2
captObj = cv2.VideoCapture("ma_video.avi")
assert captObj.isOpened(), "Erreur lors de l'ouverture du fichier video!"
isFrameReturned, img = captObj.read()
```

Acquisition et lecture vidéo

- OpenCV permet une acquisition facile à partir de la caméra (en temps réel) :

```
import cv2
captObj = cv2.VideoCapture(CAMERA_ID) # 0 pour la 1ere camera, 1 pour la
      seconde...
assert captObj.isOpened(), "Erreur lors de l'ouverture de la camera!"
isFrameReturned, img = captObj.read()
```

- La même interface peut être utilisée pour lire un *fichier* :

```
import cv2
captObj = cv2.VideoCapture("ma_video.avi")
assert captObj.isOpened(), "Erreur lors de l'ouverture du fichier video!"
isFrameReturned, img = captObj.read()
```

- Dans tous les cas, la vidéo est traitée image par image (chaque appel à read() renvoie une nouvelle image). Si le traitement est plus rapide que le nombre d'images par seconde renvoyées par la caméra, ou si la fin du fichier vidéo est atteinte, read() retourne **False** pour l'indiquer.

Acquisition et lecture vidéo

- OpenCV permet une acquisition facile à partir de la caméra (en temps réel) :

```
import cv2
captObj = cv2.VideoCapture(CAMERA_ID) # 0 pour la 1ere camera, 1 pour la
      seconde...
assert captObj.isOpened(), "Erreur lors de l'ouverture de la camera!"
isFrameReturned, img = captObj.read()
```

- La même interface peut être utilisée pour lire un *fichier* :

```
import cv2
captObj = cv2.VideoCapture("ma_video.avi")
assert captObj.isOpened(), "Erreur lors de l'ouverture du fichier video!"
isFrameReturned, img = captObj.read()
```

- Dans tous les cas, la vidéo est traitée image par image (chaque appel à read() renvoie une nouvelle image). Si le traitement est plus rapide que le nombre d'images par seconde renvoyées par la caméra, ou si la fin du fichier vidéo est atteinte, read() retourne **False** pour l'indiquer.
- On peut aussi lire une seule image (JPG, PNG, etc.) :

```
import cv2
img = cv2.imread("mon_image.jpg")
```

Affichage et enregistrement

- OpenCV offre la possibilité d'afficher facilement une ou plusieurs images :

```
import cv2
img = cv2.imread("mon_image.jpg")
cv2.namedWindow("Affichage") # Le nom de la fenetre a creer
cv2.imshow("Affichage", img) # Afficher "img" dans la fenetre "Affichage"
cv2.waitKey()                # Tres important!
```

Affichage et enregistrement

- OpenCV offre la possibilité d'afficher facilement une ou plusieurs images :

```
import cv2
img = cv2.imread("mon_image.jpg")
cv2.namedWindow("Affichage") # Le nom de la fenetre a creer
cv2.imshow("Affichage", img) # Afficher "img" dans la fenetre "Affichage"
cv2.waitKey()                # Tres important!
```

- Il est **nécessaire** d'appeler la fonction waitKey pour que la fenêtre soit mise à jour !

Affichage et enregistrement

- OpenCV offre la possibilité d'afficher facilement une ou plusieurs images :

```
import cv2
img = cv2.imread("mon_image.jpg")
cv2.namedWindow("Affichage") # Le nom de la fenetre a creer
cv2.imshow("Affichage", img) # Afficher "img" dans la fenetre "Affichage"
cv2.waitKey()                # Tres important!
```

- Il est **nécessaire** d'appeler la fonction waitKey pour que la fenêtre soit mise à jour !
- On peut enregistrer une vidéo traitée par OpenCV à l'aide de la classe *VideoWriter* :

```
import cv2
captObj = cv2.VideoCapture(CAMERA_ID) # Ouverture de la camera
fourcc = cv2.VideoWriter_fourcc(*'XVID') # Preparation a enregistrer
writeObj = cv2.VideoWriter('fichier.avi', fourcc, fps=30, frameSize=(640, 480))
isFrameReturned, img = captObj.read()
while isFrameReturned == True: # Tant que la camera envoie des images
  writeObj.write(img) # Ecrire l'image
  isFrameReturned, img = captObj.read() # Demander une nouvelle image
```

Affichage et enregistrement

- OpenCV offre la possibilité d'afficher facilement une ou plusieurs images :

```
import cv2
img = cv2.imread("mon_image.jpg")
cv2.namedWindow("Affichage") # Le nom de la fenetre a creer
cv2.imshow("Affichage", img) # Afficher "img" dans la fenetre "Affichage"
cv2.waitKey()                # Tres important!
```

- Il est **nécessaire** d'appeler la fonction waitKey pour que la fenêtre soit mise à jour !
- On peut enregistrer une vidéo traitée par OpenCV à l'aide de la classe *VideoWriter* :

```
import cv2
captObj = cv2.VideoCapture(CAMERA_ID) # Ouverture de la camera
fourcc = cv2.VideoWriter_fourcc(*'XVID') # Preparation a enregistrer
writeObj = cv2.VideoWriter('fichier.avi', fourcc, fps=30, frameSize=(640, 480))
isFrameReturned, img = captObj.read()
while isFrameReturned == True: # Tant que la camera envoie des images
  writeObj.write(img) # Ecrire l'image
  isFrameReturned, img = captObj.read() # Demander une nouvelle image
```

- Il est aussi possible d'enregistrer une image avec *imwrite*.

Qu'est-ce qu'une image ?

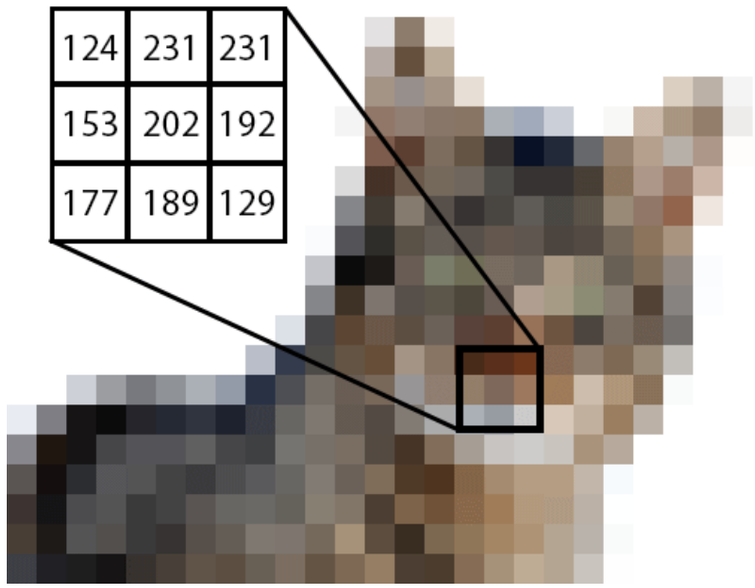
- Une image est une grille régulière de valeurs (pixels) ;

Qu'est-ce qu'une image ?

- Une image est une grille régulière de valeurs (pixels) ;
- Généralement composée de 1, 3 ou 4 canaux.

Qu'est-ce qu'une image ?

- Une image est une grille régulière de valeurs (pixels) ;
- Généralement composée de 1, 3 ou 4 canaux.



Segmentation

La segmentation est le partitionnement d'une image en multiples segments (groupes de pixels).

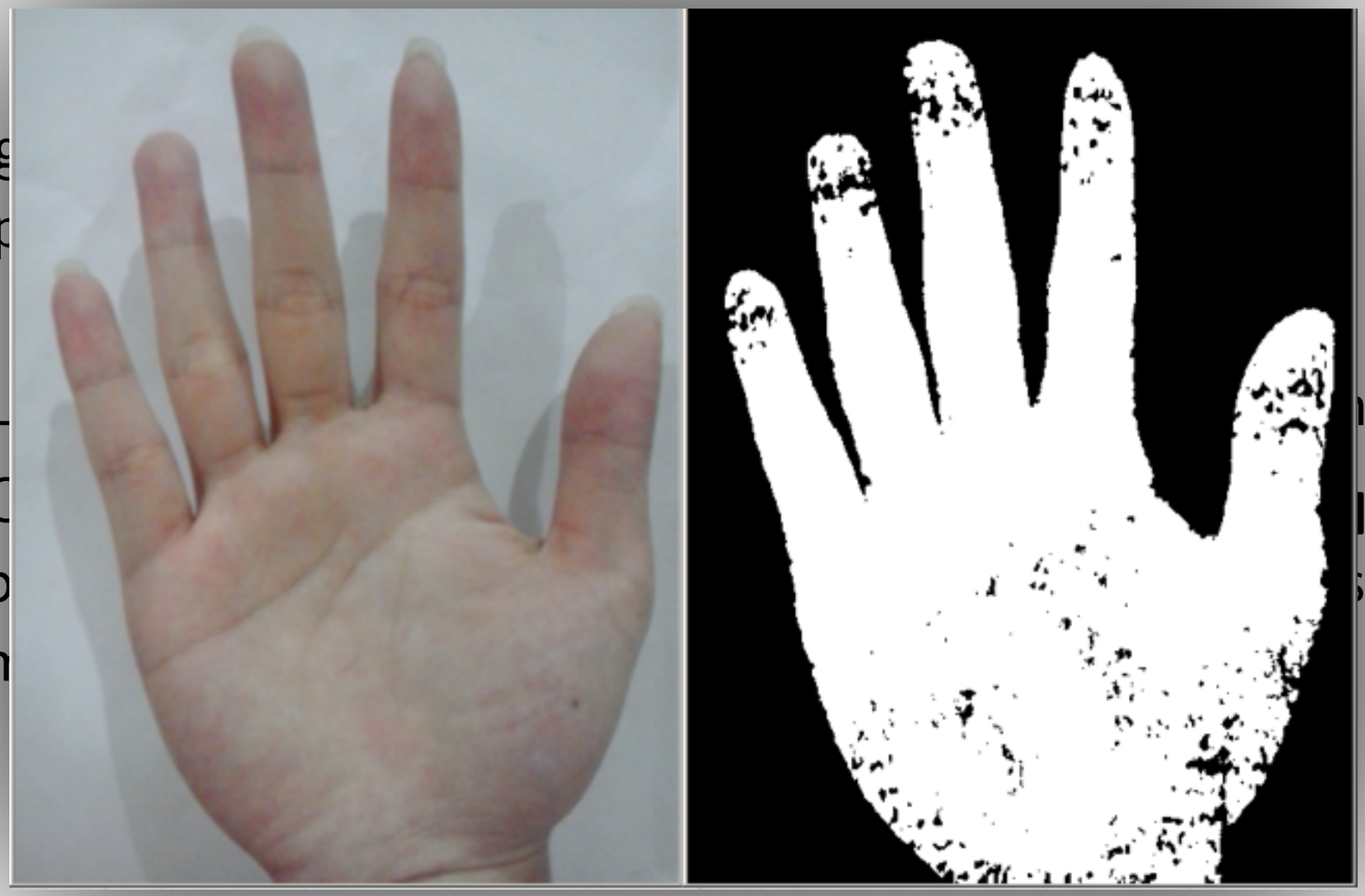
- Les fonctions *threshold* et *inRange* binarisent les images.

Segmentation

La segmentation est le partitionnement d'une image en multiples segments (groupes de pixels).

- Les fonctions *threshold* et *inRange* binarisent les images.
- Ces fonctions sont utilisées pour retourner un **masque** binaire qui ne conserve que les régions intéressantes : marqueurs sur le robot, forme à reproduire, etc.

Segmentation

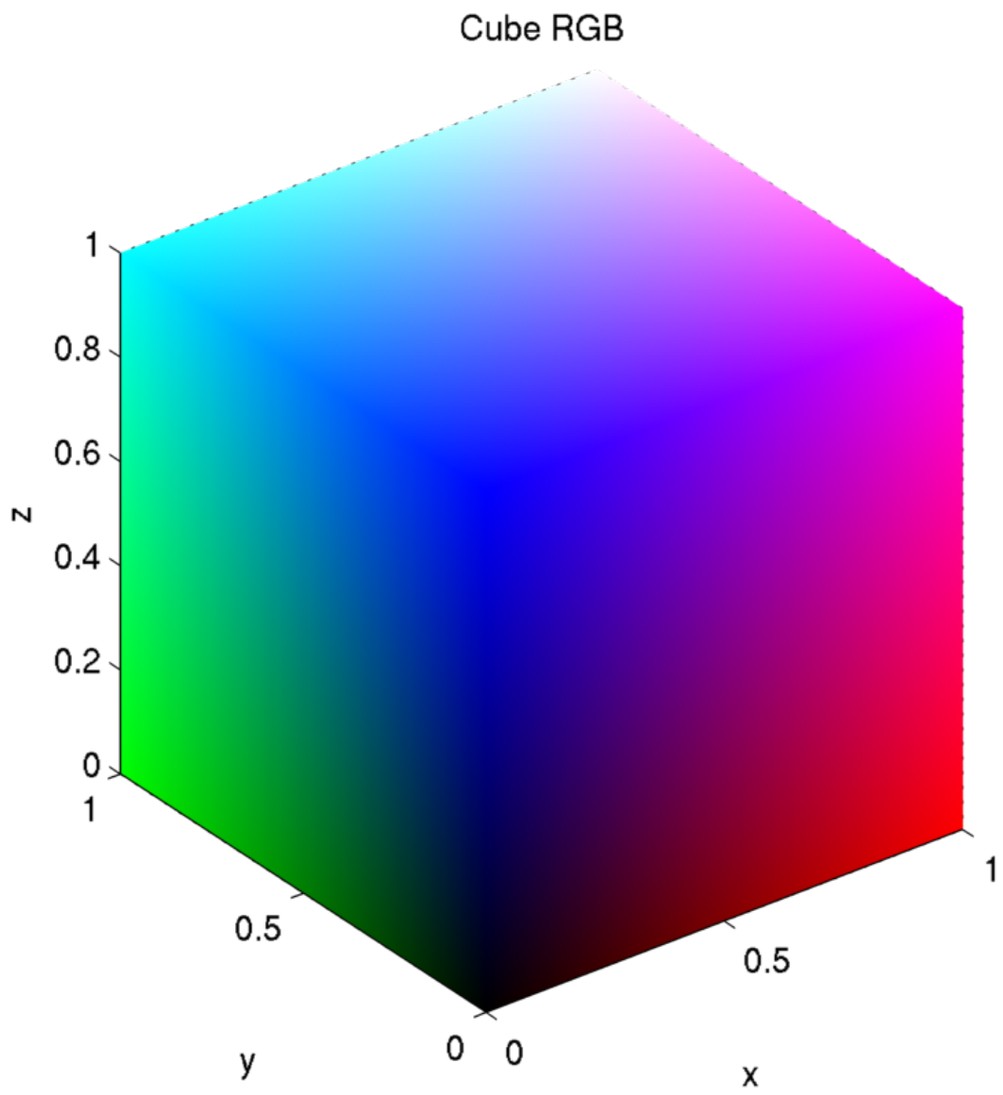


La seg
multip

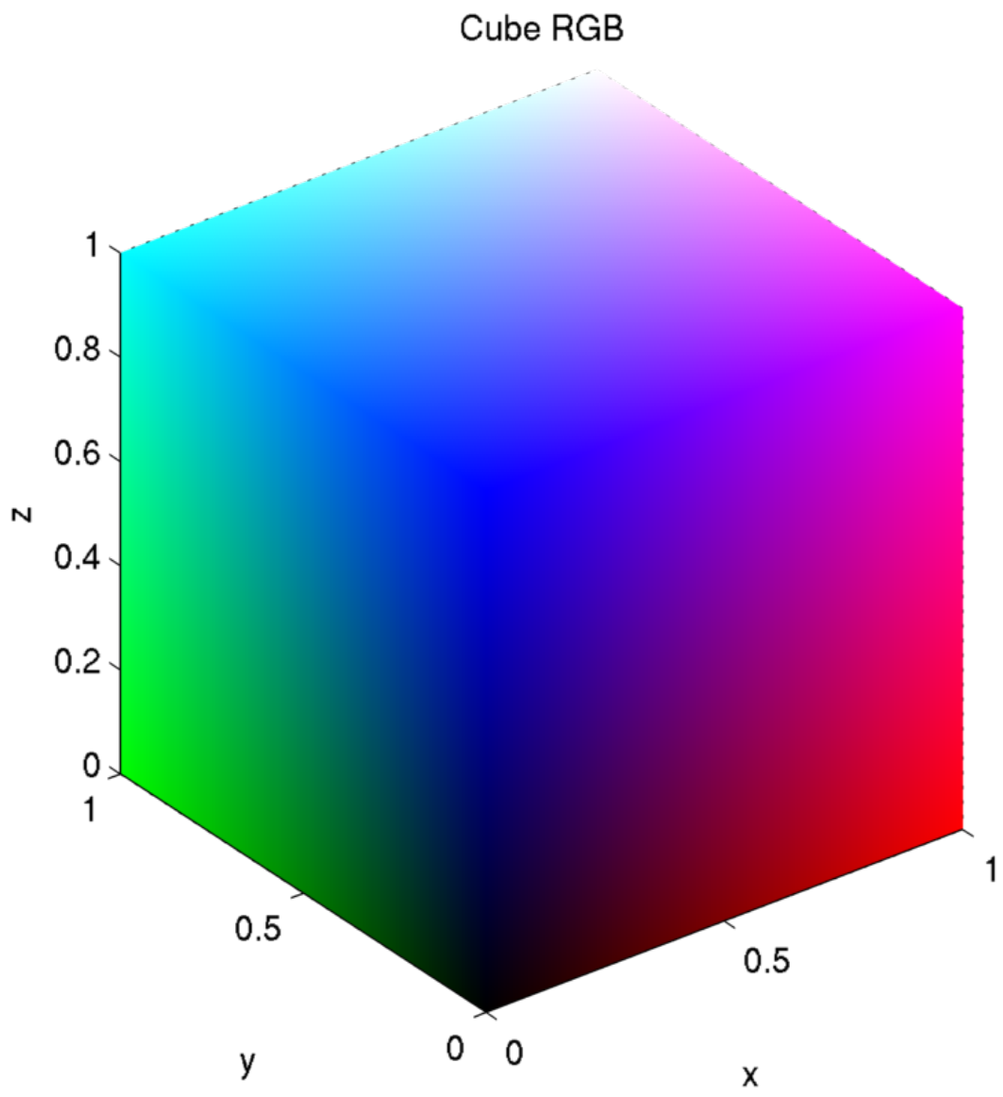
- L ages.
- C **que**
  b s :
  m

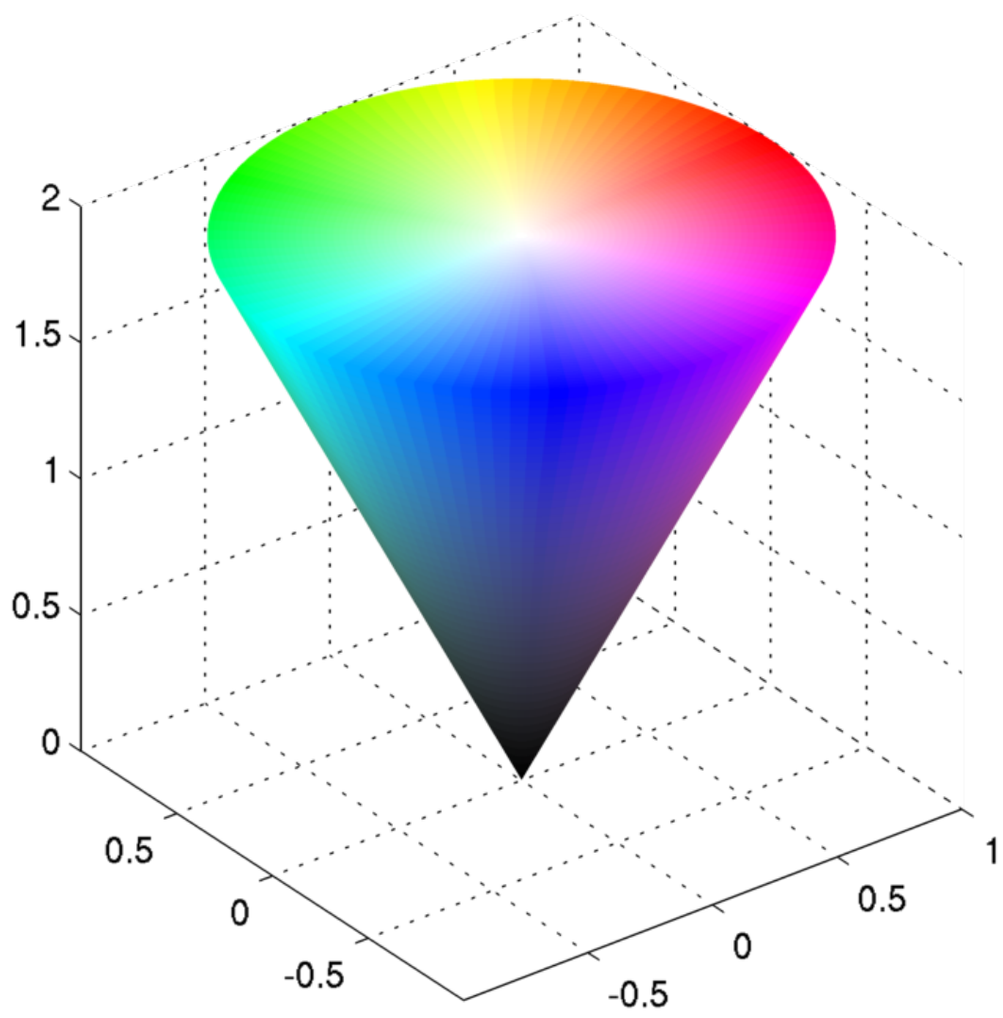
Image : http://answers.opencv.org/question/3300/skin-detection/

Changement d'espace colorimétrique

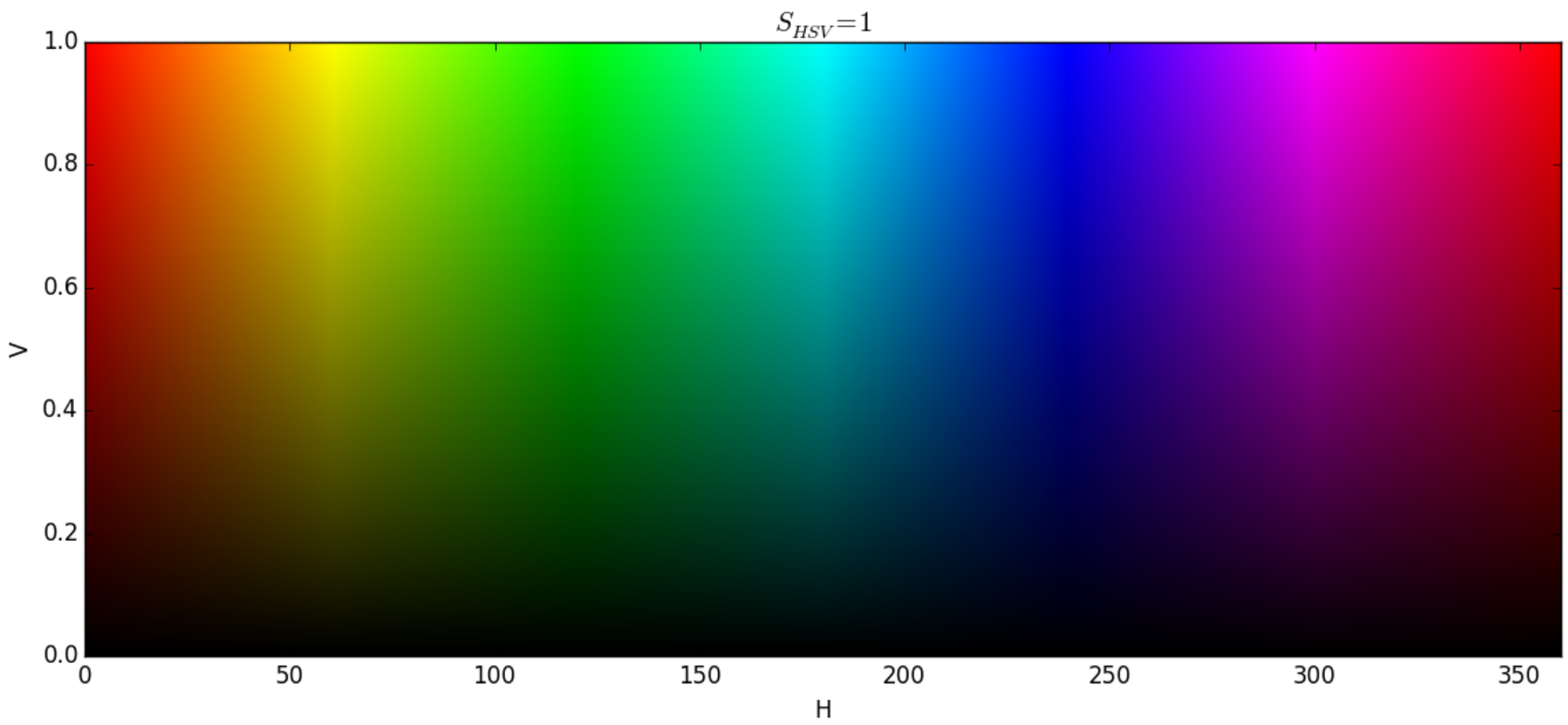


Changement d'espace colorimétrique





Changement d'espace colorimétrique



Changement d'espace colorimétrique

- La fonction *cvtColor* permet de passer d'un espace de couleurs à l'autre :

```
import cv2
img_bgr = cv2.imread("mon_image.jpg")
img_hsv = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2HSV)
img_bgr_again = cv2.cvtColor(img_hsv, cv2.COLOR_HSV2BGR)
img_gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
```

Attention !

- OpenCV encode la teinte sur 180 degrés au lieu de 360

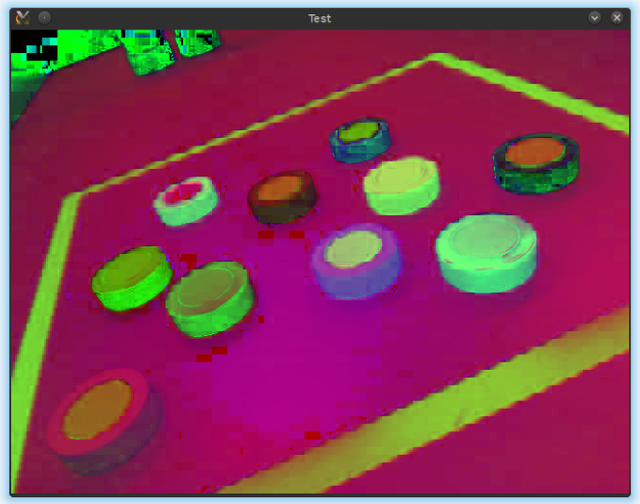
Changement d'espace colorimétrique

- La fonction *cvtColor* permet de passer d'un espace de couleurs à l'autre :

```
import cv2
img_bgr = cv2.imread("mon_image.jpg")
img_hsv = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2HSV)
img_bgr_again = cv2.cvtColor(img_hsv, cv2.COLOR_HSV2BGR)
img_gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
```

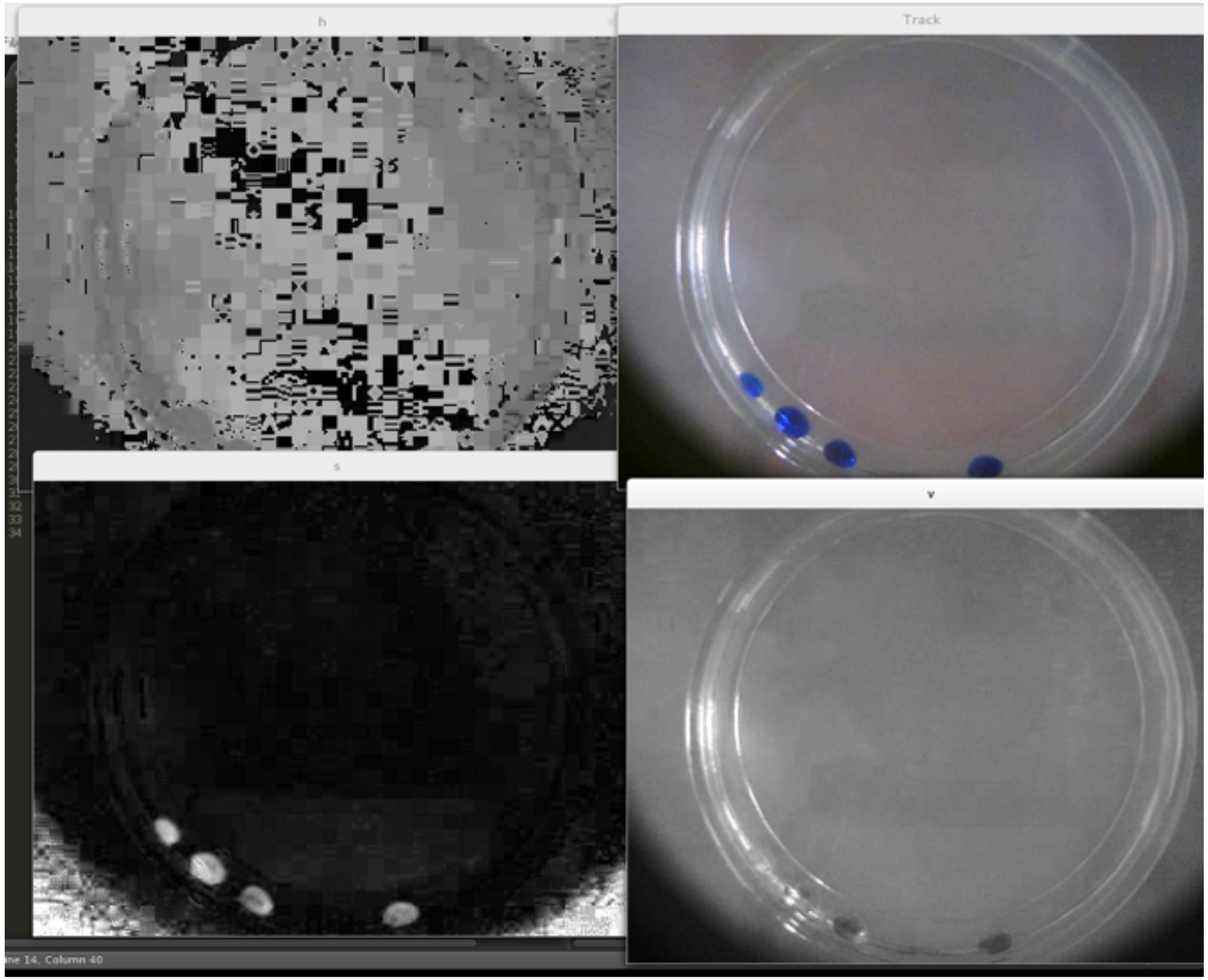
Attention !

- OpenCV encode la teinte sur 180 degrés au lieu de 360
- Attention à l'affichage : OpenCV suppose que l'image à afficher est toujours une image BGR !



Changement d'espace colorimétrique

H



RGB

S

V

image : http://stackoverflow.com/questions/17239253/opencv-bgr2hsv-creates-lots-of-artifacts/28201863

Segmentation par couleur

- Il est beaucoup plus simple de travailler en HSV pour segmenter selon la couleur.

Segmentation par couleur

- Il est beaucoup plus simple de travailler en HSV pour segmenter selon la couleur.
- Deux paramètres principaux pour chaque composant : la valeur recherchée et la tolérance sur celle-ci.

Segmentation par couleur

- Il est beaucoup plus simple de travailler en HSV pour segmenter selon la couleur.
- Deux paramètres principaux pour chaque composant : la valeur recherchée et la tolérance sur celle-ci.

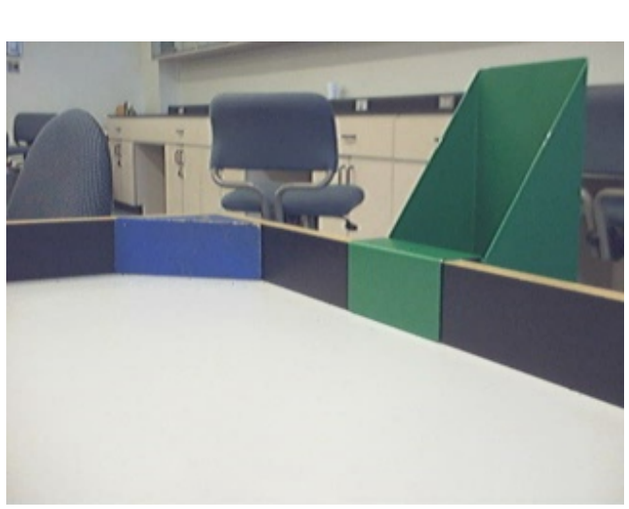


Image originale

Segmentation par couleur

- Il est beaucoup plus simple de travailler en HSV pour segmenter selon la couleur.
- Deux paramètres principaux pour chaque composant : la valeur recherchée et la tolérance sur celle-ci.

Image originale

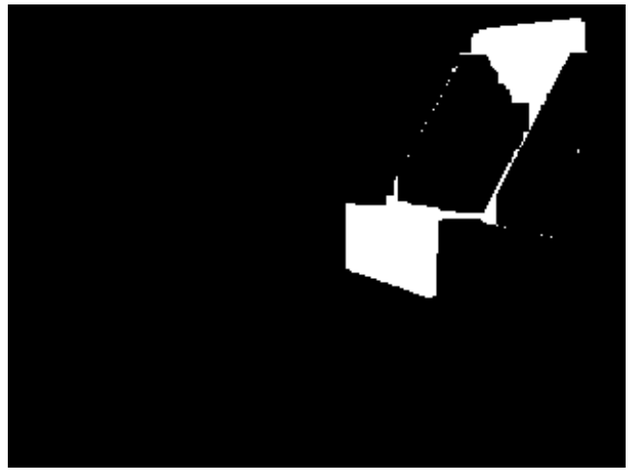
Segmentée (bleu)

Segmentation par couleur

- Il est beaucoup plus simple de travailler en HSV pour segmenter selon la couleur.
- Deux paramètres principaux pour chaque composant : la valeur recherchée et la tolérance sur celle-ci.

Image originale

Segmentée (bleu)



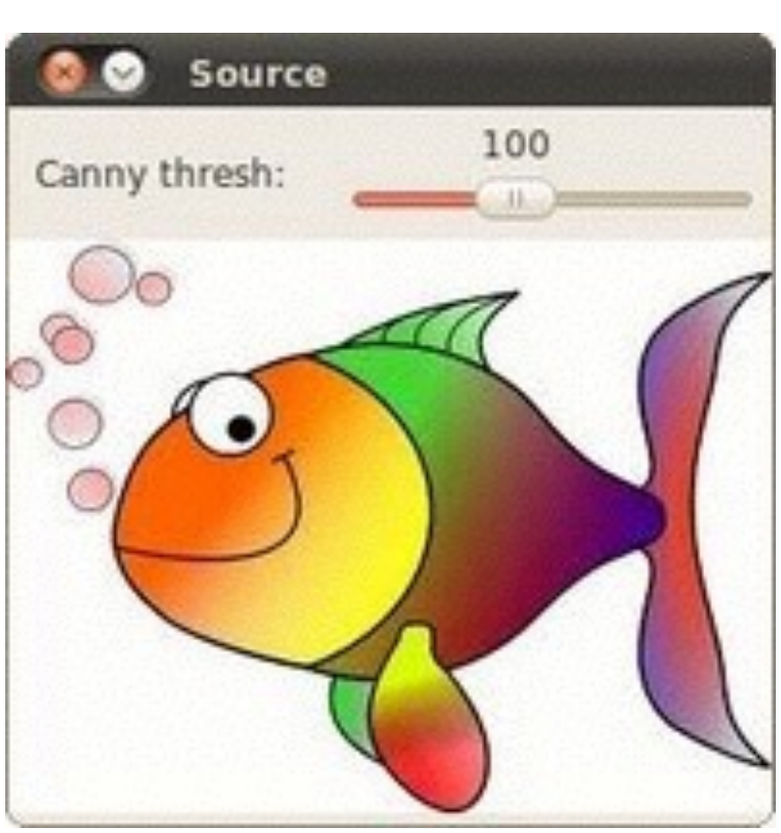
Segmentée (vert)

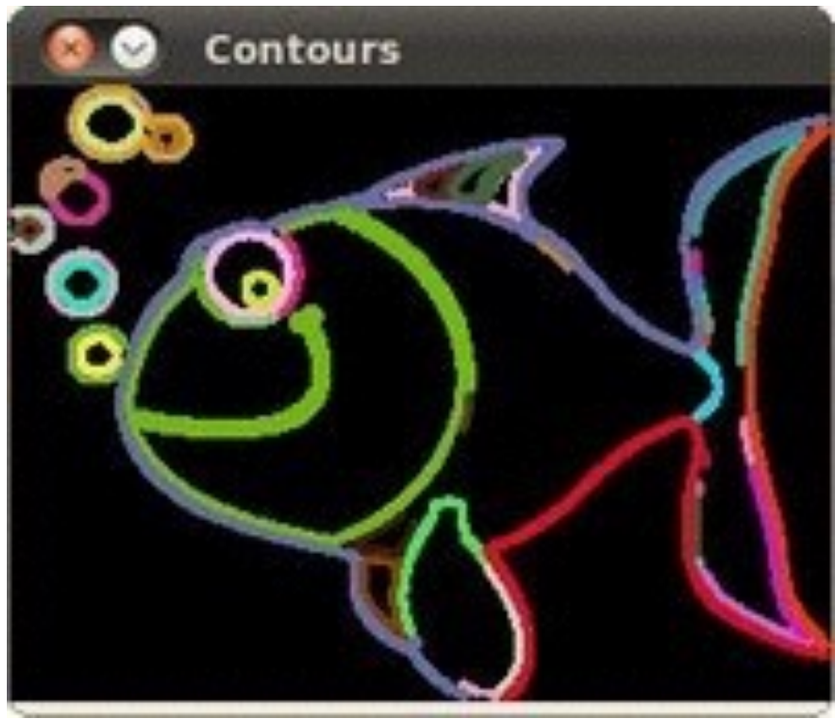
Étiquetage et englobement de forme

- *findContours* retourne la liste des séquences de pixels représentant les contours des pixels groupés

Étiquetage et englobement de forme

- *findContours* retourne la liste des séquences de pixels représentant les contours des pixels groupés



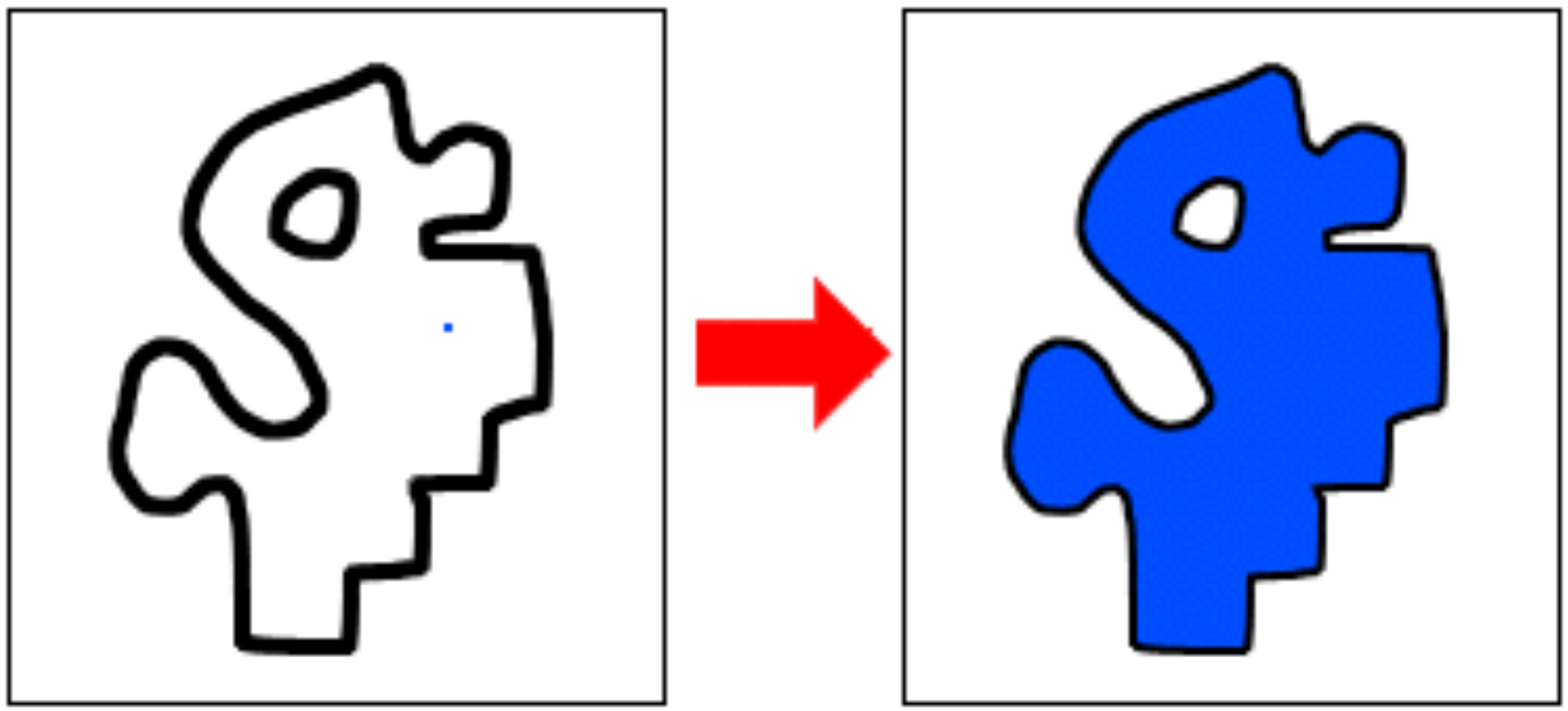


Étiquetage et englobement de forme

- *findContours* retourne la liste des séquences de pixels représentant les contours des pixels groupés
- *floodFill* colorie tous les pixels connectés à un pixel initial

Étiquetage et englobement de forme

- *findContours* retourne la liste des séquences de pixels représentant les contours des pixels groupés
- *floodFill* colorie tous les pixels connectés à un pixel initial



Étiquetage et englobement de forme

- *findContours* retourne la liste des séquences de pixels représentant les contours des pixels groupés
- *floodFill* colorie tous les pixels connectés à un pixel initial
- *connectedComponents* identifie et groupe les pixels

Étiquetage et englobement de forme

- *findContours* retourne la liste des séquences de pixels représentant les contours des pixels groupés
- *floodFill* colorie tous les pixels connectés à un pixel initial
- *connectedComponents* identifie et groupe les pixels

OpenCV peut trouver la forme géométrique englobante des groupes de pixels :

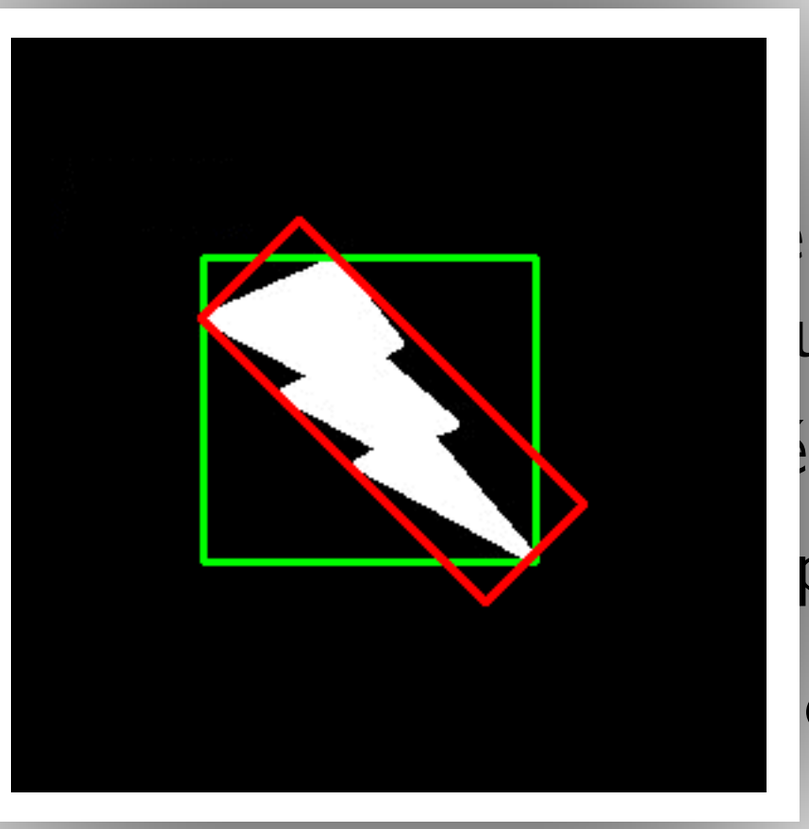
Étiquetage et englobement de forme

- *findContours* retourne la liste des séquences de pixels représentant les contours des pixels groupés
- *floodFill* colorie tous les pixels connectés à un pixel initial
- *connectedComponents* identifie et groupe les pixels

OpenCV peut trouver la forme géométrique englobante des groupes de pixels :

- *boundingRect* : rectangle englobant

Étiquetage et englobement de forme



- *findContours* r ences de pixels représentant le upés
- *floodFill* color s à un pixel initial
- *connectedCom* pe les pixels

OpenCV peut trou englobante des groupes de pixels :

- *boundingRect* : rectangle englobant

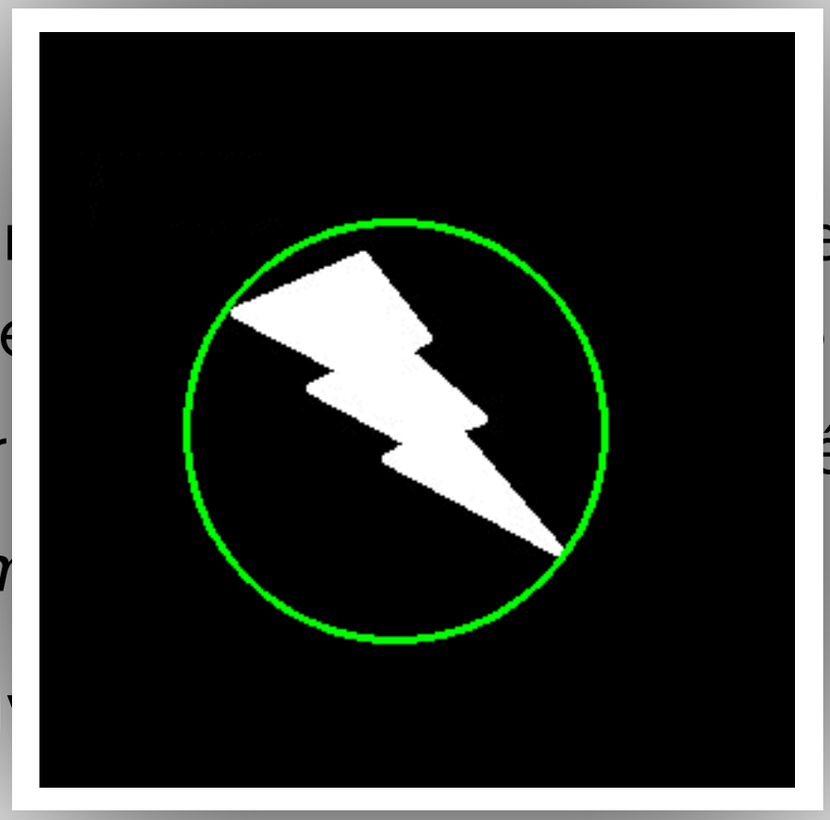
Étiquetage et englobement de forme

- *findContours* retourne la liste des séquences de pixels représentant les contours des pixels groupés
- *floodFill* colorie tous les pixels connectés à un pixel initial
- *connectedComponents* identifie et groupe les pixels

OpenCV peut trouver la forme géométrique englobante des groupes de pixels :

- *boundingRect* : rectangle englobant
- *minEnclosingCircle* : cercle de rayon minimal

Étiquetage et englobement de forme



- *findContours* r ences de pixels représentant le upés
- *floodFill* color és à un pixel initial
- *connectedCom* pe les pixels

OpenCV peut trou englobante des groupes de pixels :

- *boundingRect* : rectangle englobant
- *minEnclosingCircle* : cercle de rayon minimal

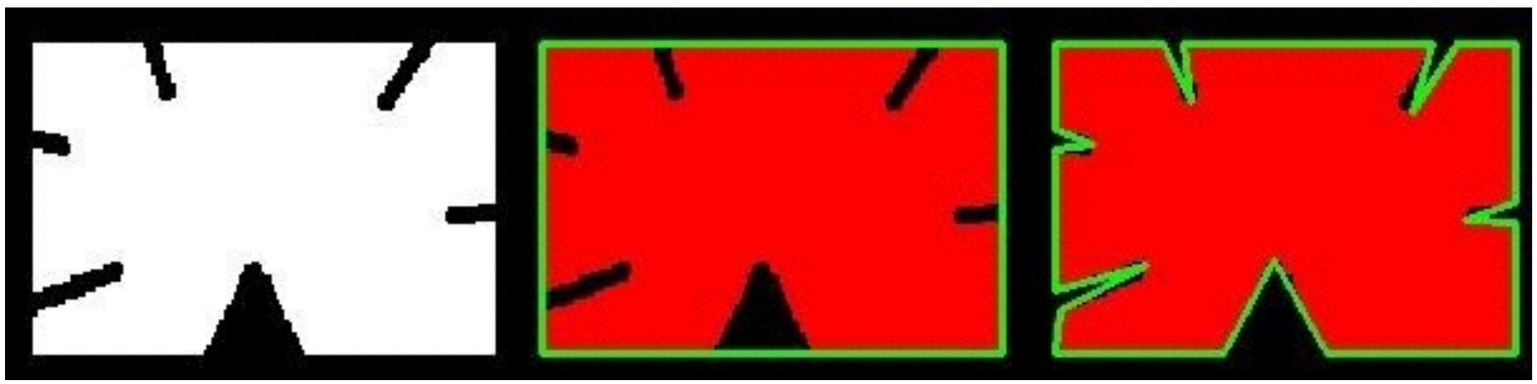
Étiquetage et englobement de forme

- *findContours* retourne la liste des séquences de pixels représentant les contours des pixels groupés
- *floodFill* colorie tous les pixels connectés à un pixel initial
- *connectedComponents* identifie et groupe les pixels

OpenCV peut trouver la forme géométrique englobante des groupes de pixels :

- *boundingRect* : rectangle englobant
- *minEnclosingCircle* : cercle de rayon minimal
- *approxPolyDP* : polygone arbitraire

Étiquetage et englobement de forme



- *fi* ... els re
- *fl* ... initial
- *co* ... s

OpenCV peut trouver la forme géométrique englobante des groupes de pixels :

- *boundingRect* : rectangle englobant
- *minEnclosingCircle* : cercle de rayon minimal
- *approxPolyDP* : polygone arbitraire

Traitements au niveau des pixels et filtrage

- Les fonctions *dilate* et *erode* sont disponibles.

Astuce

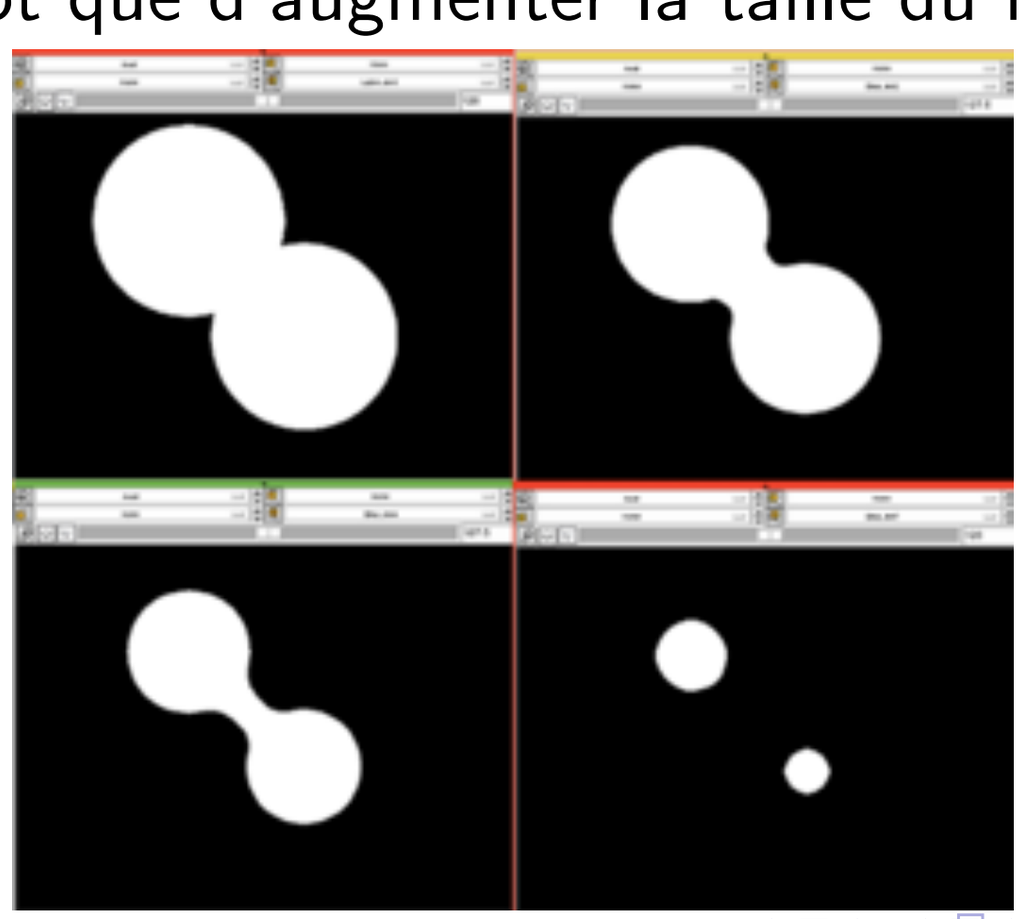
Il vaut parfois mieux faire plusieurs **itérations** d'érosion ou de dilatation plutôt que d'augmenter la taille du noyau.

Traitements au niveau des pixels et filtrage

- Les fonctions *dilate* et *erode* sont disponibles.

Astuce

Il vaut parfois mieux faire plusieurs **itérations** d'érosion ou de dilatation plutôt que d'augmenter la taille du noyau.



Traitements au niveau des pixels et filtrage

- Les fonctions *dilate* et *erode* sont disponibles.

Astuce

Il vaut parfois mieux faire plusieurs **itérations** d'érosion ou de dilatation plutôt que d'augmenter la taille du noyau.

- On peut obtenir ouverture et fermeture en combinant *erode* et *dilate* dans l'ordre voulu (par exemple dilate(erode(img)) pour une ouverture).

Traitements au niveau des pixels et filtrage

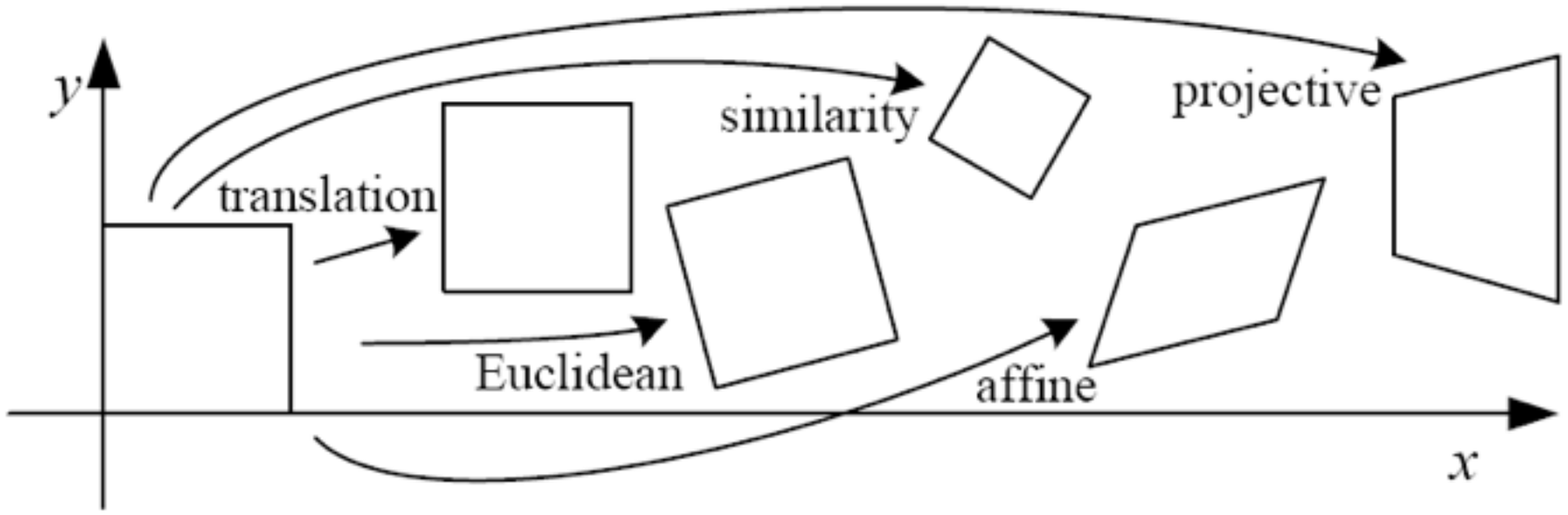
- Les fonctions *dilate* et *erode* sont disponibles.

Astuce

Il vaut parfois mieux faire plusieurs **itérations** d'érosion ou de dilatation plutôt que d'augmenter la taille du noyau.

- On peut obtenir ouverture et fermeture en combinant *erode* et *dilate* dans l'ordre voulu (par exemple dilate(erode(img)) pour une ouverture).
- Il est possible de faire un filtrage avec un filtre moyenneur standard (fonction *blur*), un filtre à noyau gaussien (fonction *gaussianBlur*), etc.

Transformations géométriques



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	

Merci à Jean-François Lalonde

Transformations géométriques

Type	Estimer la transformée	Calculer les Coordonnées	Appliquer la transformée à une image
Rotation	getRotationMatrix2D()	transform()	warpAffine()
Affine	getAffineTransform()	transform()	warpAffine()
Perspective	getPerspectiveTransform()	perspectiveTransform()	warpPerspective()

Transformations géométriques

1. Obtenir des coordonnées source et destination
2. M = getPerspectiveTransform(pt_src, pt_dst)
3. sortie = warpPerspective(image_src, M, taille)

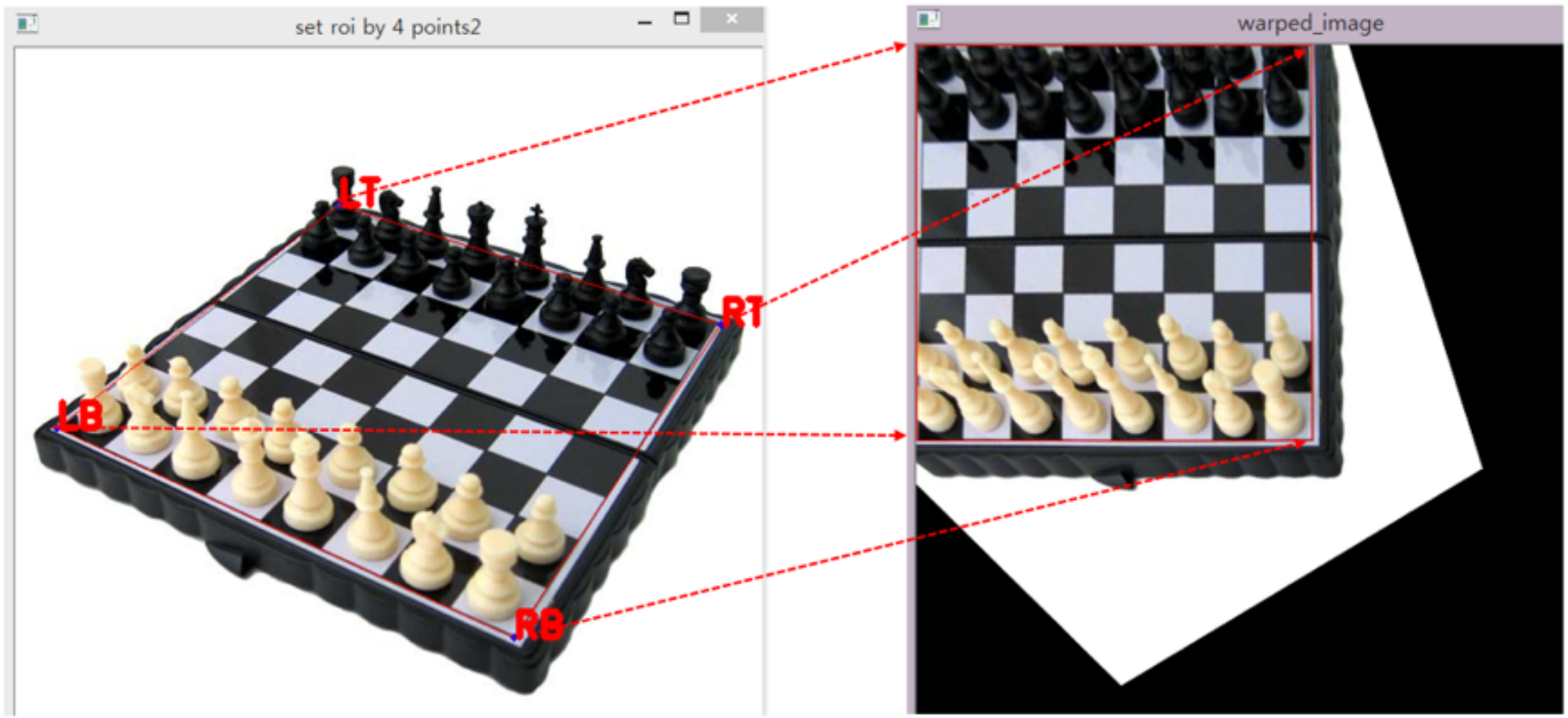


Image: http://study.marearts.com/2015/03/image-warping-using-opencv.html

Pour en savoir plus

- Il existe une multitude d'autres fonctions utiles...

Pour en savoir plus

- Il existe une multitude d'autres fonctions utiles...
- Une excellente référence : *Learning OpenCV : Computer Vision with the OpenCV Library* (disponible en ligne à la bibliothèque de l'Université Laval)

Pour en savoir plus

- Il existe une multitude d'autres fonctions utiles...
- Une excellente référence : *Learning OpenCV : Computer Vision with the OpenCV Library* (disponible en ligne à la bibliothèque de l'Université Laval)
- **Fouillez dans la documentation !**

Pour en savoir plus

- Il existe une multitude d'autres fonctions utiles...
- Une excellente référence : *Learning OpenCV : Computer Vision with the OpenCV Library* (disponible en ligne à la bibliothèque de l'Université Laval)
- **Fouillez dans la documentation** !
- Inspirez-vous des exemples et tutoriels fournis :

http://docs.opencv.org/

http://github.com/opencv/opencv/tree/master/samples