

# M1 ENJMIN - MJV102

Conception et développement informatique

2 à 6 oct. 2017

Pierre Cubaud      cubaud @ cnam.fr

Viviane Gal        gal @ cnam.fr

## 5. Captation



# Le son avec la librairie Minim

## Exemple d'un oscilloscope



```
import ddf.minim.*;  
  
Minim minim;  
AudioInput in;  
float lemin, lemax;  
float[] X = new float[1024];  
  
void setup(){  
  size(800, 600);  
  minim = new Minim(this);  
  in = minim.getLineIn  
    (Minim.MONO, 1024, 44100);  
}
```

**(monscope.pde)**

```
void draw() {  
  background(15,50,20);  
  fill(255);  
  line(0,height/2,width,height/2);  
  stroke(255);  
  X = in.left.toArray();  
  for(int i = 0; i < X.length-1; i++) {  
    float h1 = map(X[i],-1,+1,height,0);  
    float h2 = map(X[i+1],-1,+1,height,0);  
    line(i, h1, i+1, h2);  
  }  
}  
  
void stop() {  
  in.close();  
  minim.stop();  
  super.stop();  
}
```

# Bill Viola

Bill Viola: The Quintet of Remembrance (2001.395a-l) | Heilbrunn Timeline of Art History | The Metropolitan Museum of Art


http://www.metmuseum.org/TOAH/ho/11/na/ho\_2001.395a-l.htm

HEILBRUNN TIMELINE OF ART HISTORY

WORLD MAPS | TIMELINES/REGIONS | THEMATIC ESSAYS | WORKS OF ART | INDEX | BIBLIOGRAPHY

MMA Search

World Map • United States and Canada, 1900 A.D.



**The Quintet of Remembrance, 2000**  
Bill Viola (American, born 1951)  
Video installation; color video rear-projected on large screen in darkened room; Room (ideal dimensions): 12 x 18 x 24 (36.6 x 54.9 x 73.2 cm); screen: 57 x 99 in. (144.8 x 251.5 cm); Purchase, Lila Acheson Wallace Gift, 2001 (2001.395a-l)

Video art was first introduced in the early 1960s by such pioneers as June Paik and Bruce Nauman and continues to be a vital form of contemporary artistic expression. Since the early 1970s, it has been Viola's primary medium and today he is considered one of America's preeminent video artists. His dialogue with art history evolved during the 1990s in a series of videos that make reference to the narratives, human figures, and portraits in well-known works of art. *The Quintet of Remembrance* is one of four videos created between 2000 and 2001 that were inspired by the artist's study of late medieval and early Renaissance Italian and Flemish paintings and their iconography. In each, a group of five people undergo a range of emotions while the camera records the nuance of their physical reaction.

Here, Viola specifically references Hieronymus Bosch's *Christ Mocked* (*The Crowning with Thorns*) (ca. 1490–1500, National Gallery, London) and Andrea Mantegna's *Adoration of the Magi* (1495–1505, Getty Museum).

# demo videodelay

```
videodelay | Processing 1.5.1  
videodelay  
// d'... processing/  
import  
Capture  
int co  
int co  
  
int n  
int cu  
PImage  
  
void s  
size  
myCa  
fran  
for  
fr  
}  
}  
  
void draw() {  
  if (myCap.available()) {  
    myCap.read();  
    PImage img = myCap.getImage();  
    image(img, 0, 0, img.width, img.height);  
  }  
}
```

Display 0 does not exist, using the default display instead.

1

```
// d'apres http://www.flong.com/blog/2011/free-video-delay-in-processing/
```

```
import processing.video.*;
```

```
Capture myCap;
```

```
int capW = 640; //320;
```

```
int capH = 480; //240;
```

```
int nDelayFrames = 50; //100; // about 3 seconds
```

```
int currentFrame = nDelayFrames-1;
```

```
PImage frames[];
```

```
void setup() {
```

```
    size (640, 480);
```

```
    myCap = new Capture(this, capW, capH);
```

```
    frames = new PImage[nDelayFrames];
```

```
    for (int i=0; i<nDelayFrames; i++) {
```

```
        frames[i] = createImage(capW, capH, ARGB);
```

```
    }
```

```
}
```

```
void draw() {
```

```
    if (myCap.available()) {
```

```
        myCap.read();
```

```
        frames[currentFrame].loadPixels();
```

```
        arrayCopy (myCap.pixels, frames[currentFrame].pixels);
```

```
        frames[currentFrame].updatePixels();
```

```
        currentFrame = (currentFrame-1 + nDelayFrames) % nDelayFrames;
```

```
    }
```

```
    image (frames[currentFrame], 0, 0, width, height);
```

```
}
```

## capture de l'écran

capture

```
import processing.video.*;
MovieMaker mm; // Declare MovieMaker object

void setup() {
  size(600, 600);
  mm = new MovieMaker(this, 600, 600, "resultat.mov");
}

void draw() {
  ellipse(mouseX, mouseY, 50, 50);
  mm.addFrame(); // Add window's pixels to movie
}

void keyPressed() {
  mm.finish(); // Finish the movie
}
```



## Tracking video (vraiment) minimal



**Préalable :**

**se fabriquer une cible**

**mais**

**pourquoi en rouge ?**



trackVideo

```
import processing.video.*;
Capture video;

void setup(){
  size(640, 480);
  video = new Capture(this, width, height, 30);
  video.start(); // pour Processing 2.0 et +
  noStroke();
  smooth();
}
void draw() {
if (video.available()){
  float hmax = 0;
  int tx = 0;
  int ty = 0;
  video.read();
  video.loadPixels();
  int index = 0;
  for (int y = 0; y < video.height; y++) {
    for (int x = 0; x < video.width; x++) {
      float h = hue(video.pixels[index]);
      // la teinte max donne du rouge
      if (h > hmax){
        hmax = h; tx = x; ty = y;
      }
      index++;
    }
  }
  // feedback graphique
  image(video, 0, 0, width, height);
  fill(255, 255, 0, 128);
  ellipse(tx, ty, 100, 100);
}
}
```

- étudier l'impact de l'appel  
**video.available()**

- enlever **image()** et transformer  
le dessin en un curseur qui  
évolue dans le sens  
du mouvement :  
comme un miroir

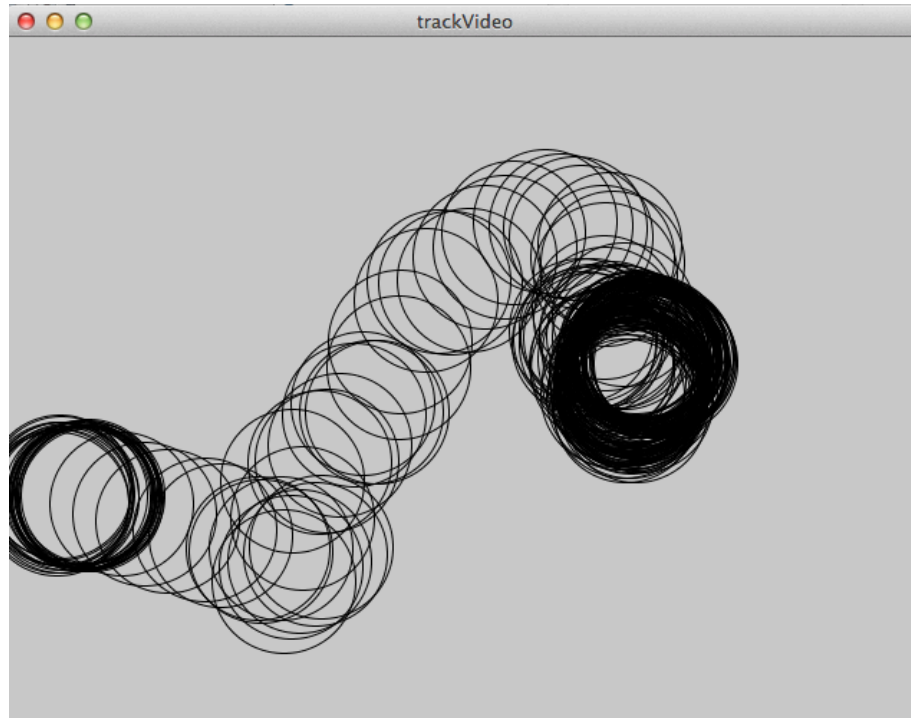


```
import processing.video.*;
Capture video;
int tx = 0;
int ty = 0;
float lemax = 0;

void setup() {
  size(640, 480);
  video = new Capture(this, width, height, 30);
  video.start();
  noStroke();
  smooth();
}
```

```
void draw() {
  if (video.available()) {
    video.read();
    video.loadPixels();
    int index = 0;
    for (int y = 0; y < video.height; y++) {
      for (int x = 0; x < video.width; x++){
        float h = hue(video.pixels[index]);
        // la teinte max donne du rouge
        if (h > lemax){
          lemax = h;
          tx = x;
          ty = y;
        }
        index++;
      }
    }
  }
  // feedback graphique
  image(video, 0, 0, width, height);
  fill(255, 255, 0, 128);
  ellipse(tx, ty, 100, 100);
}
}
```

**Utilisable comme curseur pour dessiner ?**



**problème de bruit ! (tremblements, aléa de l'analyse)**

**Une solution pour "retirer" le bruit de captation :  
utiliser un filtre passe-bas**

**variante simple très utilisée : le filtre alpha**

$$y_{n+1} = (1 - \alpha) \cdot y_n + \alpha \cdot x_{n+1}$$

**nouvelle  
estimation**

**ancienne  
estimation**

**nouvelle  
mesure**

**si alpha fort (>0.50) : réponse rapide mais bruitée  
sinon : moins de bruit mais inertie**

**=> il faut tester (ici alpha = 30% = 0.3 pas mal)**

## modification du code :

```
import processing.video.*;
Capture video;
float txe = 0;
float tye = 0;
float alpha = 0.30;
```

← nouvelles déclarations au début du code

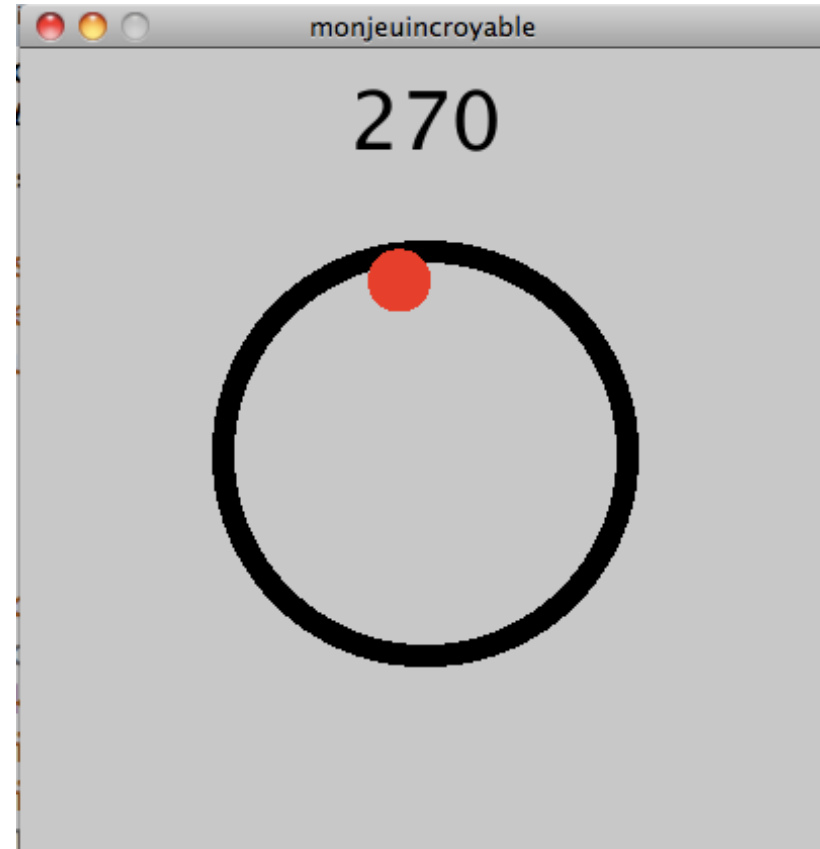
```
}
// feedback graphique
image(video, 0, 0, width, height);
txe = (1-alpha)*txe + alpha*tx;
tye = (1-alpha)*tye + alpha*ty;
stroke(255,255,0);
ellipse(txe, tye, 100, 100);
}
}
```

← à la fin, filtrage avant le tracé



- tester avec des alpha forts et faibles
- gestion de l'echec de tracking ????

## Mon jeu incroyable (v1)



```
score = 1000;  
while (curseur_pas_sur_la_cible()) {  
    score--;  
    if (score<0) noLoop();  
}
```



```
int score=300;  
int RAYON = 100;  
int x,y;
```

```
void setup(){  
  size(400,400);  
  noCursor();  
  x = width/2;  
  y = height/2-100;  
}
```

```
void draw(){  
  // objet guide  
  background(200);  
  noFill();stroke(0);strokeWeight(10);  
  ellipse(width/2,height/2,2*RAYON,2*RAYON);  
  // le pointeur  
  x = mouseX;  
  y = mouseY;  
  fill(250,0,0);noStroke();  
  ellipse(x+random(-10,10),y+random(-10,10),30,30);  
  // le score  
  float d = sqrt((x-width/2)*(x-width/2)+(y-height/2)*(y-height/2));  
  if ((d < RAYON-5)||d > RAYON+5) score--;  
  fill(0);textSize(40);textAlign(CENTER);text(str(score),width/2,50);  
  // arret du jeu  
  if (score==0) noLoop();  
}
```

**pour  
simuler  
le capteur**



monjeuVIDEO

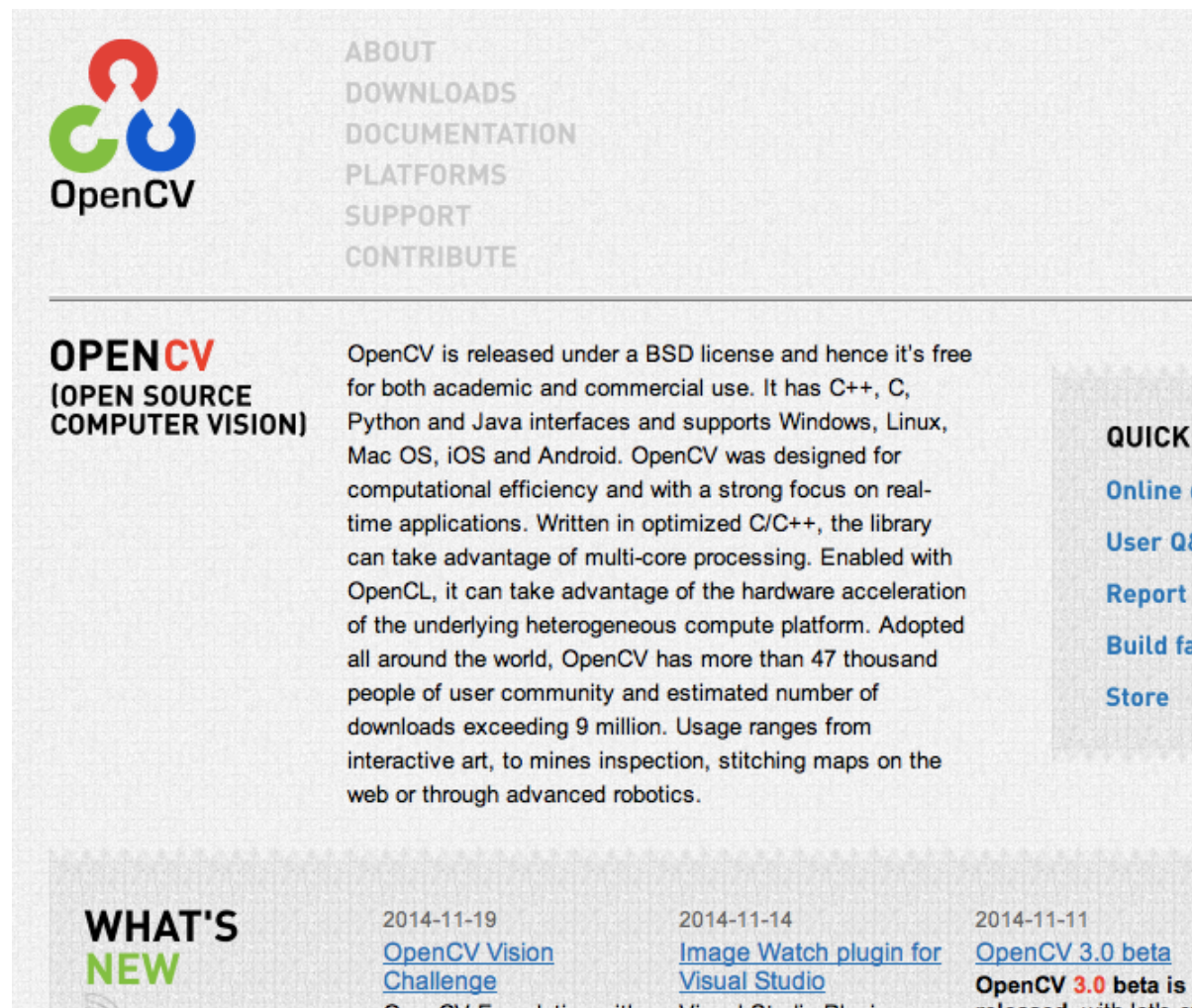
```
int score=30000;
int RAYON = 100;
int x,y;
import processing.video.*;
Capture video;
float txe = 0;
float tye = 0;
float alpha = 0.10;

void setup(){
  size(400,400);
  noCursor();
  video = new Capture(this, width, height, 30);
  //video.start(); // pour Processing 2.0 et +
  x = width/2;
  y = height/2-100;
}
```

```
void draw(){
  background(200);
  //image(video, 0, 0, width, height);
  // objet guide
  noFill();stroke(0);strokeWeight(10);
  ellipse(width/2,height/2,2*RAYON,2*RAYON);
  // le pointeur
  curseurVideo();
  x = width-(int)txe;
  y = (int)tye;
  fill(250,0,0);noStroke();
  ellipse(x,y,30,30);
  // le score
  float d = sqrt((x-width/2)*(x-width/2)+(y-height/2)*(y-height/2));
  if ((d < RAYON-5)||(d > RAYON+5)) score--;
  fill(0);textSize(40);textAlign(CENTER);text(str(score),width/2,50);
  // arret du jeu
  if (score==0) noLoop();
}
```

```
void curseurVideo(){
    float hmax = 0;
    int tx = 0;
    int ty = 0;
    video.read();
    video.loadPixels();
    int index = 0;
    for (int y = 0; y < video.height; y++) {
        for (int x = 0; x < video.width; x++) {
            float h = hue(video.pixels[index]);
            // la teinte max donne du rouge
            if (h > hmax){
                hmax = h; tx = x; ty = y;
            }
            index++;
        }
    }
    txe = (1-alpha)*txe + alpha*tx;
    tye = (1-alpha)*tye + alpha*ty;
}
```

opencv.org



The screenshot shows the OpenCV website homepage. At the top left is the OpenCV logo, which consists of three interlocking circles in red, green, and blue, with the text "OpenCV" below it. To the right of the logo is a vertical navigation menu with the following items: ABOUT, DOWNLOADS, DOCUMENTATION, PLATFORMS, SUPPORT, and CONTRIBUTE. Below the navigation menu is a horizontal line. Underneath the line, on the left, is the text "OPENCV (OPEN SOURCE COMPUTER VISION)". To the right of this text is a large paragraph of text describing OpenCV: "OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 9 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics." To the right of this paragraph is a vertical sidebar with the heading "QUICK" and several links: "Online d", "User Q&", "Report a", "Build fa", and "Store". At the bottom of the page, there is a section titled "WHAT'S NEW" with three columns of news items. The first column has the date "2014-11-19" and the link "OpenCV Vision Challenge". The second column has the date "2014-11-14" and the link "Image Watch plugin for Visual Studio". The third column has the date "2014-11-11" and the link "OpenCV 3.0 beta is released with lots of".

OpenCV

ABOUT  
DOWNLOADS  
DOCUMENTATION  
PLATFORMS  
SUPPORT  
CONTRIBUTE

**OPENCV**  
(OPEN SOURCE  
COMPUTER VISION)

OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 9 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

QUICK

Online d  
User Q&  
Report a  
Build fa  
Store

WHAT'S  
NEW

2014-11-19  
[OpenCV Vision Challenge](#)

2014-11-14  
[Image Watch plugin for Visual Studio](#)

2014-11-11  
[OpenCV 3.0 beta is released with lots of](#)

librairie créée chez Intel en Russie en 1999-2000

# installation pour Processing

<https://github.com/atduskgreg/opencv-processing>

GitHub This repository Search Enterprise Blog Sign up Sign in

atduskgreg / opencv-processing 358 Stars 105 Forks

OpenCV for Processing. A creative coding computer vision library based on the official OpenCV Java API

137 commits 3 branches 14 releases 8 contributors

branch: master opencv-processing / +

add link to javadoc in readme

atduskgreg authored Dec 8, 2014 latest commit 0fa983e972

data	Updating for working export on mac os, working on widows, and easiest...	Jul 17, 2013
examples	update getSnapshot() so it returns a PImage corresponding to the ROI ...	Dec 8, 2014
lib	fixed rpath issues for linux .so files	Nov 8, 2014
resources	update getSnapshot() so it returns a PImage corresponding to the ROI ...	Dec 8, 2014
src/gab/opencv	rewrite intro comment to OpenCV class for documentation	Dec 8, 2014
web	initial commit	Mar 31, 2013
.classpath	removed local include from classpath	Nov 8, 2014
.gitignore	add libs back to gitignore, remove local paths from classpath	Nov 4, 2014
.project	Replace instances of OpenCVPro in readme.	Jul 9, 2013

Code Issues Pull Requests Pulse Graphs

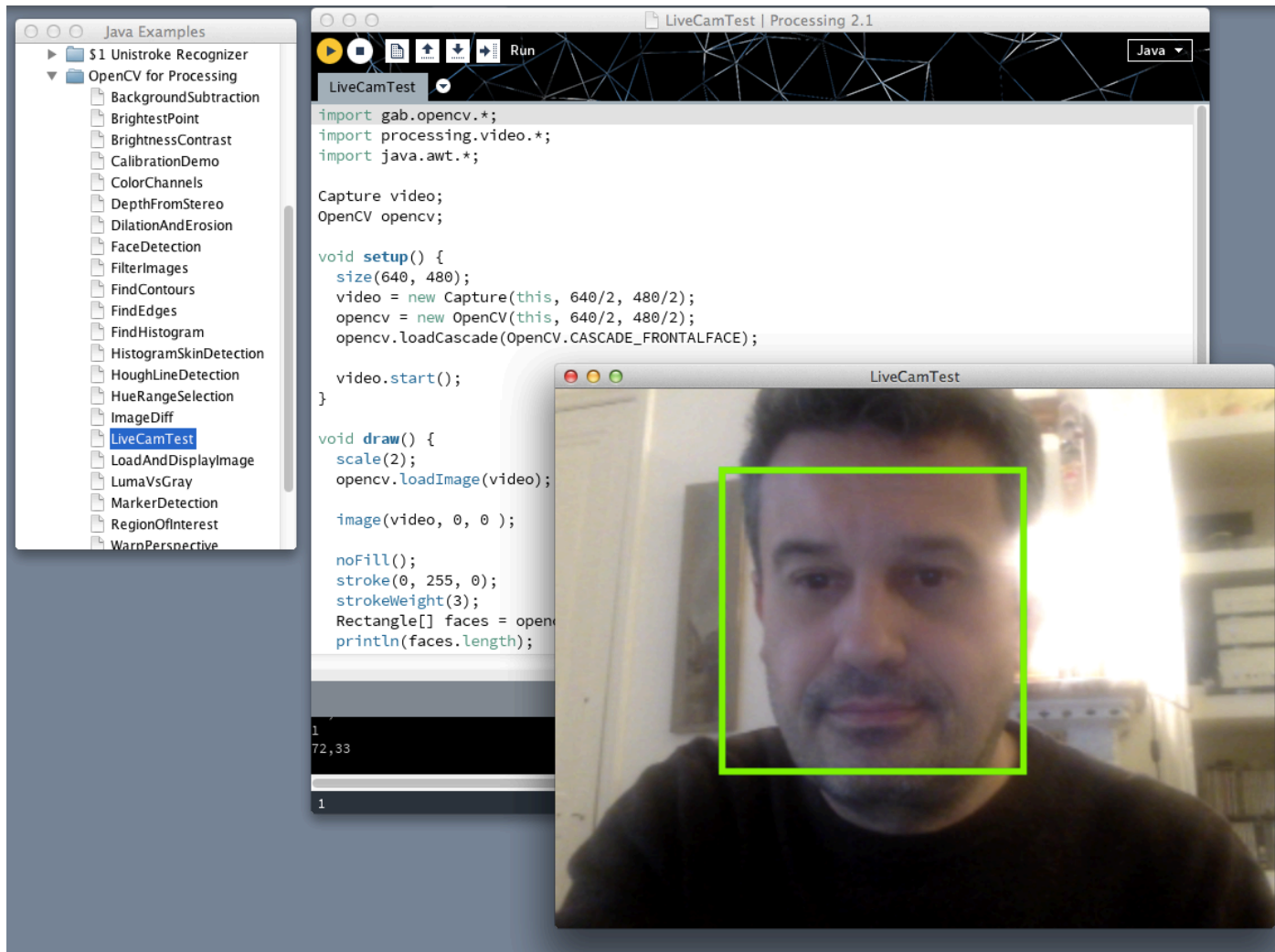
HTTPS clone URL  
https://github.com/atduskgreg/opencv-processing

You can clone with HTTPS or Subversion.

Clone in Desktop Download ZIP

remarque : il existe aussi un portage plus ancien fait aux Beaux-Arts d'Aix

# Test de la reconnaissance faciale (demo "LiveCamTest")



# Viola & Jones, 2001-4



WIKIPÉDIA  
L'encyclopédie libre

[Accueil](#)  
[Portails thématiques](#)  
[Article au hasard](#)  
[Contact](#)

[Contribuer](#)  
[Débuter sur Wikipédia](#)  
[Aide](#)  
[Communauté](#)  
[Modifications récentes](#)  
[Faire un don](#)

[Imprimer / exporter](#)  
[Créer un livre](#)  
[Télécharger comme PDF](#)  
[Version imprimable](#)

[Outils](#)  
[Pages liées](#)  
[Suivi des pages liées](#)  
[Importer un fichier](#)

[Créer un compte](#) [Se connecter](#)

Article [Discussion](#)

Lire [Modifier le code](#) [Historique](#)

## Méthode de Viola et Jones

★ Vous lisez un « [article de qualité](#) ».

La **méthode de Viola et Jones** est une méthode de [détection d'objet](#) dans une [image numérique](#), proposée par les chercheurs Paul Viola et Michael Jones en 2001. Elle fait partie des toutes premières méthodes capables de détecter efficacement et en temps réel des objets dans une image. Inventée à l'origine pour [détecter des visages](#), elle peut également être utilisée pour détecter d'autres types d'objets comme des [voitures](#) ou des [avions](#). La méthode de Viola et Jones est l'une des méthodes les plus connues et les plus utilisées, en particulier pour la [détection de visages](#) et la [détection de personnes](#).

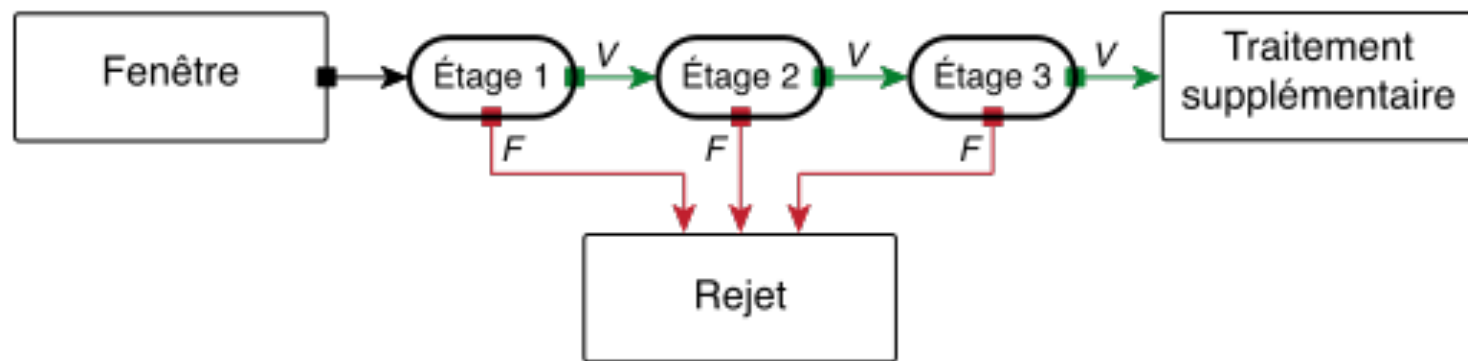
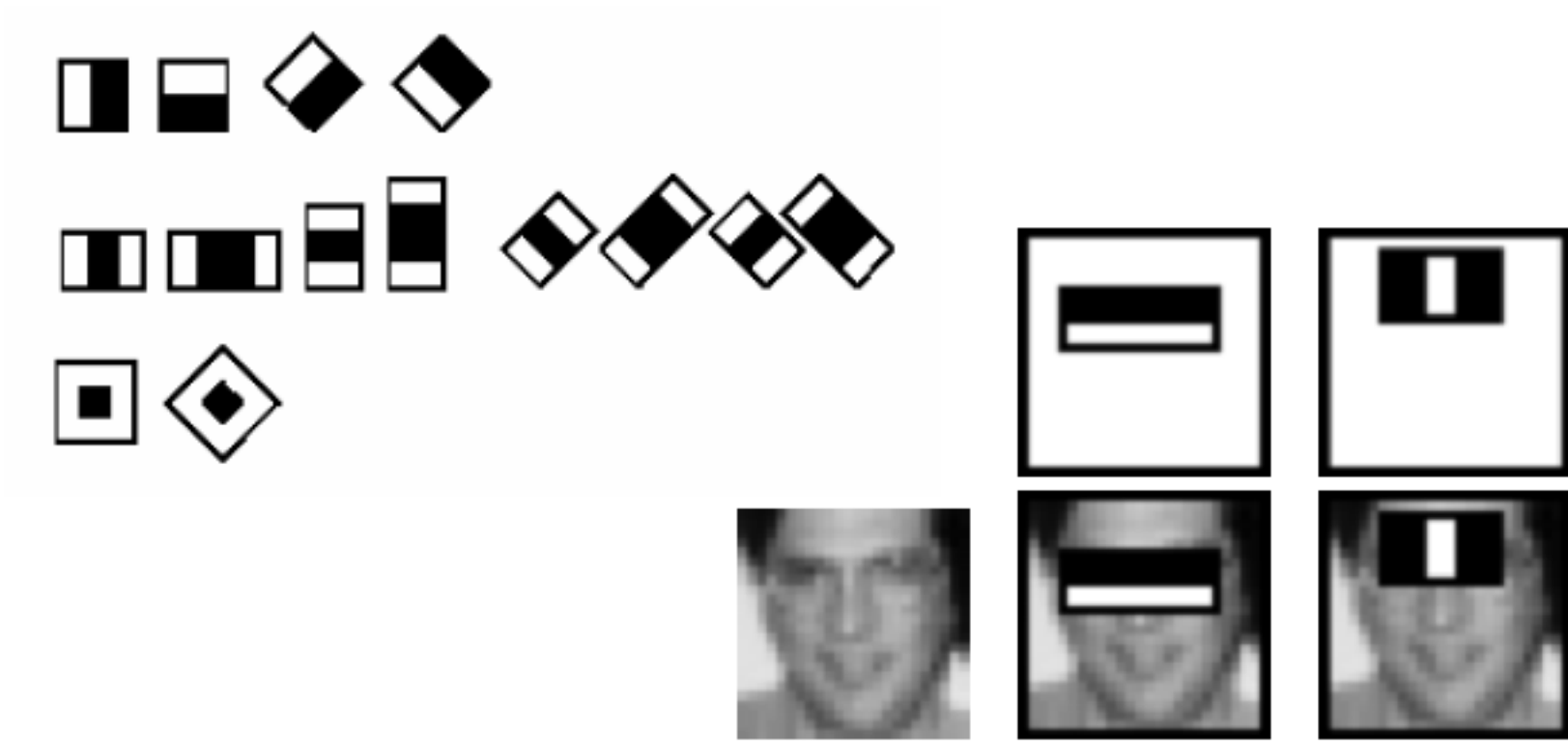
En tant que procédé d'[apprentissage supervisé](#), la méthode de Viola et Jones nécessite de quelques centaines à plusieurs milliers d'exemples de l'objet que l'on souhaite détecter, pour entraîner un [classifieur](#). Une fois son apprentissage réalisé, ce classifieur est utilisé pour détecter la présence éventuelle de l'objet dans une image en parcourant celle-ci de manière exhaustive, à toutes les positions et dans toutes les tailles possibles.

Considérée comme étant l'une des plus importantes méthodes de détection d'objet, la méthode de Viola et Jones est notamment connue pour avoir introduit plusieurs notions reprises ensuite par de nombreux chercheurs en [vision par ordinateur](#), à l'exemple de la notion d'[image intégrale](#) ou de la méthode de [classification](#) construite

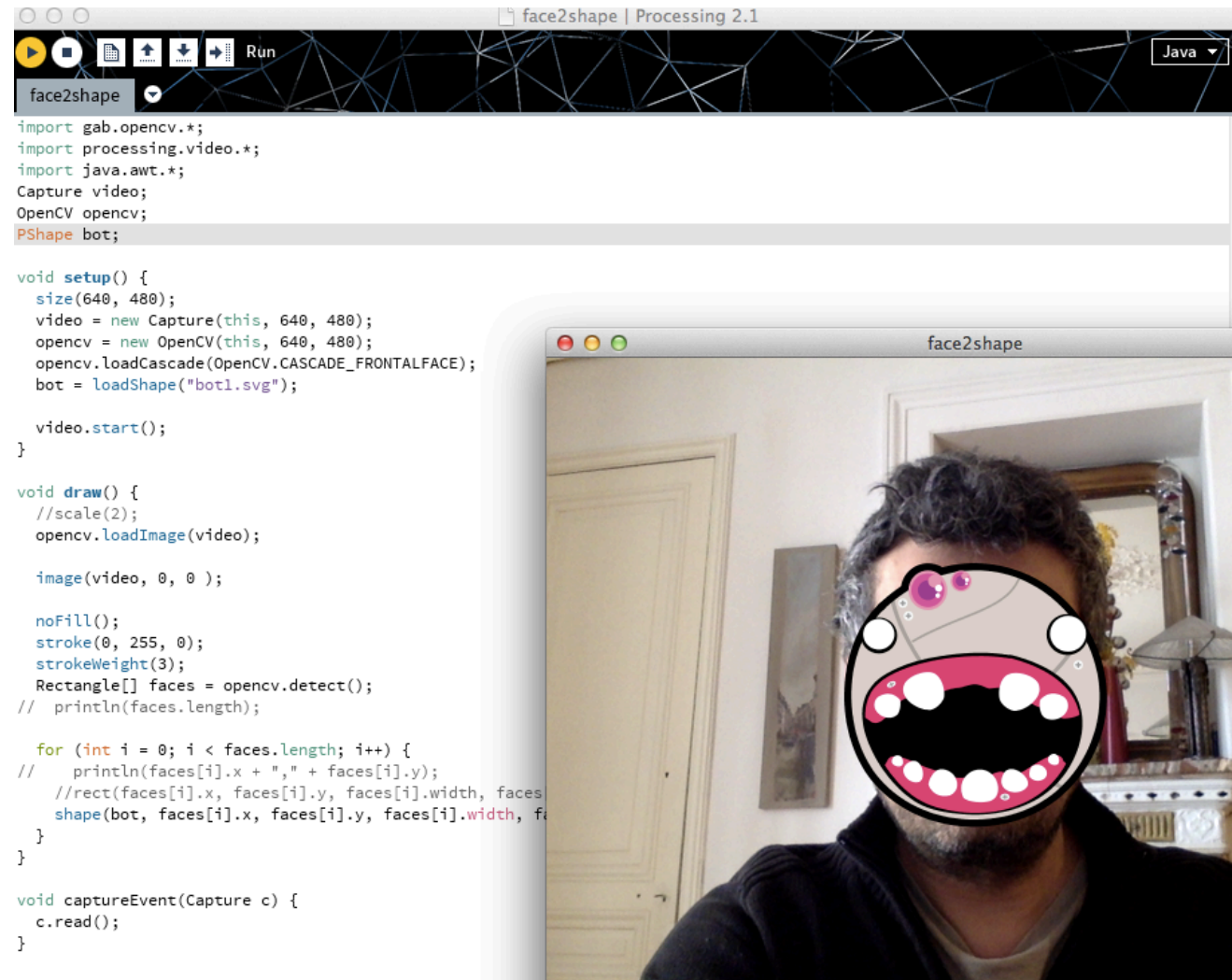


Un exemple de [détection de visage](#) par la méthode de Viola et Jones.





# Une variante ...



# marche aussi avec une image fixe

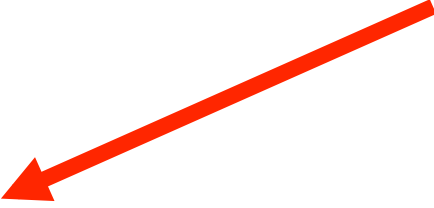


```
import gab.opencv.*;
import java.awt.Rectangle;

OpenCV opencv;
Rectangle[] faces;

void setup() {
  opencv = new OpenCV(this, "test.jpg");
  size(opencv.width, opencv.height);
  opencv.loadCascade(OpenCV.CASCADE_FRONTALFACE);
  faces = opencv.detect();
}

void draw() {
  image(opencv.getInput(), 0, 0);
  noFill(); stroke(0, 255, 0); strokeWidth(3);
  for (int i = 0; i < faces.length; i++) {
    rect(faces[i].x, faces[i].y, faces[i].width, faces[i].height);
  }
}
```



## Variante pour les yeux

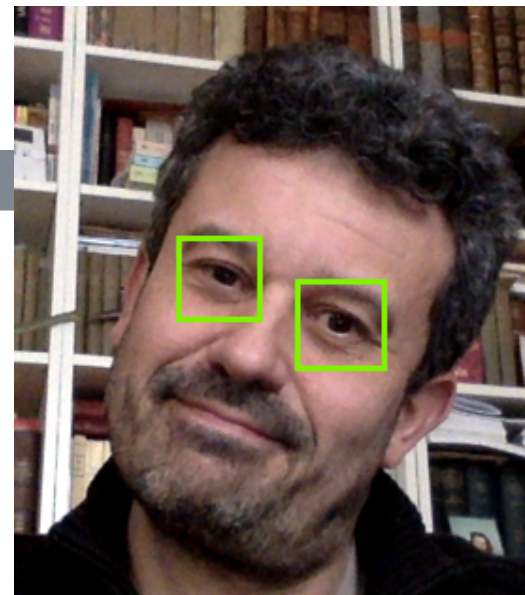
EyeDetectionCAM

```
Capture video;  
OpenCV opencv;
```

```
void setup() {  
  size(640, 480);  
  video = new Capture(this, 640, 480);  
  opencv = new OpenCV(this, 640, 480);  
  opencv.loadCascade(OpenCV.CASCADE_EYE);  
  video.start();  
}
```

```
void draw() {  
  image(video, 0, 0 );  
  opencv.loadImage(video);  
  Rectangle[] faces = opencv.detect();  
  noFill(); stroke(0, 255, 0); strokeWeight(3);  
  for (int i = 0; i < faces.length; i++) {  
    rect(faces[i].x, faces[i].y, faces[i].width, faces[i].height);  
  }  
}
```

```
void captureEvent(Capture c) {
```



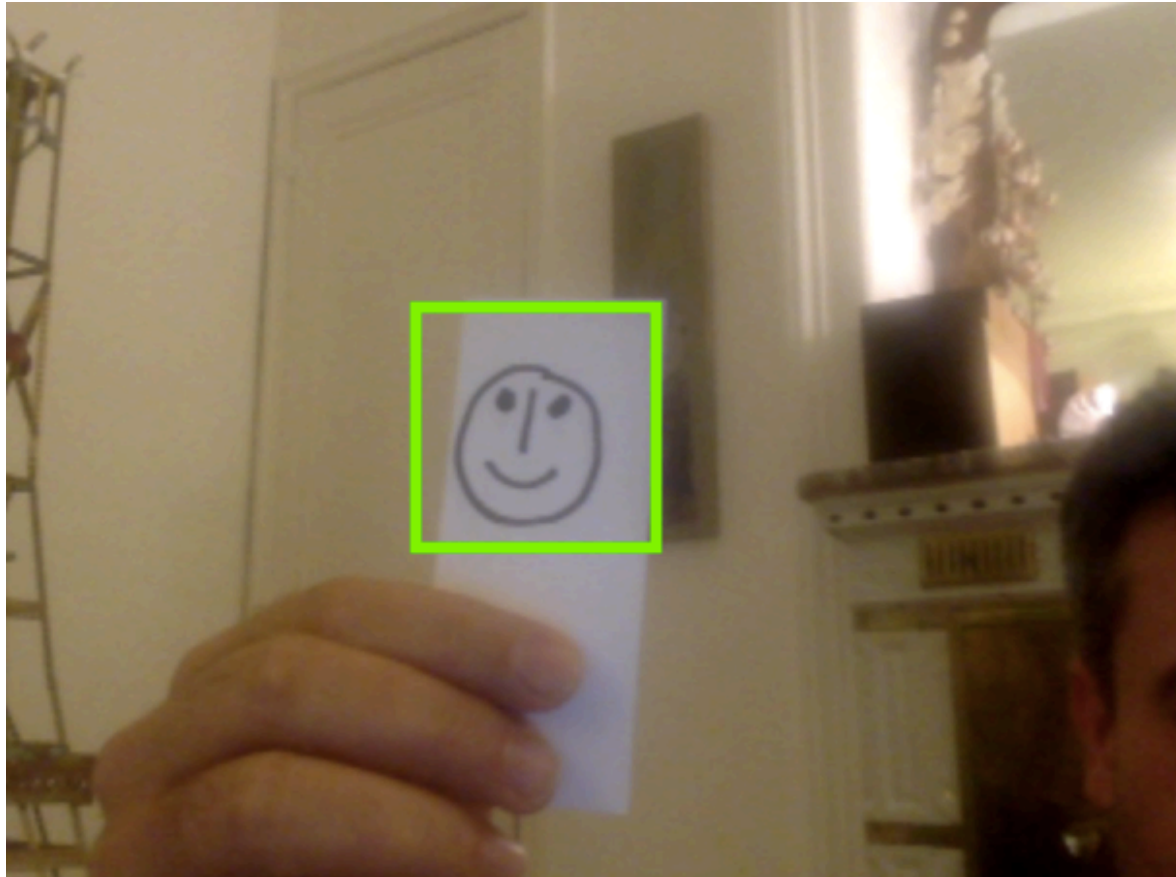
**Seule chose  
qui change :-)**

## Autres cascades à tester ...

Nom	Date de modification	Taille	Type
haarcascade_clock.xml	20 janv. 2014 23:50	104 Ko	Texte XML
haarcascade_eye_tree_eyeglasses.xml	20 janv. 2014 23:50	1,1 Mo	Texte XML
haarcascade_eye.xml	20 janv. 2014 23:50	506 Ko	Texte XML
haarcascade_frontalface_alt_tree.xml	20 janv. 2014 23:50	3,6 Mo	Texte XML
haarcascade_frontalface_alt.xml	20 janv. 2014 23:50	920 Ko	Texte XML
haarcascade_frontalface_alt2.xml	20 janv. 2014 23:50	837 Ko	Texte XML
haarcascade_frontalface_default.xml	20 janv. 2014 23:50	1,3 Mo	Texte XML
haarcascade_fullbody.xml	20 janv. 2014 23:50	637 Ko	Texte XML
haarcascade_lefteye_2splits.xml	20 janv. 2014 23:50	323 Ko	Texte XML
haarcascade_lowerbody.xml	20 janv. 2014 23:50	531 Ko	Texte XML
haarcascade_mcs_eyepair_big.xml	20 janv. 2014 23:50	358 Ko	Texte XML
haarcascade_mcs_eyepair_small.xml	20 janv. 2014 23:50	410 Ko	Texte XML
haarcascade_mcs_lefttear.xml	20 janv. 2014 23:50	313 Ko	Texte XML
haarcascade_mcs_lefteye.xml	20 janv. 2014 23:50	778 Ko	Texte XML
haarcascade_mcs_mouth.xml	20 janv. 2014 23:50	720 Ko	Texte XML
haarcascade_mcs_nose.xml	20 janv. 2014 23:50	1,6 Mo	Texte XML
haarcascade_mcs_righttear.xml	20 janv. 2014 23:50	325 Ko	Texte XML
haarcascade_mcs_righteye.xml	20 janv. 2014 23:50	1,4 Mo	Texte XML
haarcascade_mcs_upperbody.xml	20 janv. 2014 23:50	1,5 Mo	Texte XML
haarcascade_profileface.xml	20 janv. 2014 23:50	1,1 Mo	Texte XML
haarcascade_righteye_2splits.xml	20 janv. 2014 23:50	325 Ko	Texte XML
haarcascade_upperbody.xml	20 janv. 2014 23:50	1 Mo	Texte XML
hogcascade_pedestrians.xml	20 janv. 2014 23:50	130 Ko	Texte XML
lbpcascade_frontalface.xml	20 janv. 2014 23:50	52 Ko	Texte XML

Macintosh HD > Utilisat... > pcubau > Docum... > Proces... > librairie... > opencv > library > cascade-files

## Reprise de mon jeu incroyable



**C'est trop fatiguant avec sa vraie tête**

