Noelia Vállez Enano

José Luis Espinosa Aranda

Taller-Reto: Visión por Computador

1ª Semana de la visión por computador

Del 6 al 11 de noviembre de 2017

visioncr

¿Qué es la visión por computador?

La mayor parte de la información nos llega a través de nuestros ojos...

¿Qué es la visión por computador?

Modelado 3D

Reconocimiento de objetos

Seguimiento de objetos

Control de calidad de la producción

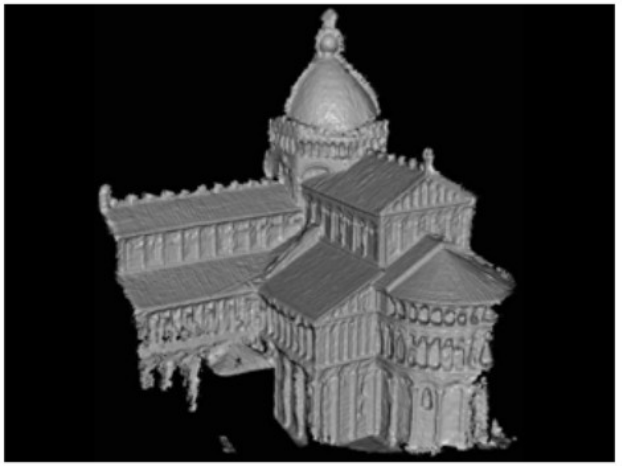
Imagen médica

Seguridad durante la conducción

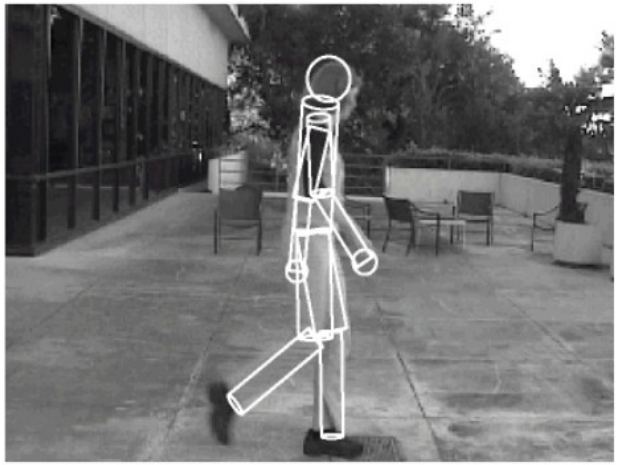
OCR...

¿Qué es la visión por computador?

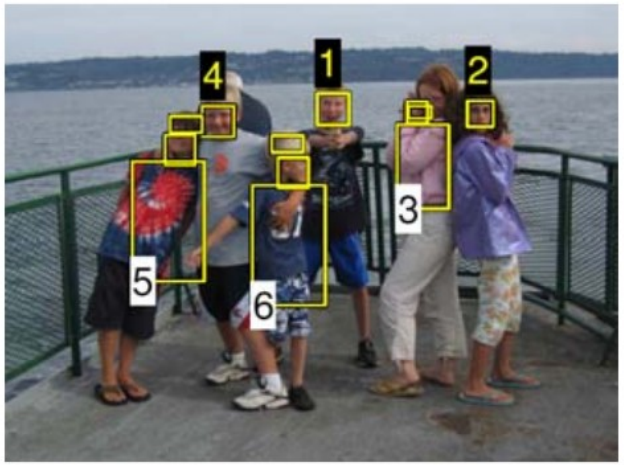
(a)



(b)



(c)

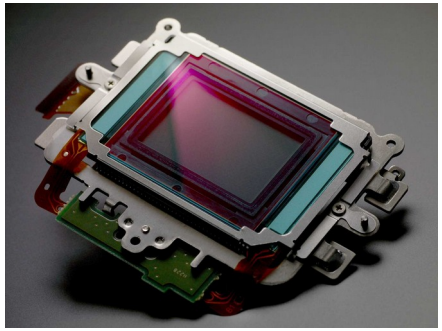


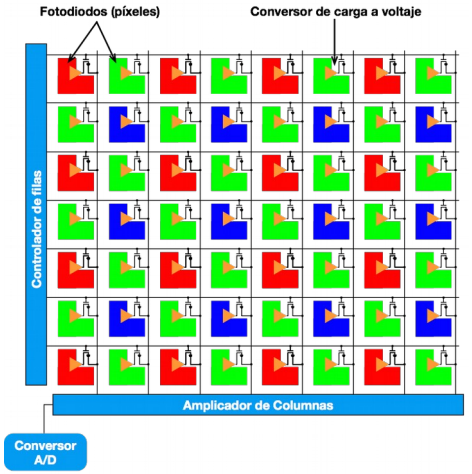
(d)

¿Qué es la visión por computador?

Engloba diferentes disciplinas:

- Física
- Química
- Matemáticas
- Electrónica
- Informática





¿Qué es la visión por computador?

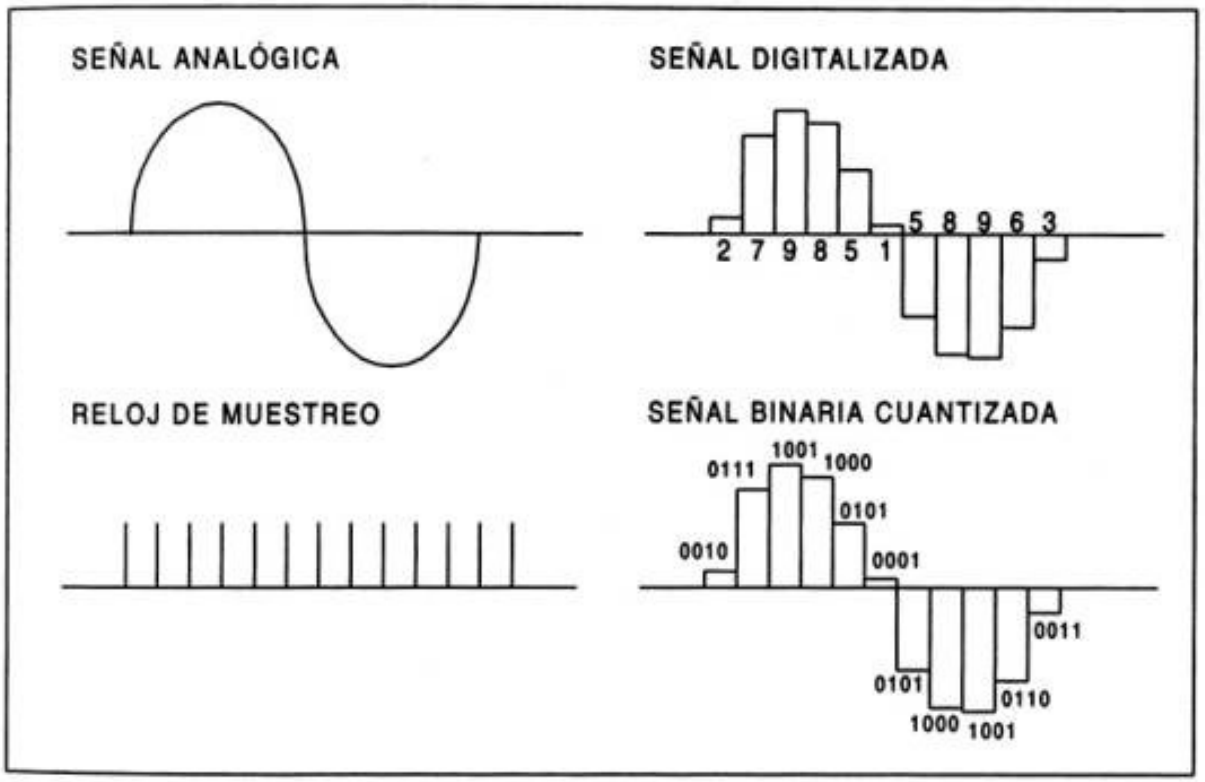
Etapas:

Procesamiento de Imágenes

Digitalización de la imagen

Procesamiento de Imágenes

Digitalización:



Procesamiento de Imágenes

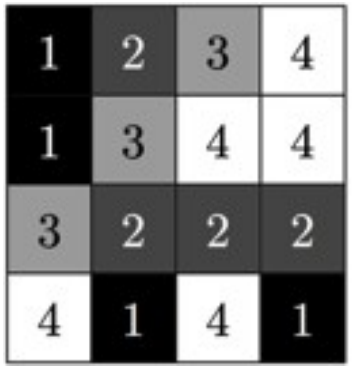
Digitalización:

Matriz de la imagen:

$$f = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{pmatrix}$$

Niveles de gris:

$$L = \{0, 1, 2, ..., L-1\}$$



Level	*Size zone, s*		
g	1	2	3
1	2	1	0
2	1	0	1
3	0	0	1
4	2	0	1

Procesamiento de Imágenes

Tipos de imagen

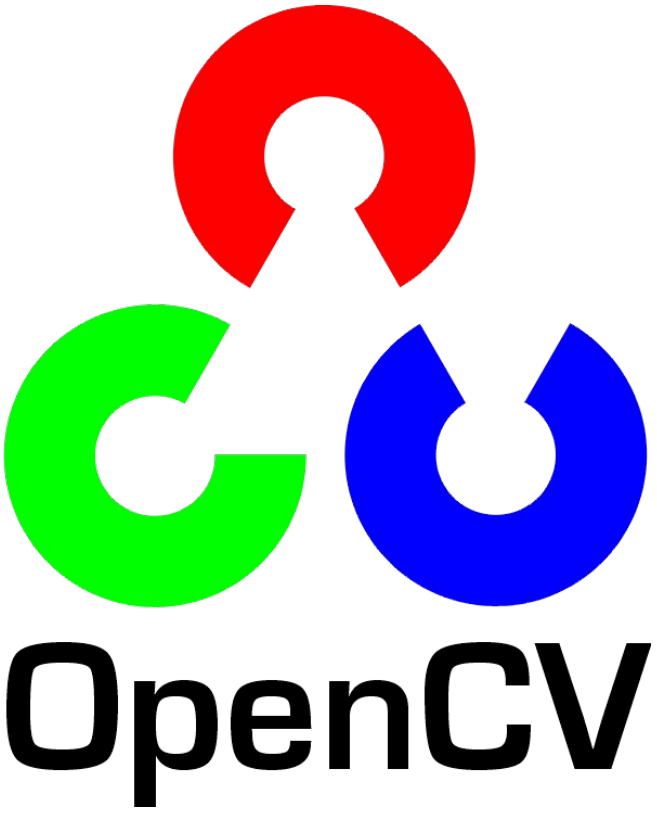
Pixel = ([0...255][0...255][0...255])

Pixel = [0...255]

Pixel = [0...1]

Librerías

Processing (JAVA)



Primeros pasos con Processing

- **Es un lenguaje de programación sencillo**
- **El prototipado se realiza mediante un *sketch***
- **Con pocas líneas de código se consiguen programas muy potentes**
- **Permite la visualización de contenidos gráficos**
- **Sencillo de instalar: http://processing.org**

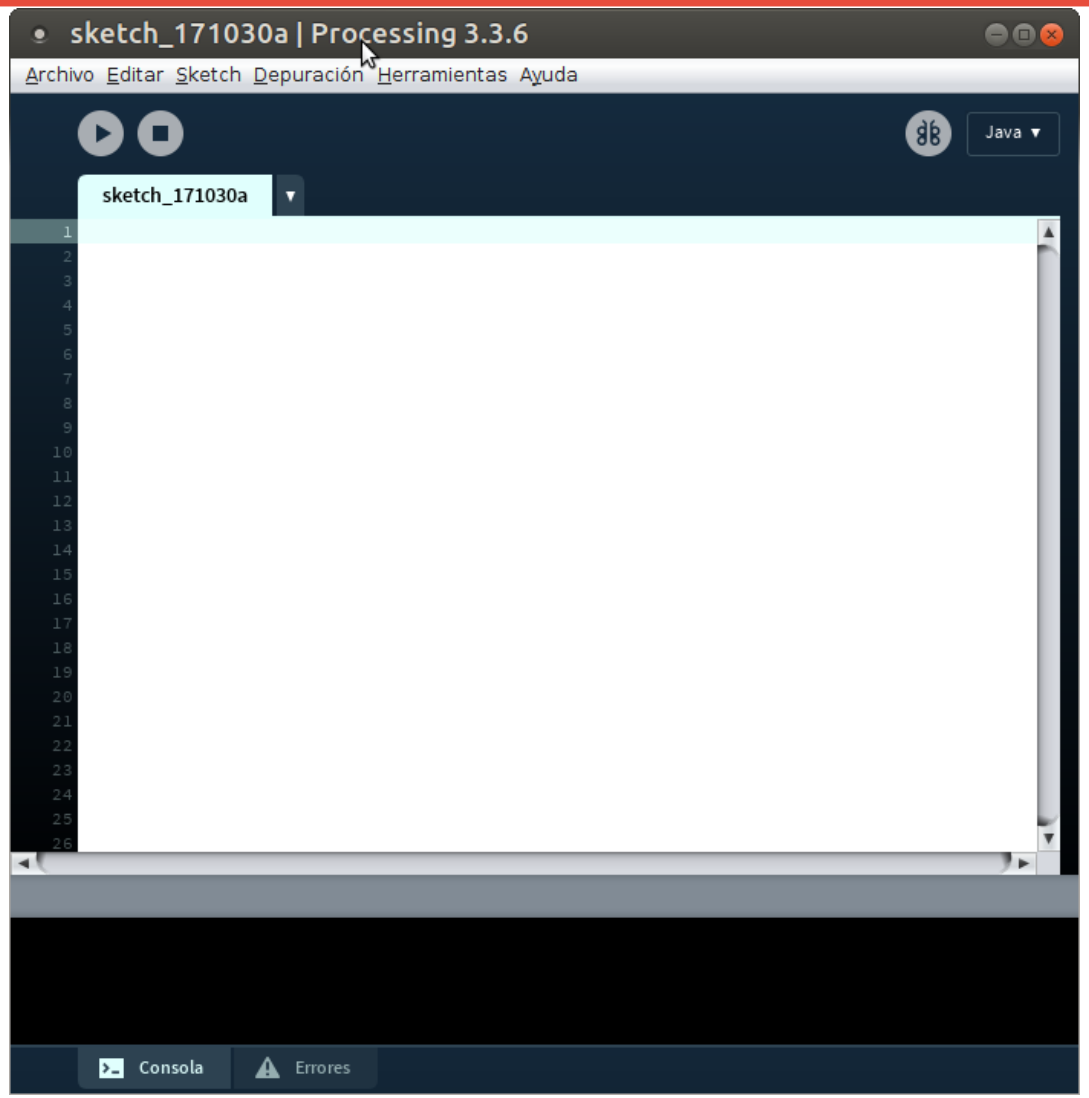
  Descargar la versión correspondiente a nuestro sistema operativo y descomprimir

Primeros pasos con Processing

- **Instalación de las librerías**

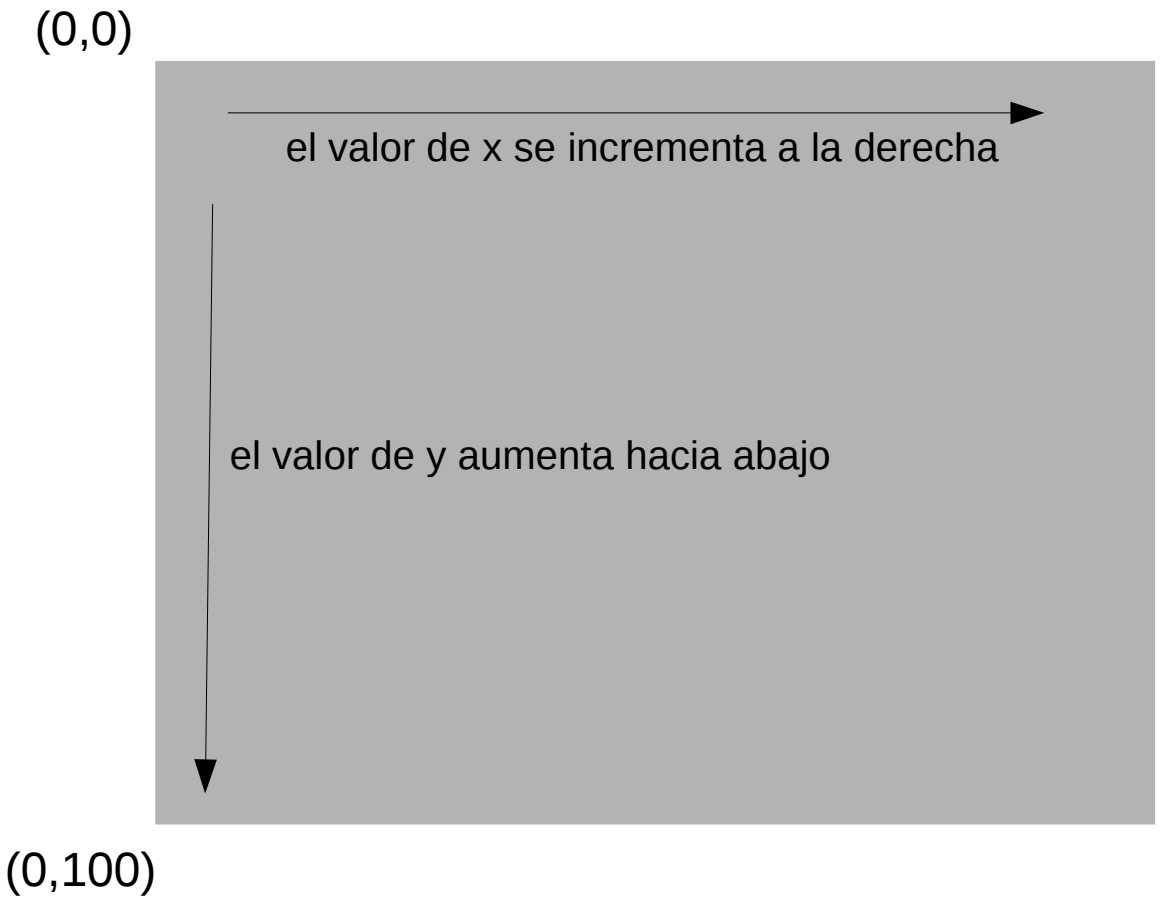
1) Ejecutar el entorno de desarrollo
2) Hacer clic en Sketch > Import library… > add library…
3) Buscar la librería “Video”, cuyo autor es “The Processing Foundation”, hacer clic para elegirla e “Install” para instalarla
4) Buscar la librería “OpenCV for Processing”, cuyo autor es “Greg Borenstein”, hacer clic para elegirla e “Install” para instalarla

Primeros pasos con Processing



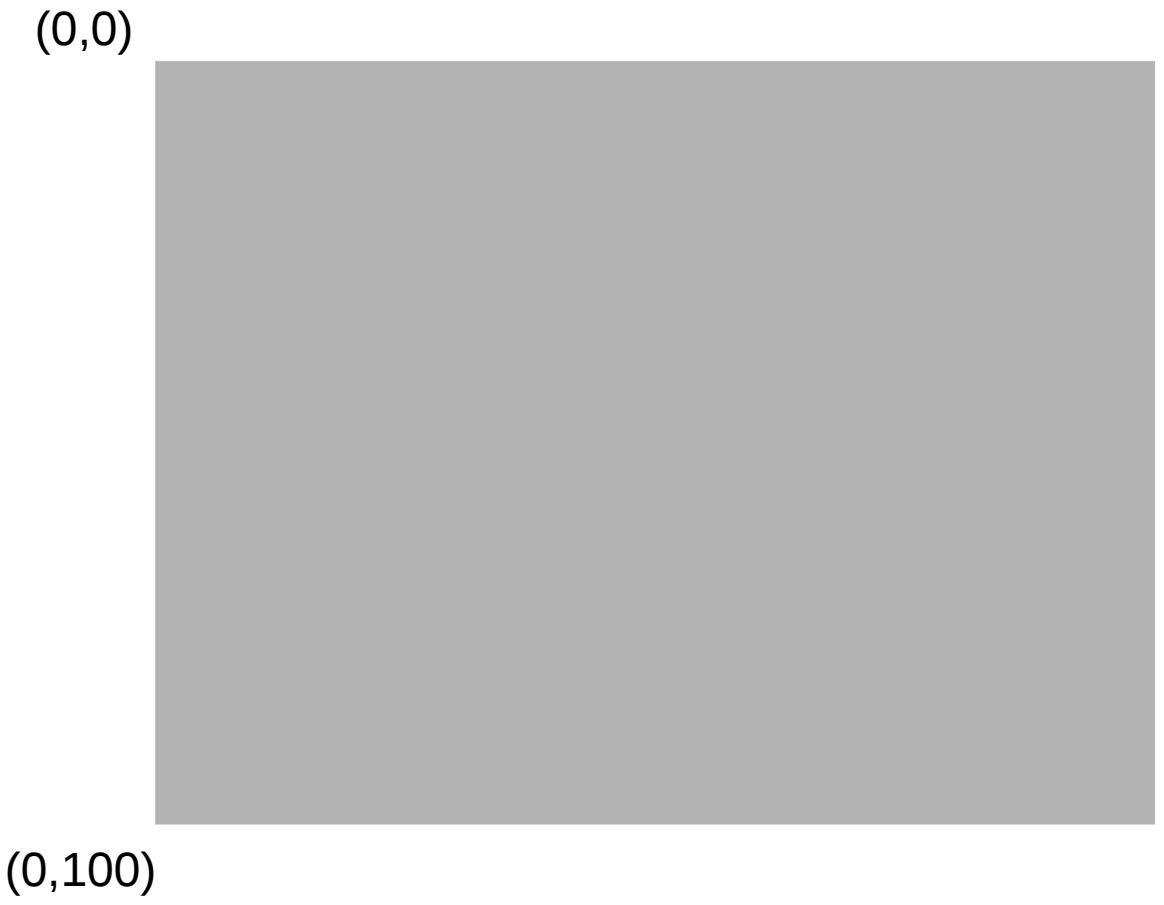
Primeros pasos con Processing

Pensad en una gráfica (x,y)



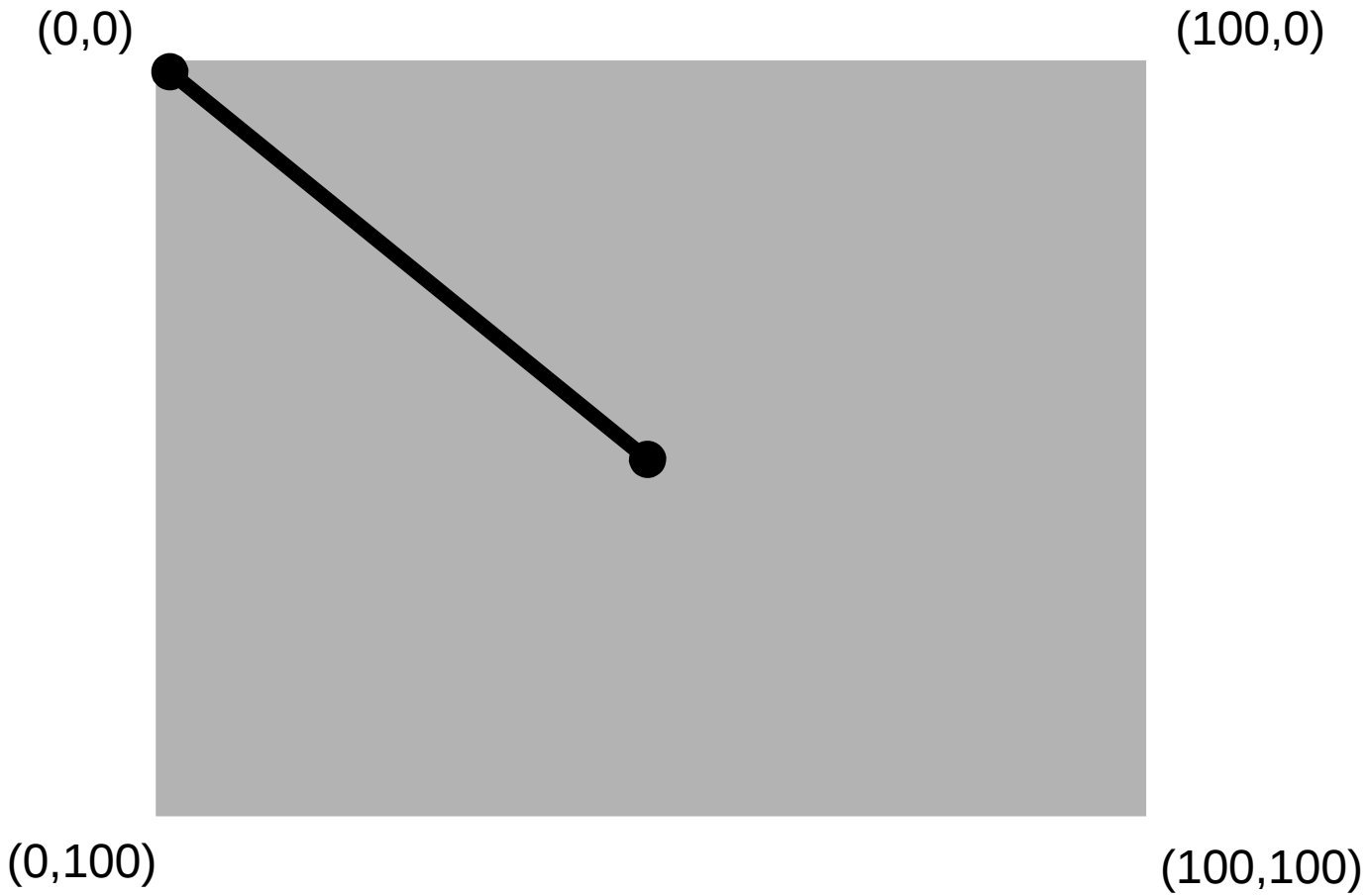
En la pantalla del ordenador el origen (0,0) se encuentra siempre en la esquina superior izquierda

Primeros pasos con Processing



¿Como dibujaríais una línea desde el punto (0,0) al (50,50)?

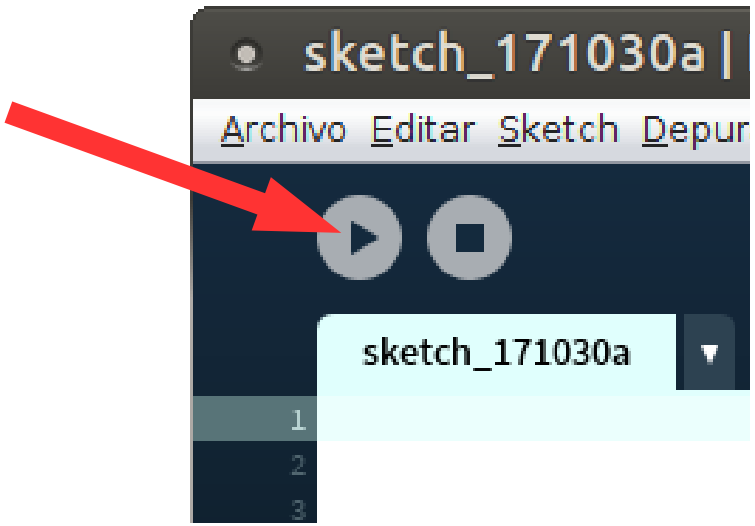
Primeros pasos con Processing

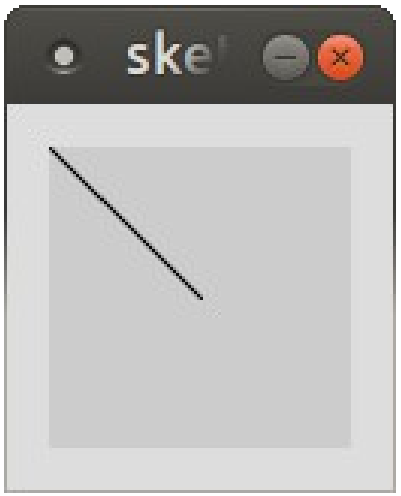


Primeros pasos con Processing

Primer programa, ¡Hola Mundo!

```
size(100, 100); //configura el tamaño de la ventana
line(0, 0, 50, 50); //dibuja una línea desde (0,0) a (50,50)
```

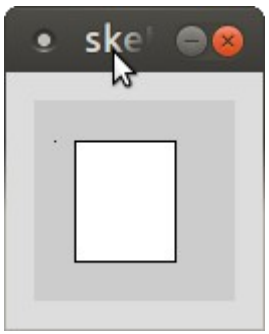


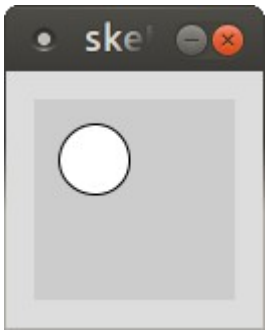


sketch01.pde

Otras formas básicas

```
point(10, 20); //punto en (10,20)
rect(20,20, 50,60); //rectángulo desde (20,20) a (50,60)
triangle(15,23, 28,20, 26,75); //triangulo con los 3 puntos definidos
quad(38,31, 86,20, 69,63, 30,76); //cuadrángulo definido por 4 puntos
ellipse(30,30,35,35); //elipse indicando centro de elipse, radio x e y
```





sketch02.pde

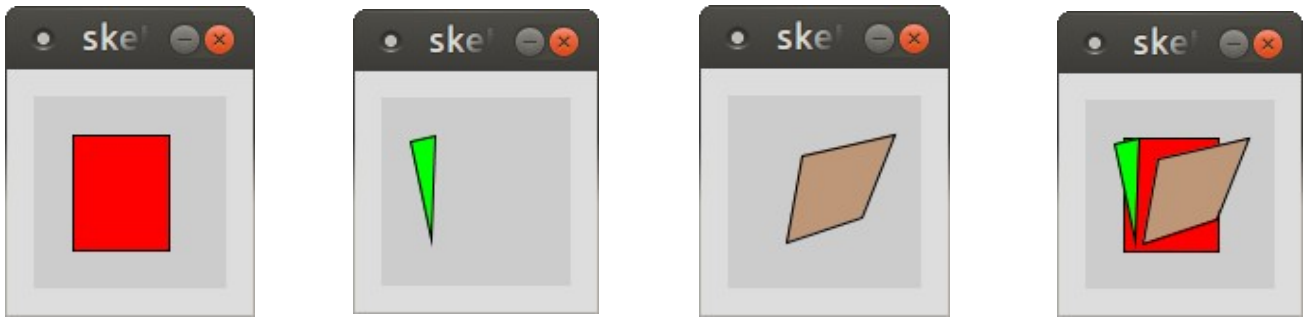
Representación de colores

- Los colores se representan como combinación de ROJO, VERDE y AZUL (RGB)
- 0 representa que no existe contribución de un color
- 255 representa que existe la máxima contribución de un color
- Rojo puro: (255,0,0)
- Verde puro: (0,255,0)
- Azul puro: (0,0,255)

Primeros pasos con Processing

Rellenar formas con colores

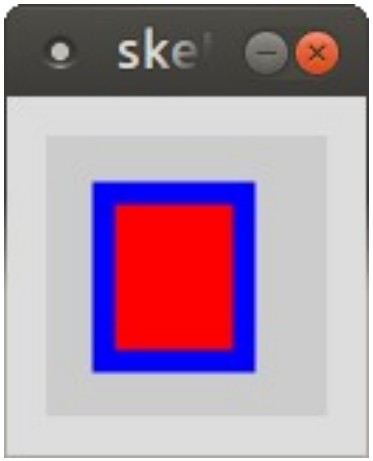
```
fill(255,0,0); //rellenar con rojo
rect(20,20, 50,60); //rectángulo desde (20,20) a (50,60)

fill(0,255,0); //rellenar con verde
triangle(15,23, 28,20, 26,75); //triangulo con los 3 puntos definidos

fill(190,150,120); //rellenar con marrón
quad(38,31, 86,20, 69,63, 30,76); //cuadrángulo definido por 4 puntos
```



sketch03.pde

Color y tamaño de línea

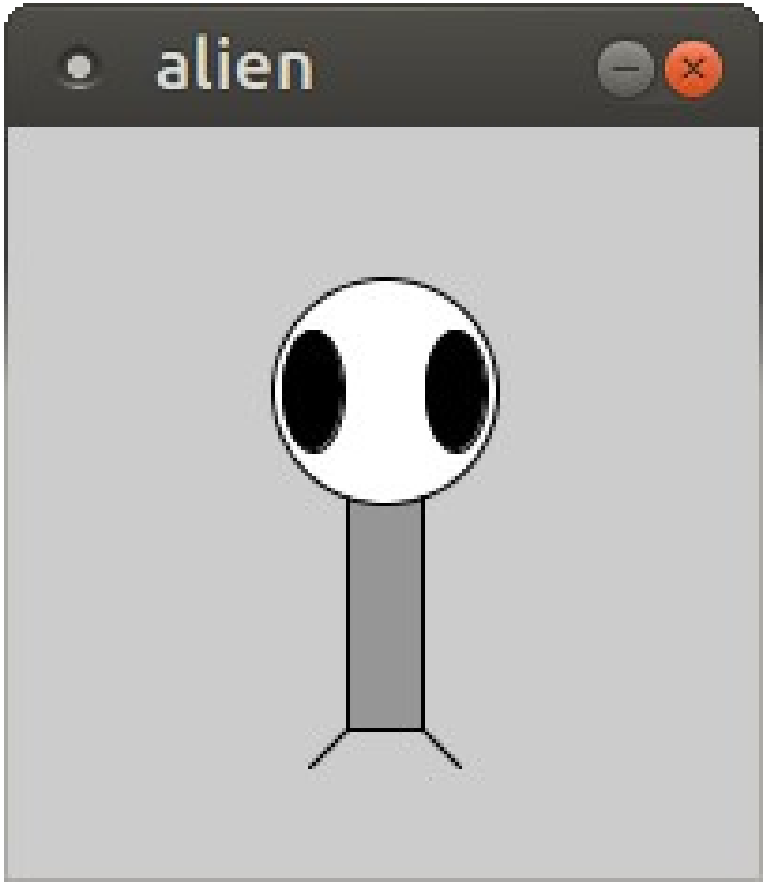
```
fill(255,0,0); //rellenar con rojo
stroke(0,0,255); //borde de color azul
strokeWeight(8); //tamaño de borde 8
rect(20,20, 50,60); //rectángulo desde (20,20) a (50,60)
```



sketch04.pde

Primeros pasos con Processing

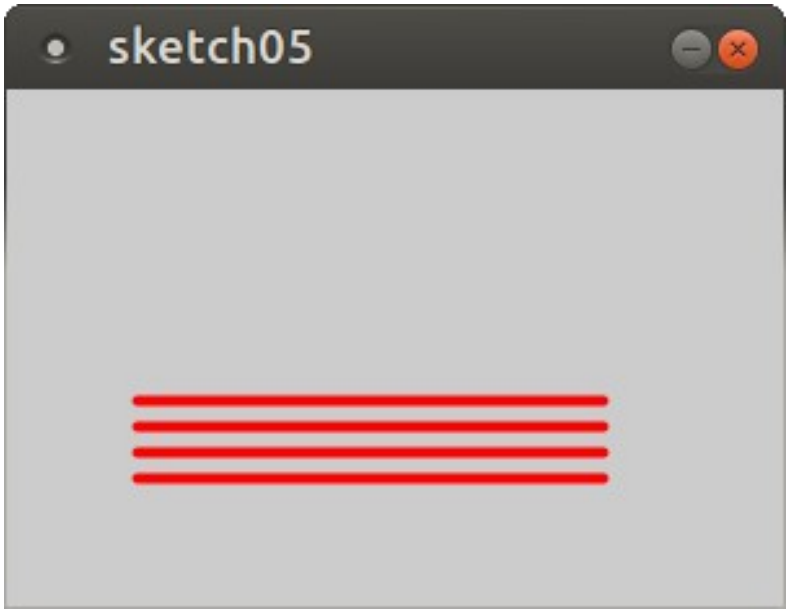
```
size(200, 200);
// cuerpo
fill(150, 150, 150);
rect(90,60,20,100);
// cabeza
fill(255, 255, 255);
ellipse(100,70,60,60);
// ojos
fill(0, 0, 0);
ellipse(81,70,16,32);
ellipse(119,70,16,32);
// piernas
stroke(0, 0, 0);
line(90,160,80,170);
line(110,160,120,170);
```



alien.pde

Uso de variables (igual que en Java)

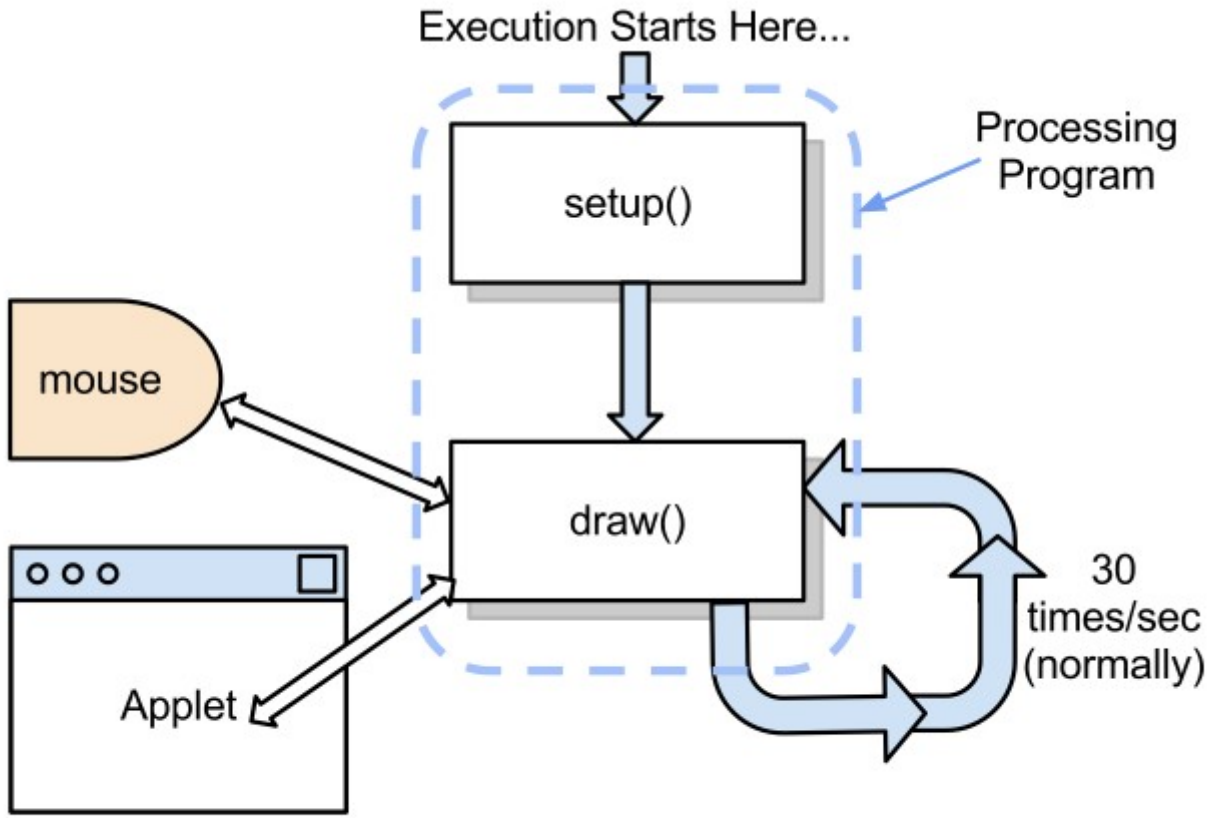
```
size(300,200);
stroke(255, 0, 0); // linea de color rojo
strokeWeight(4); // ancho de línea 4
int a = 50;
int b = 120;
int c = 180;
int distancia = 10;
line(a,b, a+c, b);
line(a,b+distancia, a+c, b+distancia);
line(a,b+2*distancia, a+c, b+2*distancia);
line(a,b+3*distancia, a+c, b+3*distancia);
```



sketch05.pde

Como programar en Processing

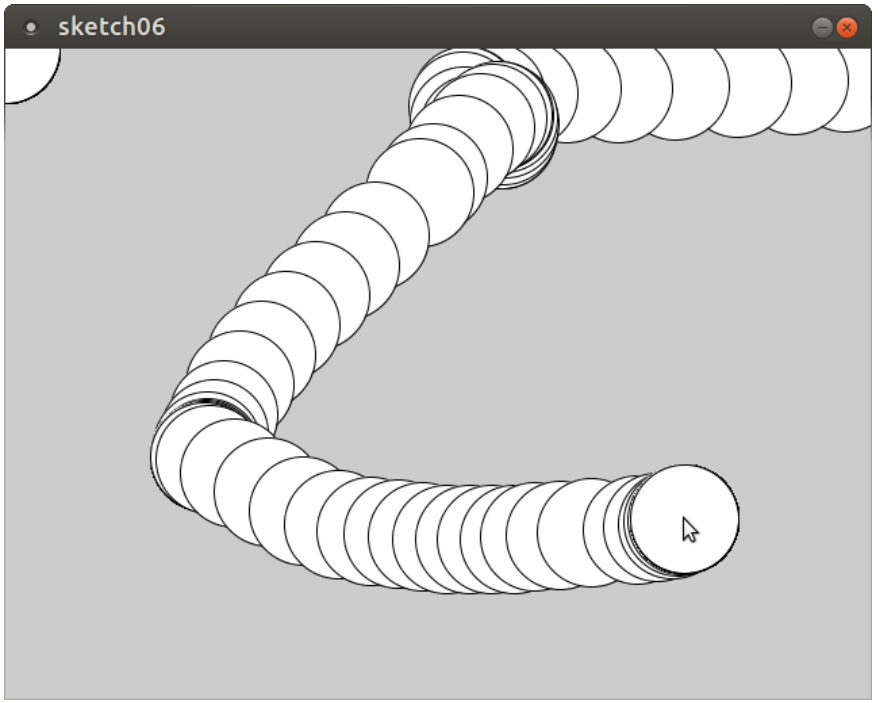
- **setup()** solo se ejecuta una vez al inicio del programa
- **draw()** se ejecuta repetidamente, en bucle
  se llama unas 30 veces por segundo, se suele usar para simular animaciones o tratar vídeo



Como programar en Processing

Ejemplo básico

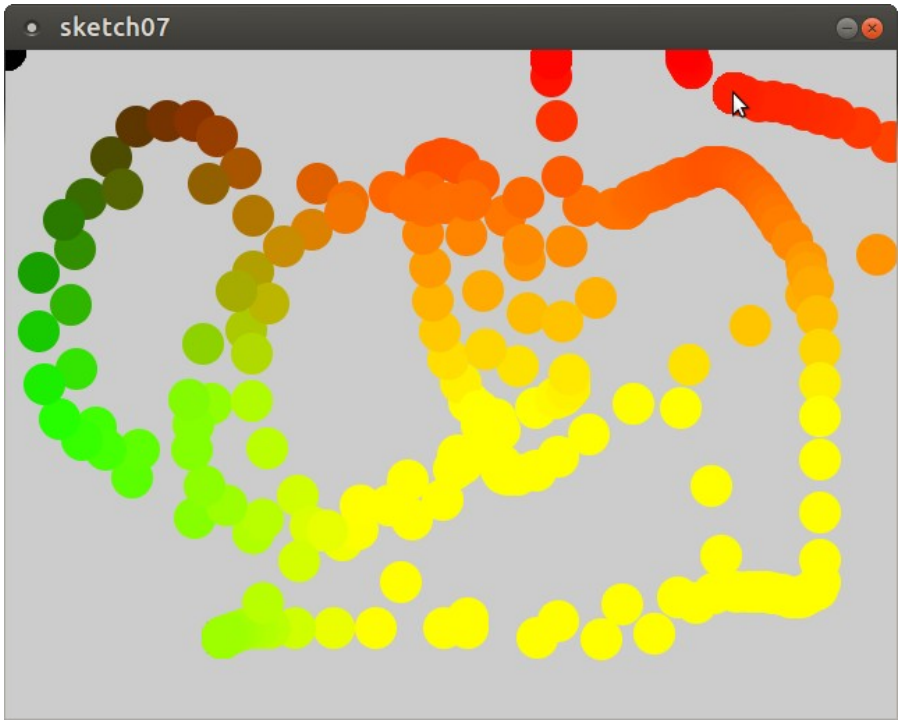
```
void setup() {
  size(640, 480);
}

void draw() {
  ellipse(mouseX, mouseY, 80, 80);
}
```



sketch06.pde

Ejemplo básico 2

```
void setup() {
  size(640, 480);
  smooth();
  noStroke();
}

void draw() {
  fill(mouseX, mouseY, 0);
  ellipse(mouseX, mouseY, 30, 30);
}
```



sketch07.pde

Webcam en Processing (parte 1)

```
import processing.video.*; //cargamos la librería de video

Capture video; //objeto para capturar video

void setup() {
  size(640, 480); //ventana
  noStroke();

  video = new Capture(this, 640, 480); //se crea el objeto para el video
  video.start(); //activamos el video
}
```

sketch08.pde

Webcam en Processing (parte 2)

```
void draw() {

  if (video.available() == true) {
    video.read();
  }

  image(video,0,0); //coloca la imagen

}
```

sketch08.pde

Leer un fichero de imagen (OpenCV)


```
import gab.opencv.*; //cargamos OpenCV

OpenCV opencv; //objeto para la imagen

void setup() {
  opencv = new OpenCV(this, "test.jpg"); //cargamos la imagen
  size(380, 380);

  image(opencv.getInput(), 0, 0); //mostramos la imagen
}
```

sketch09.pde

Procesamiento de imagen

Filtros básicos de imagen (OpenCV)


```
import gab.opencv.*; //cargamos OpenCV

OpenCV opencv; //objeto para la imagen

void setup() {
  opencv = new OpenCV(this, "test.jpg"); //cargamos la imagen
  opencv.useColor(); //trabajar con la imagen en color
  size(600, 300);

  image(opencv.getInput(), 0, 0, 300, 300); //mostramos la imagen
  opencv.blur(30); //emborronamos la imagen
  image(opencv.getOutput(),300,0,300,300);
}
```

sketch10.pde

Procesamiento de imagen

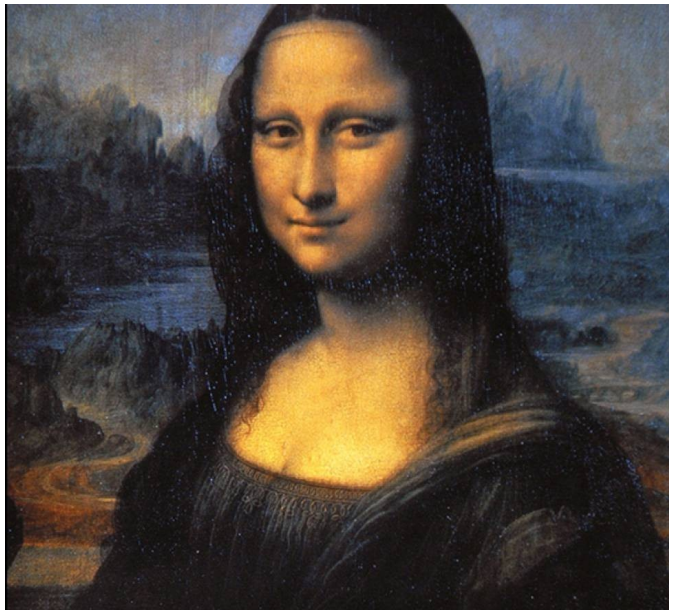
Otros filtros básicos de imagen

```
opencv.flip(OpenCV.VERTICAL); //rotar
opencv.invert(); //invertir colores
opencv.brightness(128); //brillo
opencv.brightness((int)map(mouseX,0,width,-255,255)); //brillo ratón
opencv.contrast(5); //contraste
opencv.contrast(map(mouseX,0,width,0,255)); //contraste raton
opencv.threshold(127); //umbralizado
opencv.inRange(122,132); //rango
```

sketch10.pde

Procesamiento de imagen

Umbralizado (binarización de una imagen)



Procesamiento de imagen

Detección de bordes (detectar formas)

Canny:

opencv.findCannyEdges(20,75);

Scharr:

opencv.findScharrEdges(OpenCV.HORIZONTAL);

Sobel:

opencv.findSobelEdges(1,0);

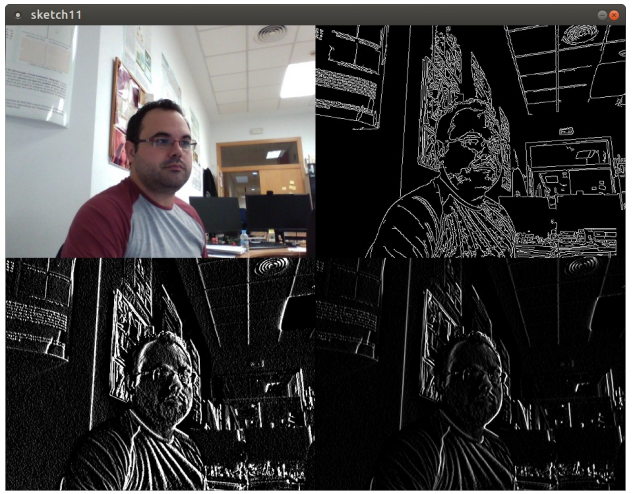
Detección de bordes (ejemplo webcam 1)

```
import gab.opencv.*;
import processing.video.*;

OpenCV opencv;
Capture video;

void setup() {
  size(960, 720);
  opencv = new OpenCV(this, 640, 480);
  video = new Capture(this, 640, 480);
  video.start();
}
```

sketch11.pde

Detección de bordes (ejemplo webcam 2)

```
void draw() {

  if (video.available() == true) {
    video.read();
  }
  image(video,0,0,480,360);

  opencv.loadImage(video);
  opencv.findCannyEdges(20,75);
  image(opencv.getOutput(),480,0,480,360);

  opencv.loadImage(video);
  opencv.findScharrEdges(OpenCV.HORIZONTAL);
  image(opencv.getOutput(),0,360,480,360);

  opencv.loadImage(video);
  opencv.findSobelEdges(1,0);
  image(opencv.getOutput(),
      480,360,480,360);

}
```



sketch11.pde

Procesamiento de imagen

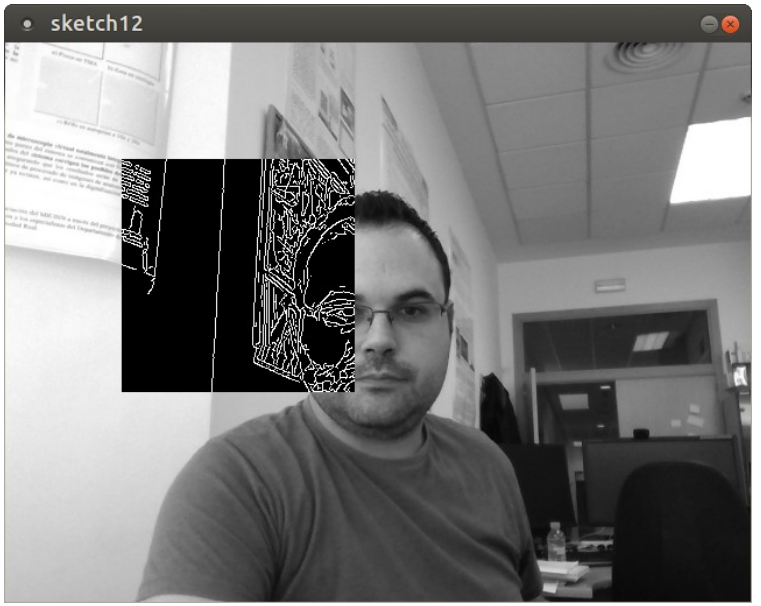
Regiones de interés 1 (editar una parte de la imagen)

```
import gab.opencv.*;
import processing.video.*;

OpenCV opencv;
Capture video;

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

sketch12.pde

Regiones de interés 2 (editar una parte de la imagen)

```
void draw() {

  if (video.available() == true) {
    video.read();
  }
  image(video,0,0,480,360);

  opencv.loadImage(video);
  opencv.setROI(100, 100, 200, 200);
  opencv.findCannyEdges(20,75);
  opencv.releaseROI();
  image(opencv.getOutput(),0,0);
}
```



sketch12.pde

Detección de Objetos

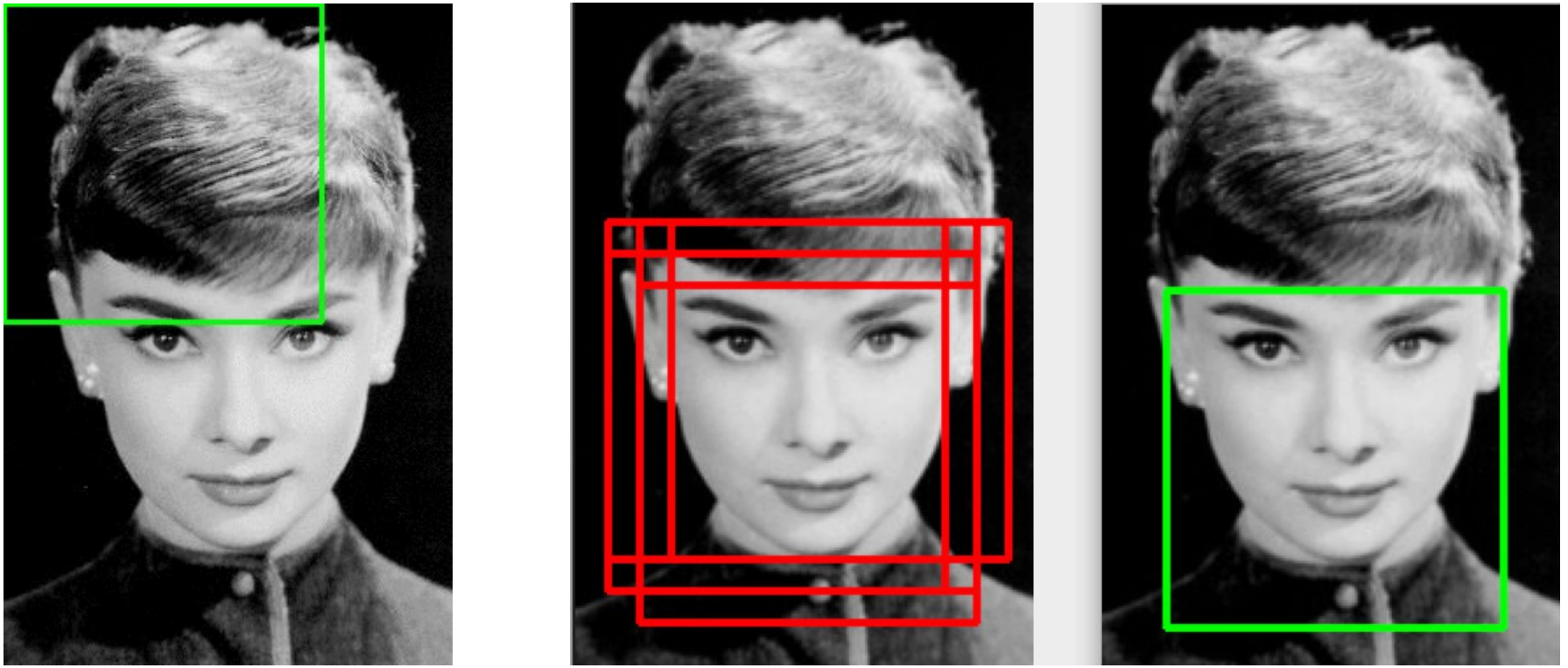
Detección de Caras:

1. **Extracción de Características**
2. **Entrenamiento del detector**
3. **Recorrer la nueva imagen en busca del objeto aplicando el detector**
4. **Obtener la región que resulte positiva**

Detección de Objetos

Detección de Caras

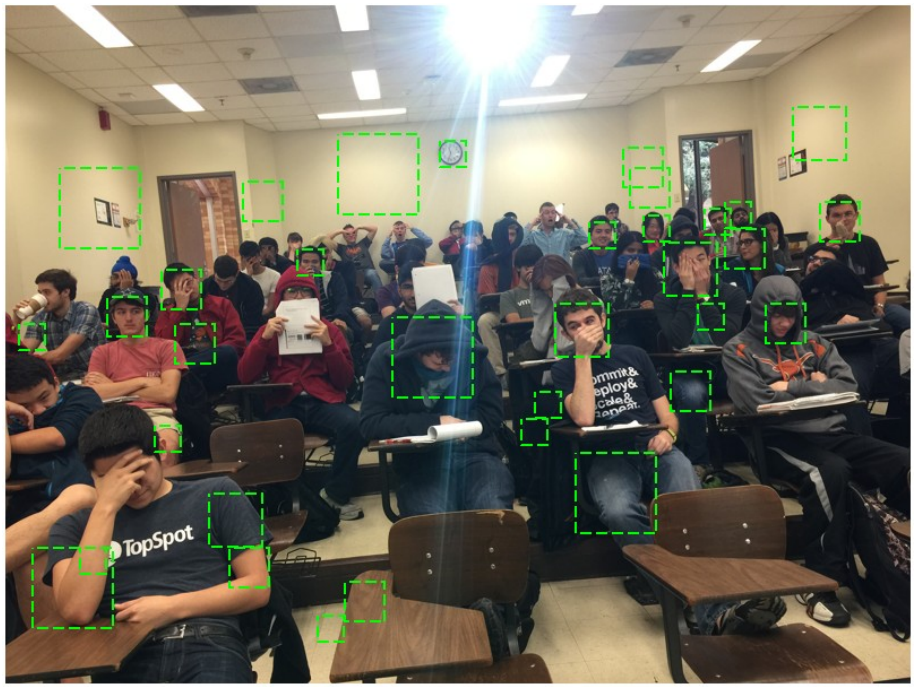
- Ventana Deslizante



Detección de Objetos

Detección de Caras

- Problemas



Detección de Objetos

Detección de Caras (1)

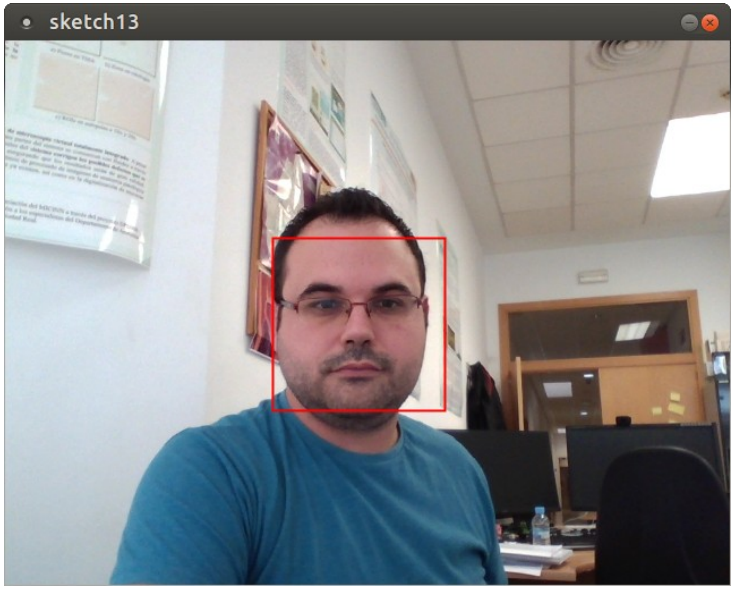
```
import gab.opencv.*;
import processing.video.*;
import java.awt.*;

OpenCV opencv;
Capture video;
Rectangle[] faces;
```

```
void setup() {

  size(640, 480);
  frameRate(30);
  noFill();
  strokeWeight(2);
  stroke(255, 0, 0);
  background(0);

  opencv = new OpenCV(this, 640, 480);
  opencv.loadCascade(OpenCV.CASCADE_FRONTALFACE);

  video = new Capture(this, 640, 480);
  video.start();
}
```

sketch13.pde

Detección de Caras (2)

```
void draw() {

  if (video.available() == true) {
    video.read();
  }

  image (video, 0, 0);

  opencv.loadImage(video);
  opencv.useColor();
  faces = opencv.detect();

  for (int i = 0; i < faces.length; i++) {
    rect(faces[i].x, faces[i].y, faces[i].width, faces[i].height);
  }
}
```

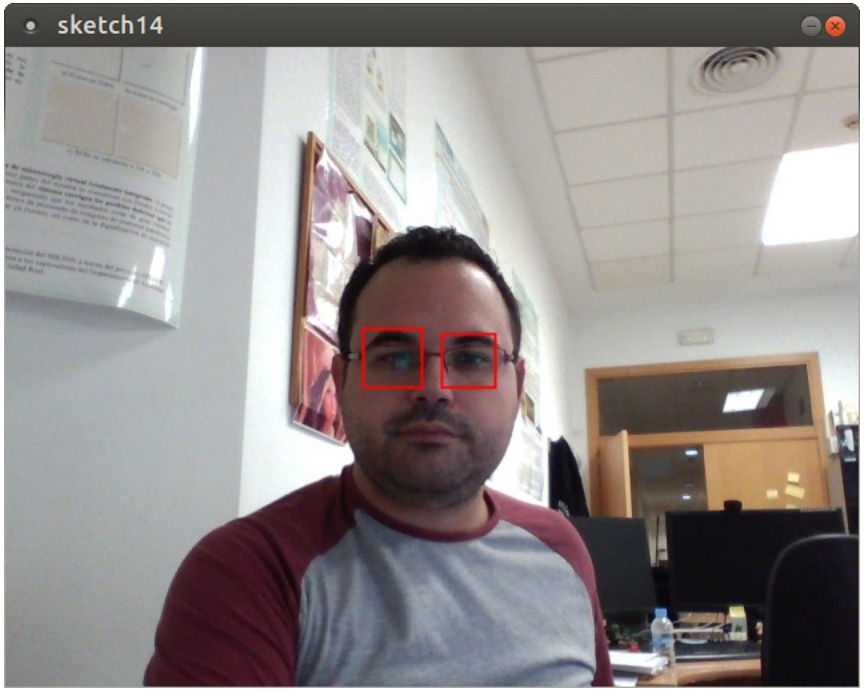


sketch13.pde

Detección de Objetos

Detección de Ojos

```
float DETECT_SCALE = 1.2;
int DETECT_MINNEIGHBOURS = 7;
int DETECT_MINSIZE = 0;
int DETECT_MAXSIZE = 0;

.....

opencv.loadCascade(OpenCV.CASCADE_EYE);

.....

eyes = opencv.detect(DETECT_SCALE, DETECT_MINNEIGHBOURS, 0, DETECT_MINSIZE,
DETECT_MAXSIZE);
```



sketch14.pde

Detección de Objetos

Otros detectores

CASCADE_FULLBODY (cuerpo completo)

CASCADE_LOWERBODY (parte inferior del cuerpo)

CASCADE_MOUTH (boca)

CASCADE_NOSE (nariz)

CASCADE_PEDESTRIAN (peatón)

CASCADE_PEDESTRIANS (peatones)

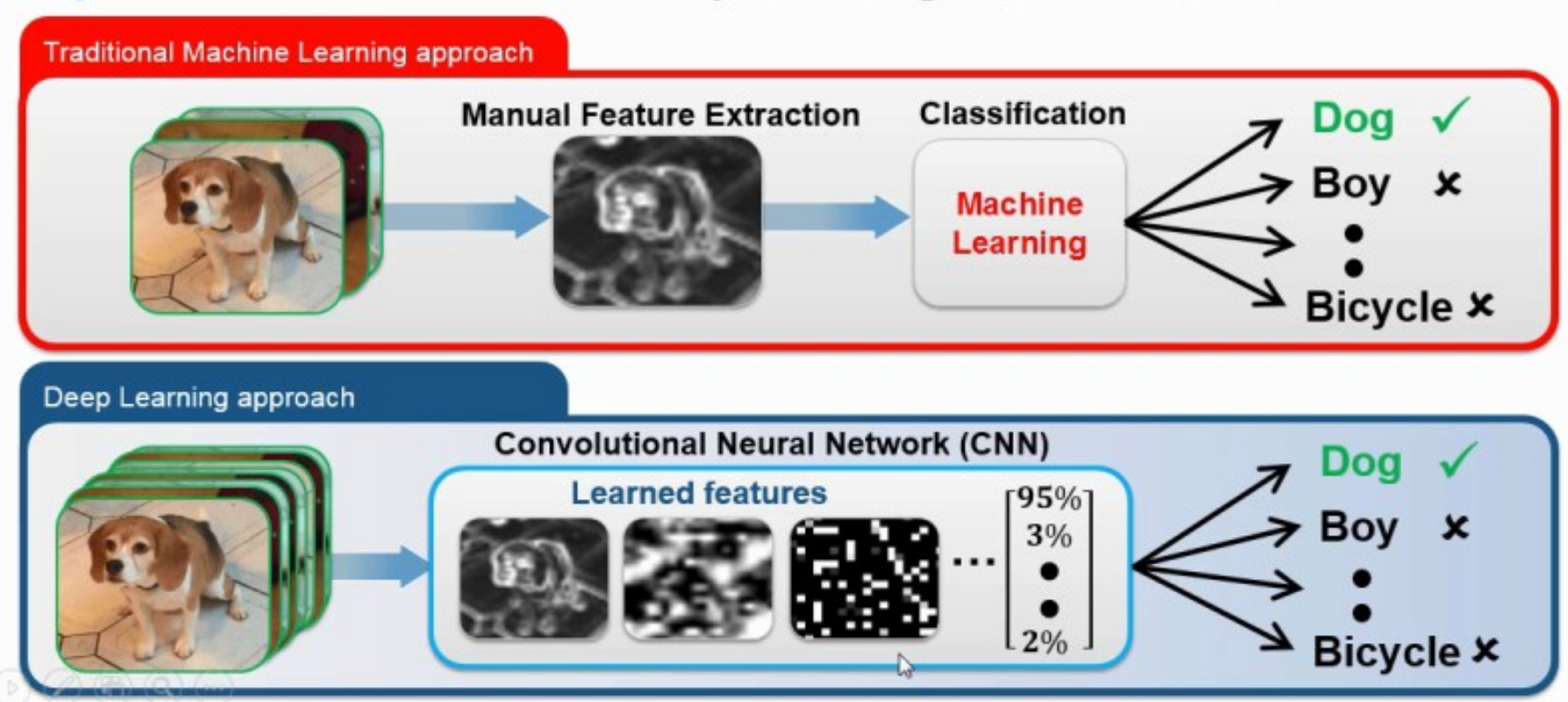
CASCADE_PROFILEFACE (cara de perfil)

CASCADE_RIGHT_EAR (oreja derecha)

CASCADE_UPPERBODY (parte superior del cuerpo)

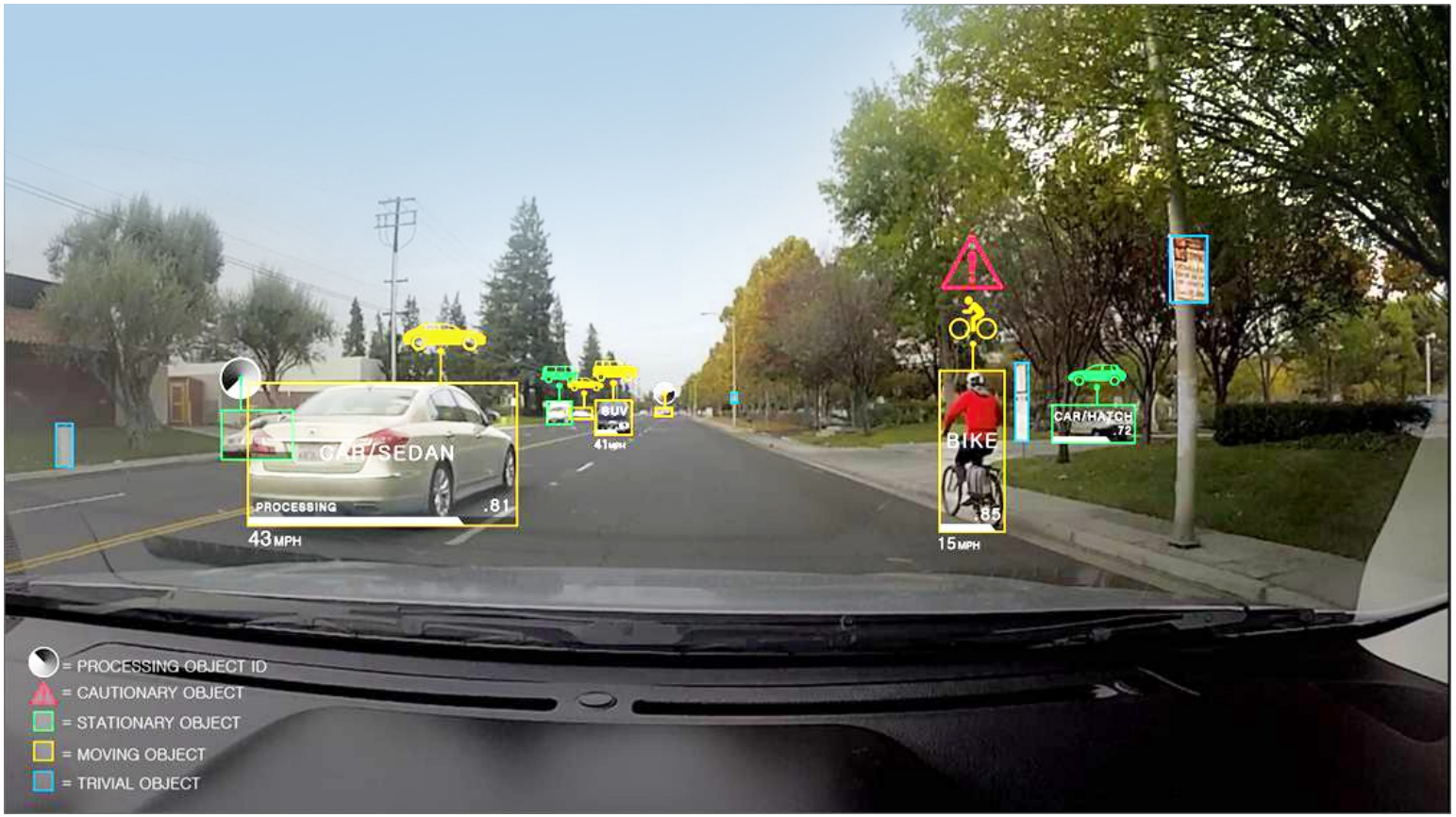
El Futuro… ¡Es ahora!

Deep Learning



El Futuro… ¡Es ahora!

Deep Learning



10 minutos de descanso y…

¡Comenzamos con el Reto!