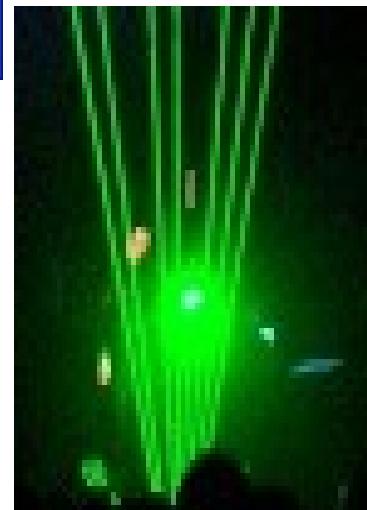
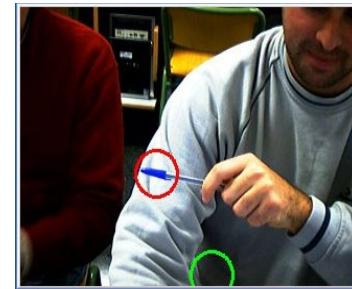
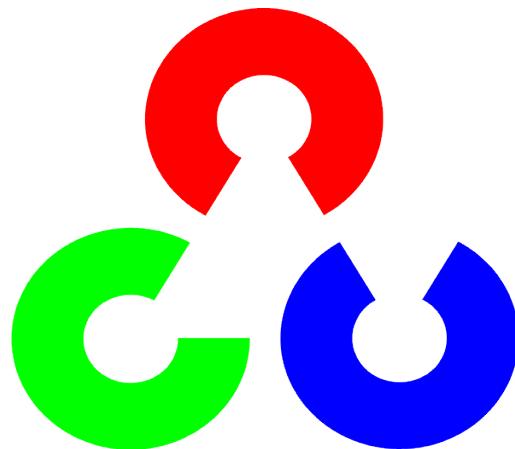
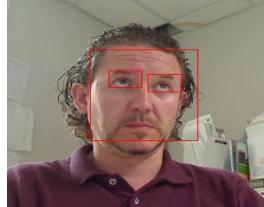
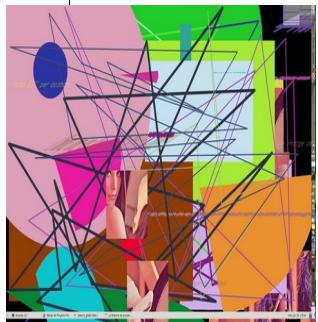


Seminario

Interacción mediante el uso de técnicas de Visión por Computador con OpenCV



Sistemas Multimedia Interactivos e Inmersivos

Grado de Ingeniero en Informática. Escola Tècnica Superior de Enginyeria Informàtica. Curso 2021/2022

Manuel Agustí

Objetivos

- Conocer la representación de una imagen en un sistema basado en computador.
- Formalizar el desarrollo del tratamiento de imágenes para sistemas de características multimedia.
- Conocer y experimentar las operaciones más elementales.
- Conocer la disponibilidad de librerías al efecto y proponer que se experimente con OpenCV.

Índice

1. Introducción

1. Introducción a la VxC

2. Ejemplos de aplicaciones interactivas basadas en VxC

3. Operaciones básicas de análisis de imagen

2. Uso de OpenCV: operaciones básicas

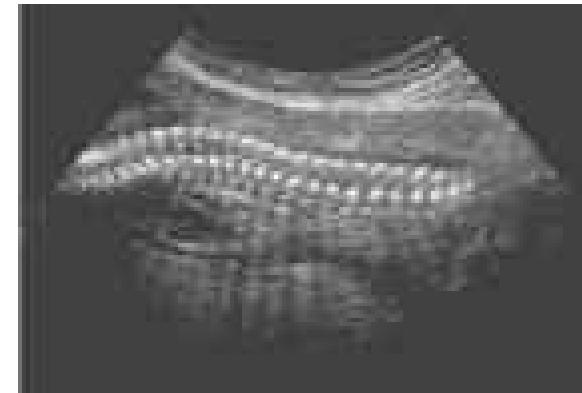
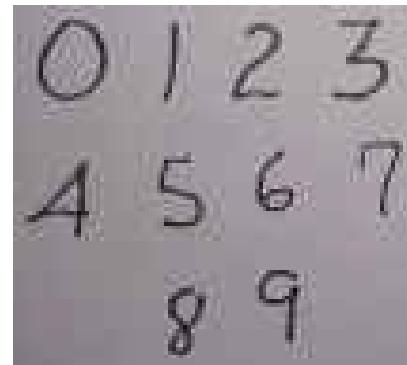
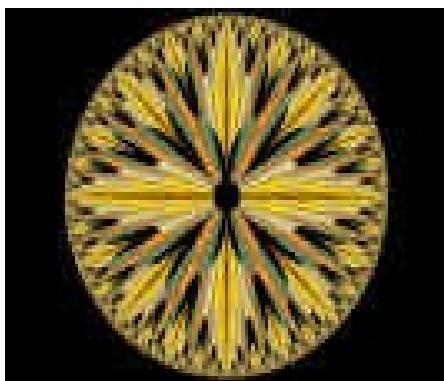
3. Aplicaciones

4. Bibliografía

5. Para acabar el tema de VxC ...

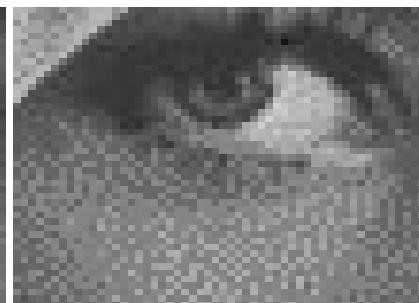
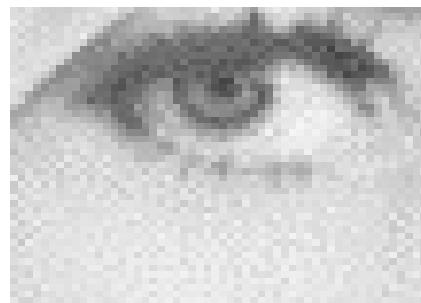
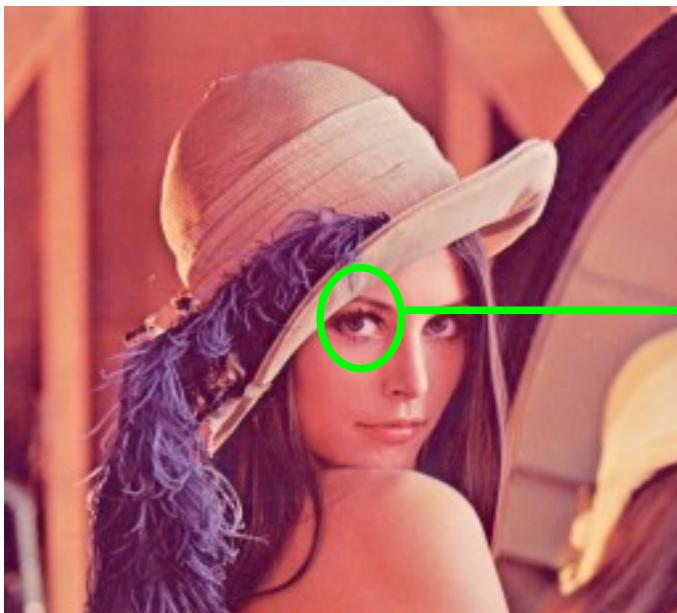
Introducción

- Revisión del medio 'imagen'
 - Las **imágenes digitales** son matrices de números, reales o complejos, representados por un número finito de bits.



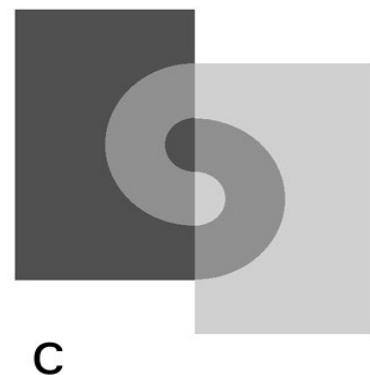
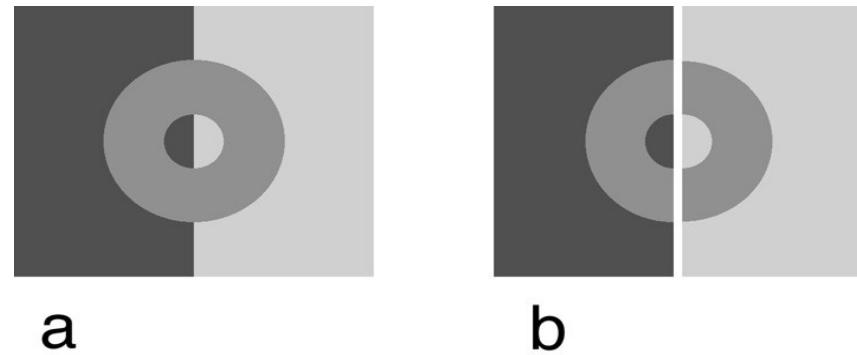
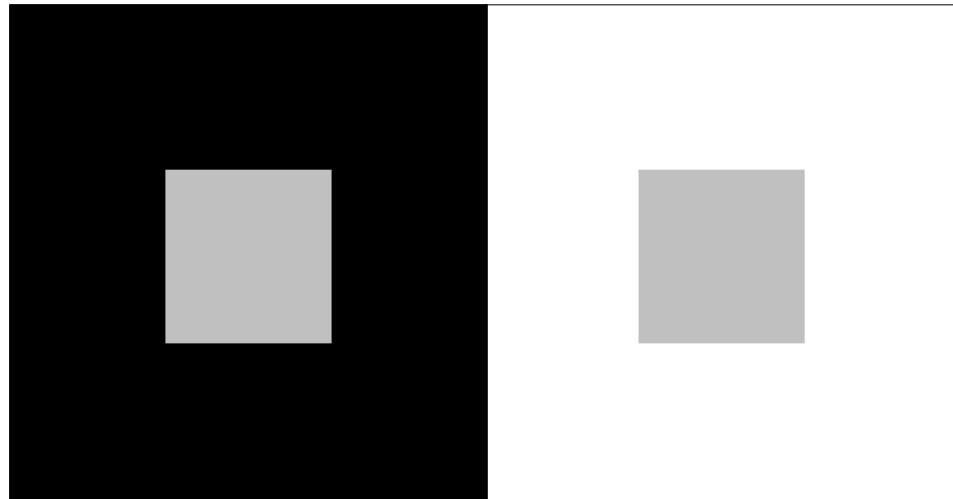
Introducción (II)

- Revisión del medio 'imagen'
 - Resolución espacial --> $I(x,y)$
 - Cuantización --> $I(x,y) \in [0, L]$ ($L+1$ valores)



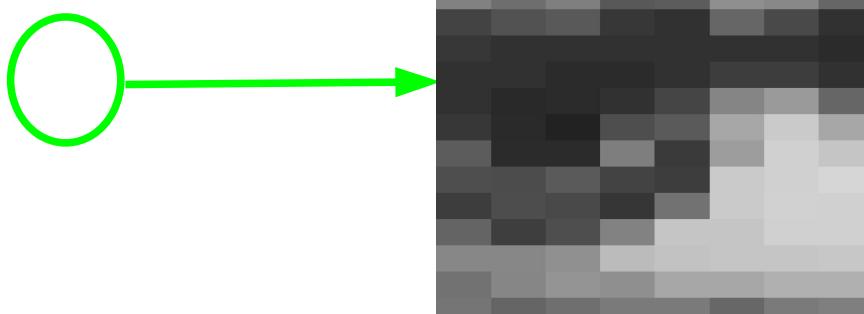
Introducción (III)

- Revisión del medio 'imagen'
 - Resolución espacial --> $I(x,y)$
 - Cuantización --> $I(x,y) \in [0, L]$ ($L+1$ valores)
 - Percepción e ilusiones



Introducción (IV)

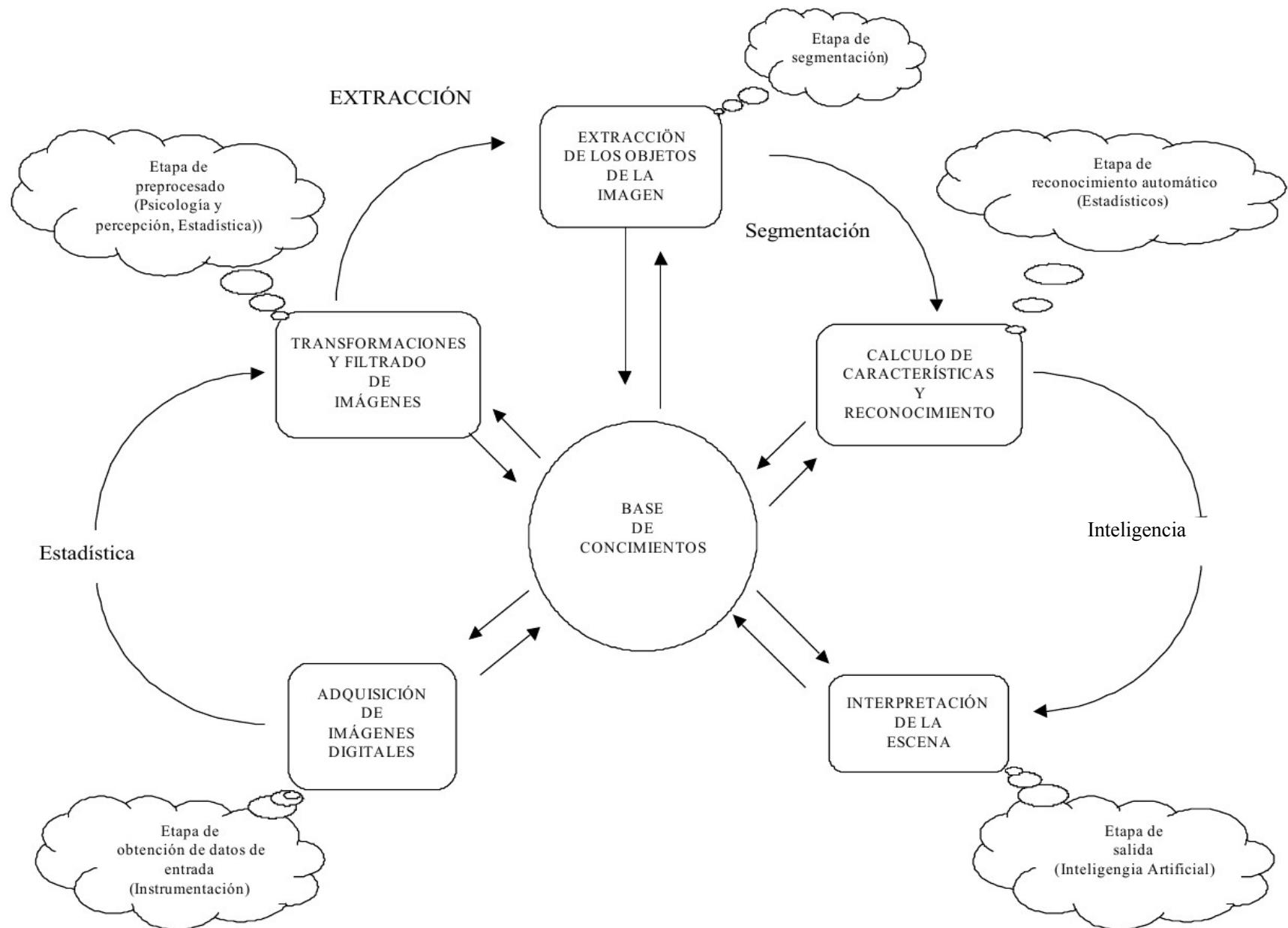
- Representación o modelado de imágenes
 - Modelos: descripción de una imagen
 - Dependencia de aplicación/percepción
 - Imágenes estáticas
 - Mapa de bits: valores de luminancia
 - Operadores locales --> Relación entre píxeles
 - Operadores globales --> Estadísticos
 - ® Vectoriales: objetos
 - ⌚ Secuencia de imágenes



129	110	127	88	92	142	136	102
67	82	89	55	49	101	73	49
55	49	49	49	49	49	49	43
49	49	43	43	49	61	61	49
49	41	43	49	69	132	154	102
55	42	34	78	90	166	202	167
92	44	43	126	58	156	208	197
78	76	90	67	61	200	208	214
61	77	73	54	114	201	209	208
100	62	78	130	197	197	208	208
135	135	144	187	194	197	197	199
114	134	148	143	167	167	176	176
117	101	111	122	122	103	122	127

Introducción (V)

- VxC



Introducción (VI)

- La **visión por computador (VxC)** se refiere a la adquisición y procesado de imágenes en formato digital de n-dimensiones mediante un computador.
- Aplicaciones
 - Industria
 - Medidas sin contacto, inspección de producción, biométrica, teledetección, ...
 - BD de imágenes
 - Médicos, diseñadores, ...
 - Realidad Aumentada (AR) / Realidad Virtual (RV)
 - Formas de comunicación y de interacción
Computación emocional y *avatars*.

Índice

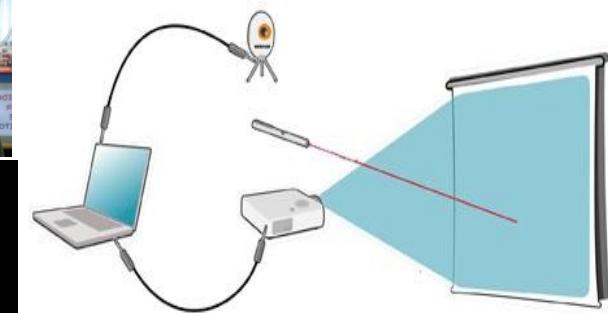
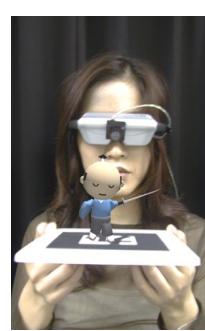
1. Introducción

1. Introducción a la VxC

2. Ejemplos de aplicaciones interactivas basadas en VxC

1. ARToolkit

2. OpenCV



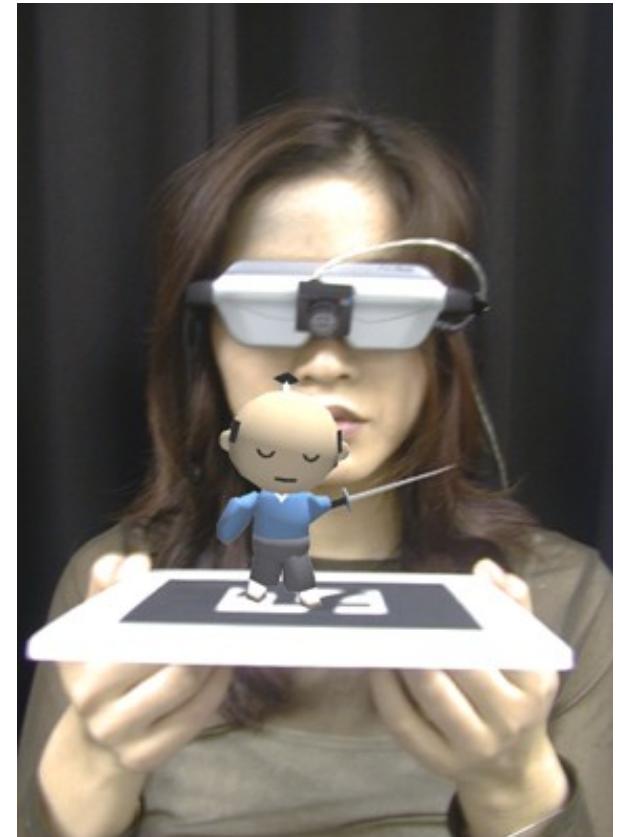
3. Operaciones básicas de análisis de imagen

2. Uso de OpenCV: operaciones básicas

3. Aplicaciones

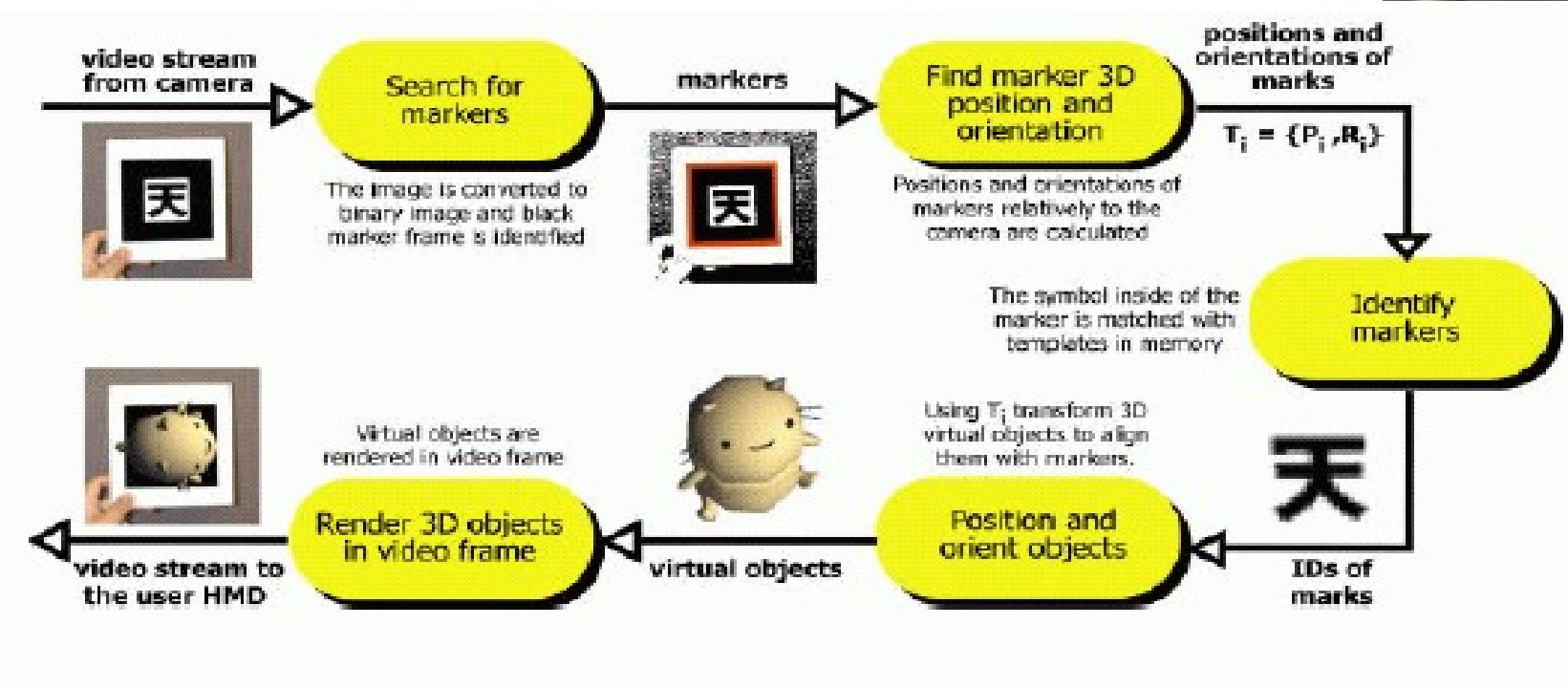
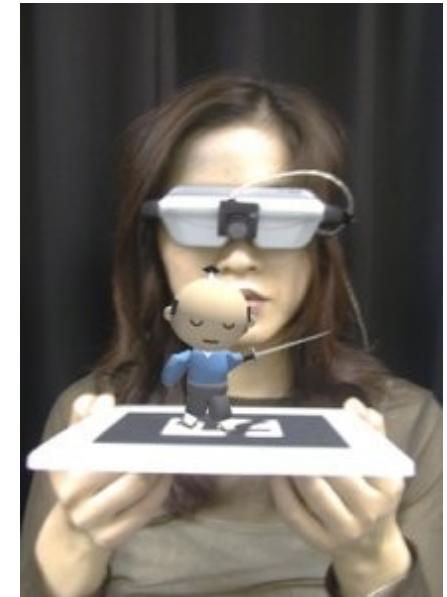
Realidad aumentada

- Integración e interacción.
 - Elementos reales y sintéticos
 - ARToolkit
- PFC de M. Ibáñez
 - Multiplataforma
 - Flexibilizar mecanismo: color
 - Explorar interacción
 - Explorar uso de medios
 - No usar dispositivos *especiales*



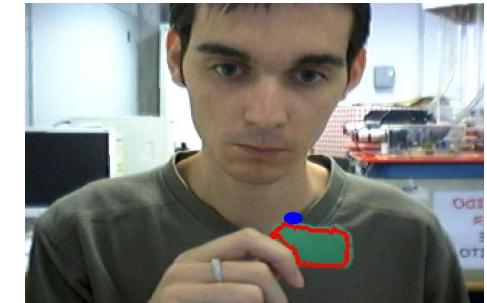
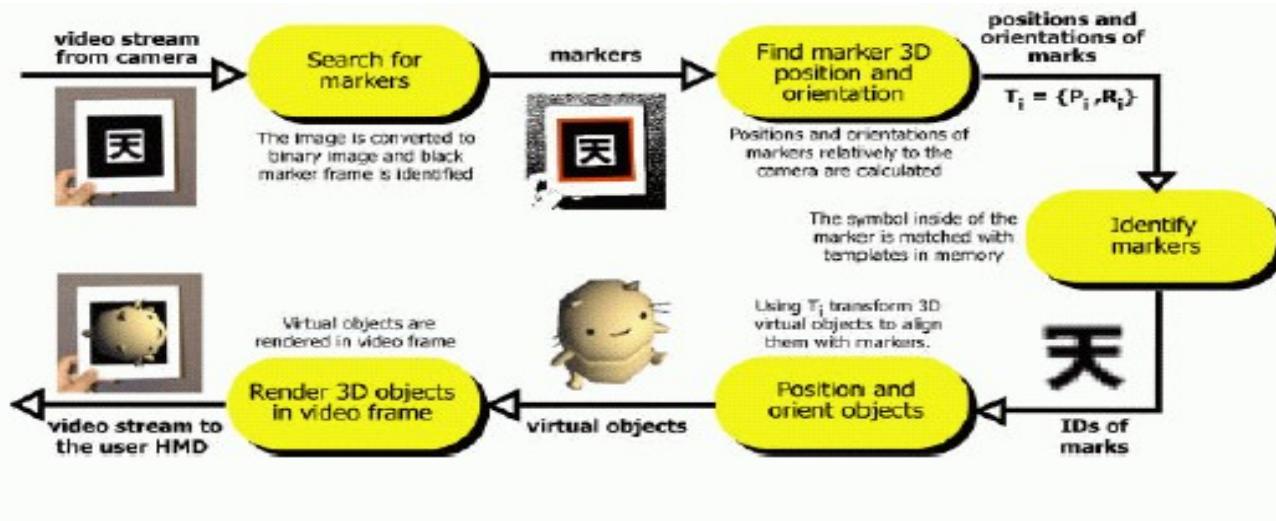
ARToolkit

- Origen: <http://www.hitl.washington.edu/artoolkit>
- Mecanismo:
 - Plantillas
 - Modo de trabajo

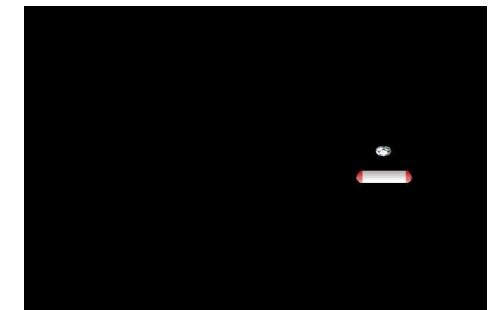


ARToolkit (II)

- Desarrollo propio sobre ARToolkit
 - Esquema

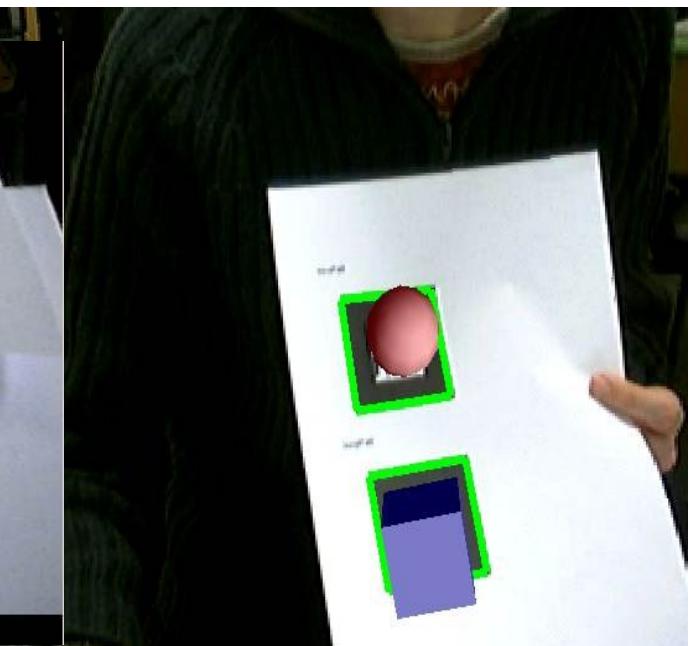
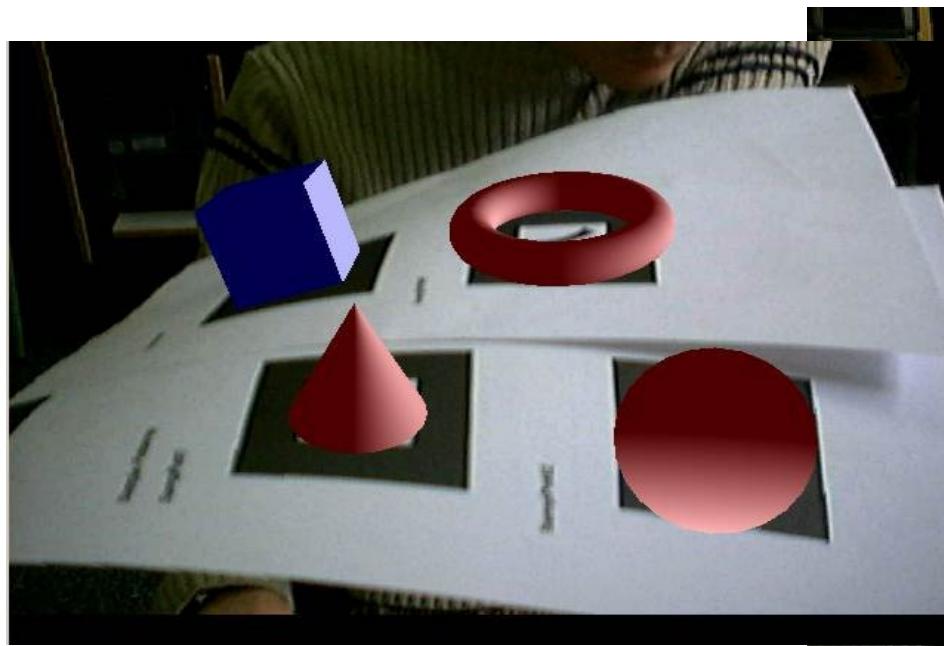
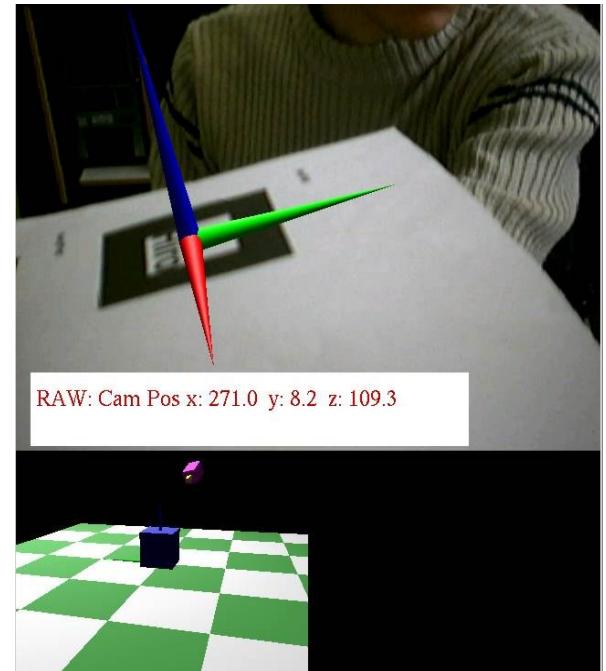
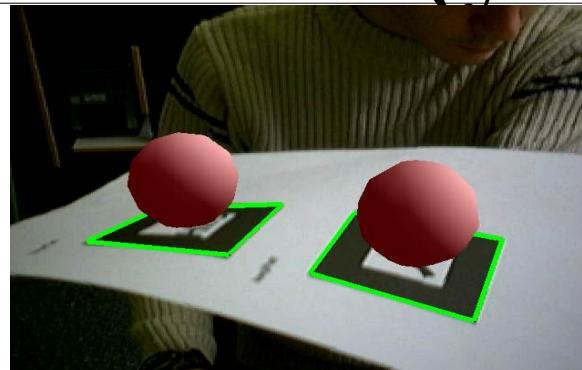


- Características
 - Multiplataforma
 - Tiempo real
 - Parametrizable por el usuario



ARToolkit (y II)

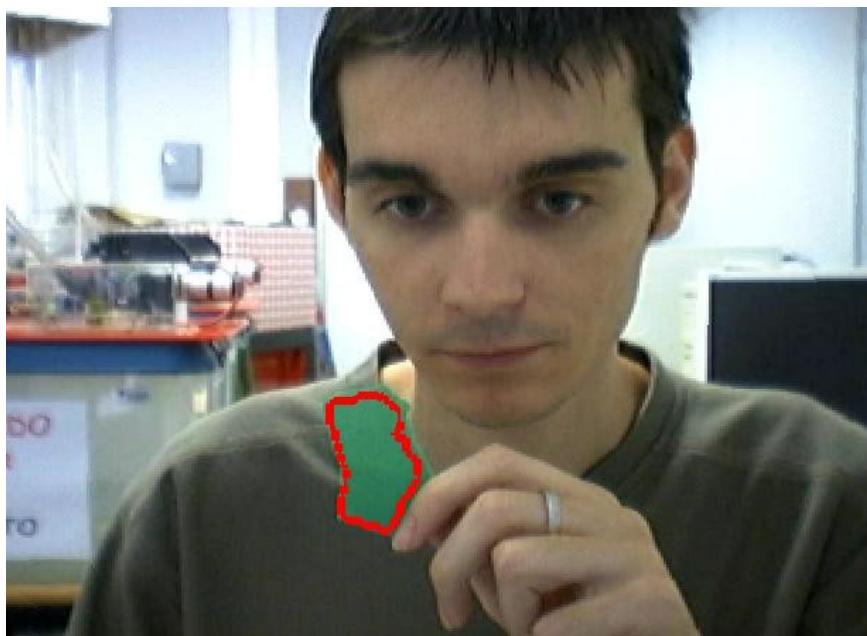
- Ejemplos
 - collideTestd
 - exviewd
 - loadMultipled
 - modeTextd
 - ...



Aplicaciones

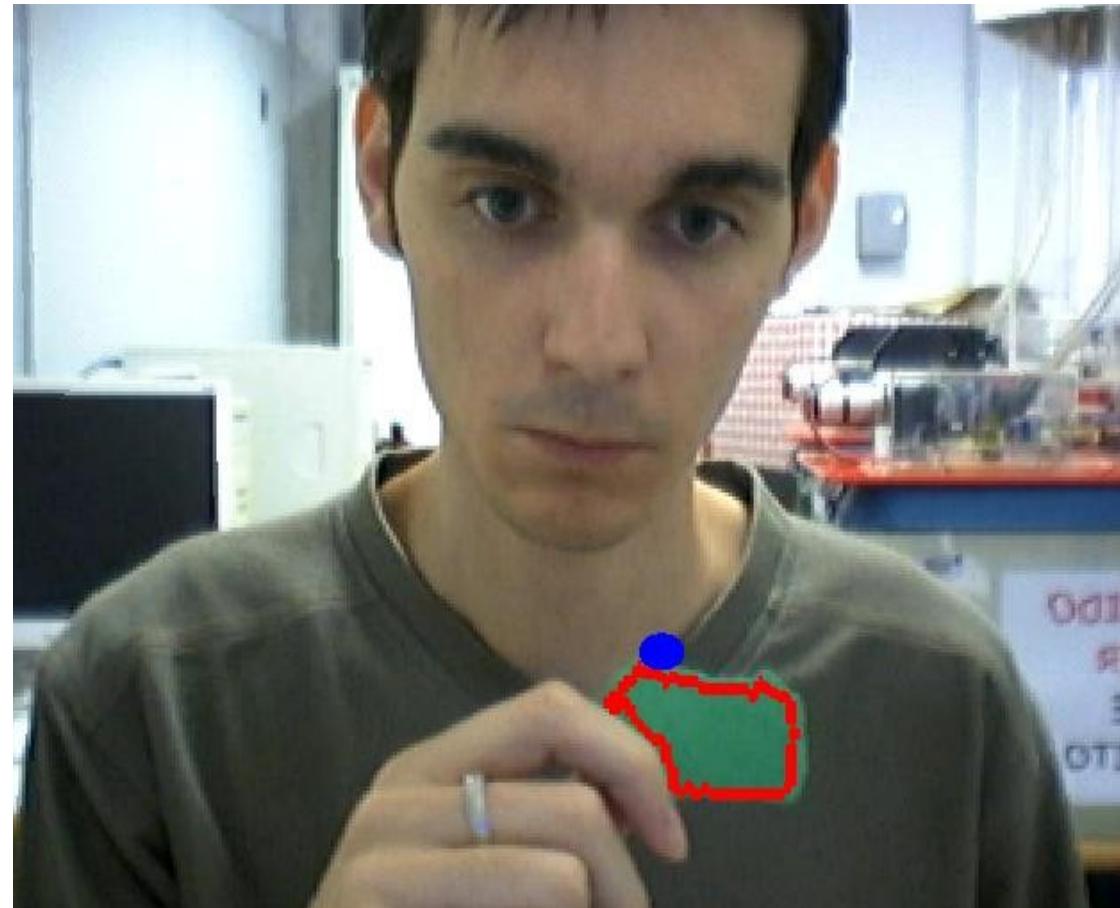
- Configuración

- Guardar (y obtener) información en (desde) fichero
- RGB? HSV!
- Adaptación al medio por definición de un rango.



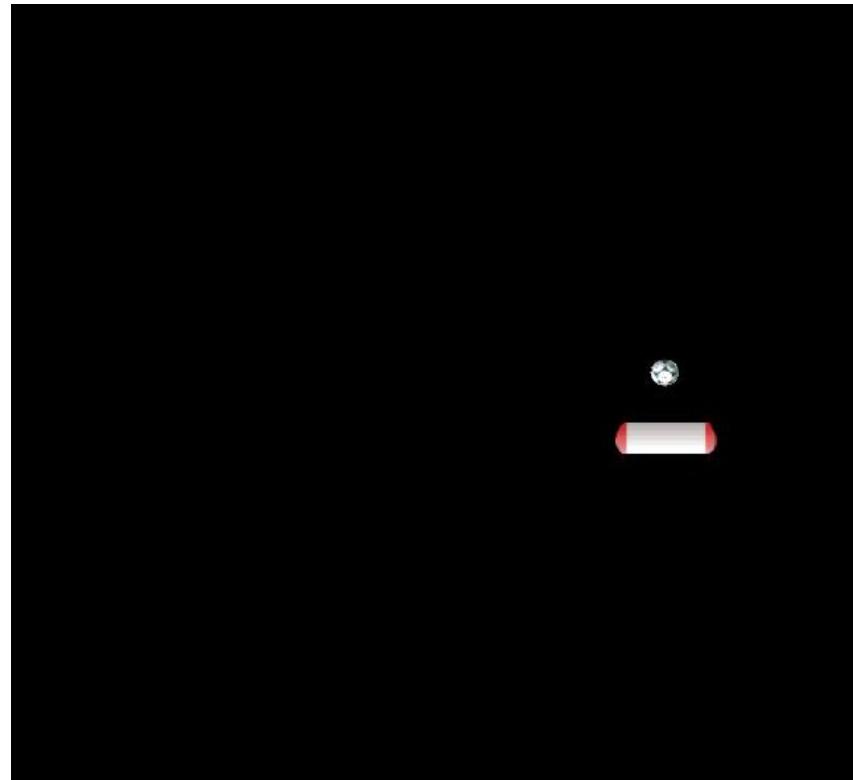
Aplicaciones (II)

- Juego de la pelota
 - Reconocer el elemento de interacción
 - Seguimiento del objeto *externo*
 - Colisión objeto *interno* y *externo*



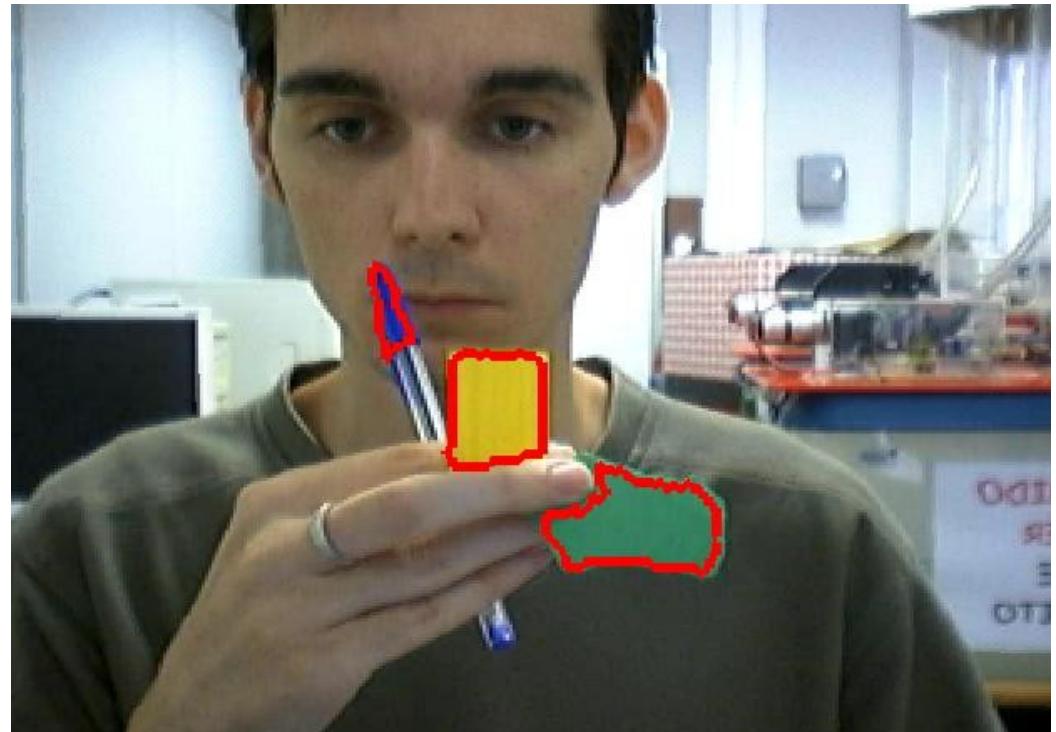
Aplicaciones (III)

- Juego de la pelota (versión final)
 - Eliminar el fondo
 - Ocultar vídeo
 - Añadir decoración
 - OpenGL: emplear texturas
 - Aspectos
 - Áreas rectangulares
 - Transparencias



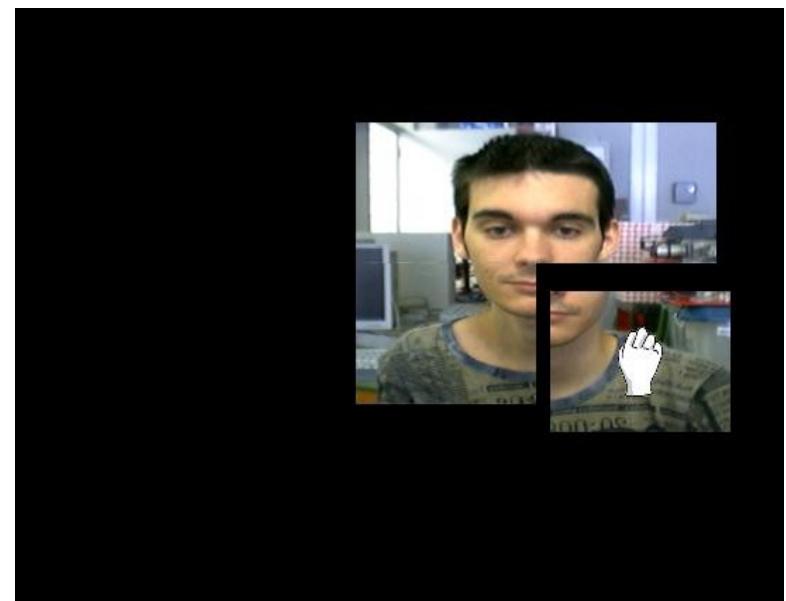
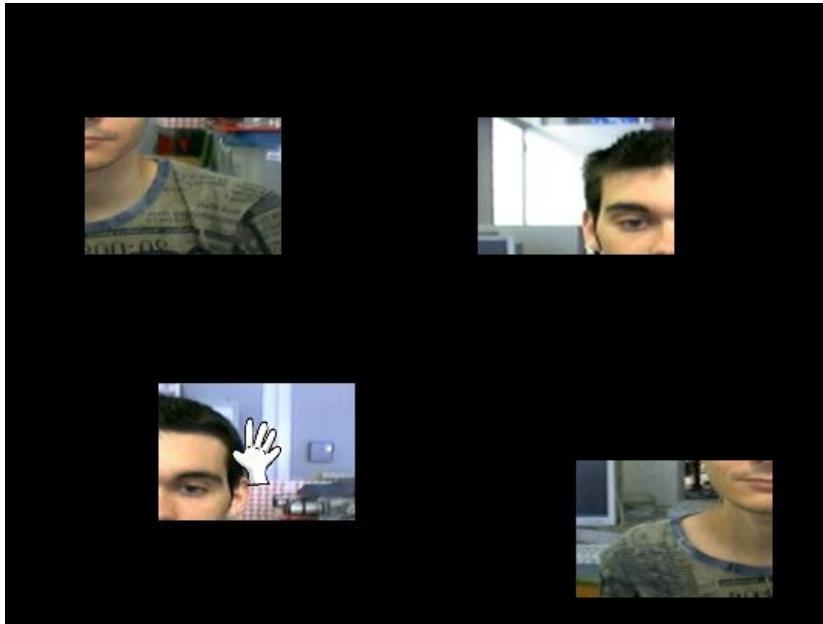
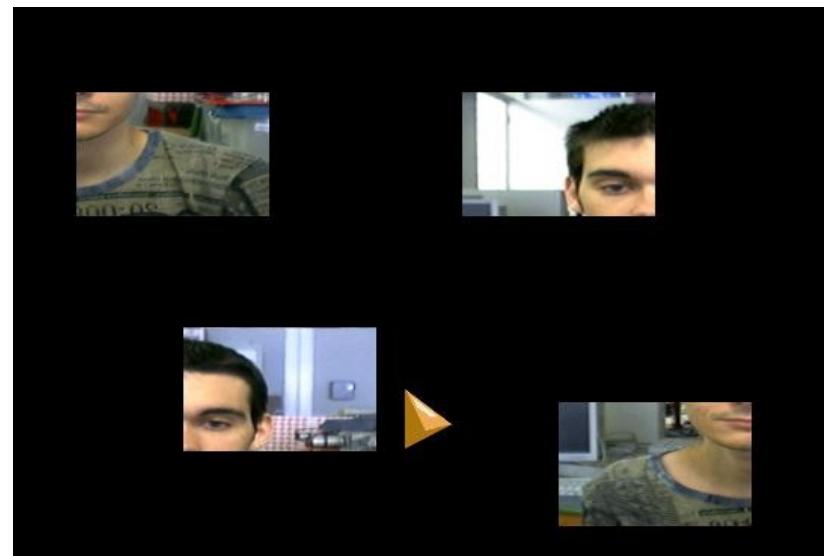
Aplicaciones (IV)

- Seguimiento de varios elementos
 - Extender la configuración
 - Añadir sonido
 - Solución multiplataforma:
 - *Allegro*



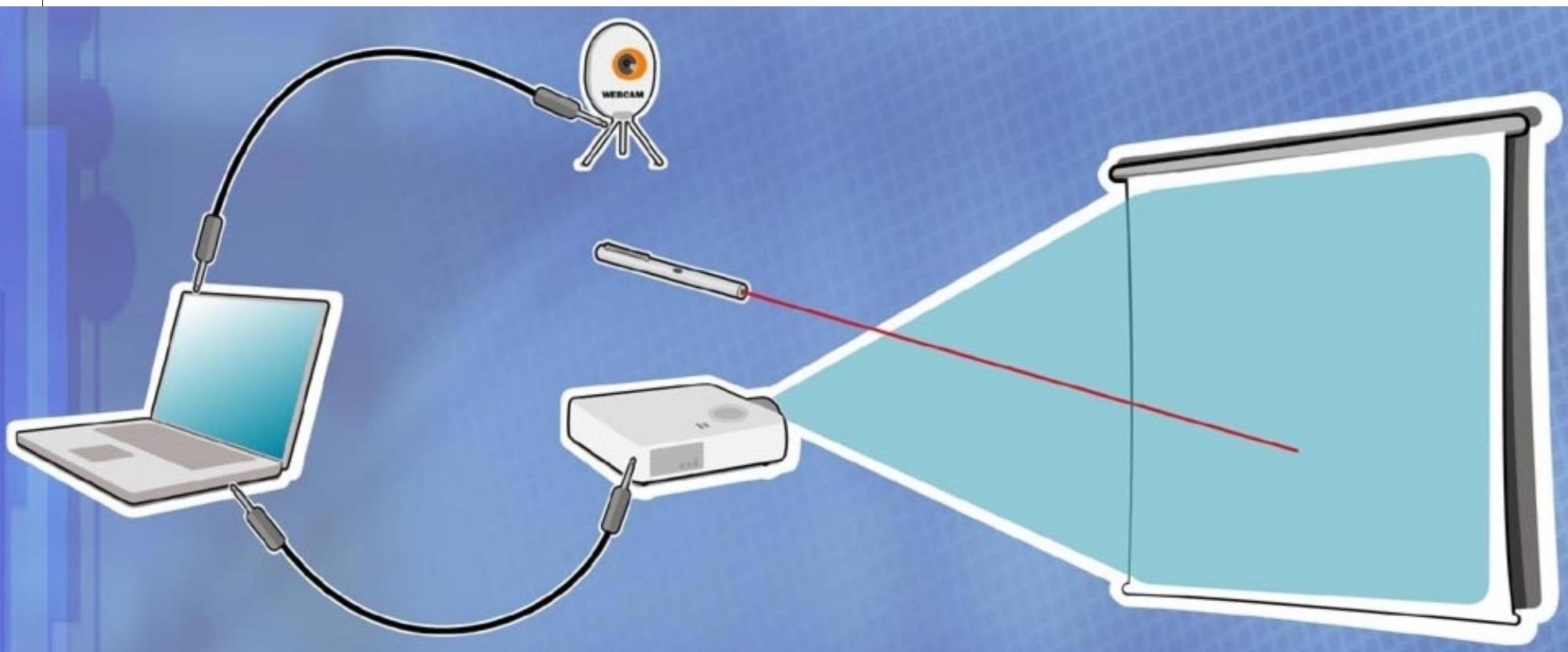
Aplicaciones (V)

- Ejemplo de interacción: simulación del *click*
 - Puzzle
 - Eventos de ratón:
 - Generación
 - Uso en la aplicación



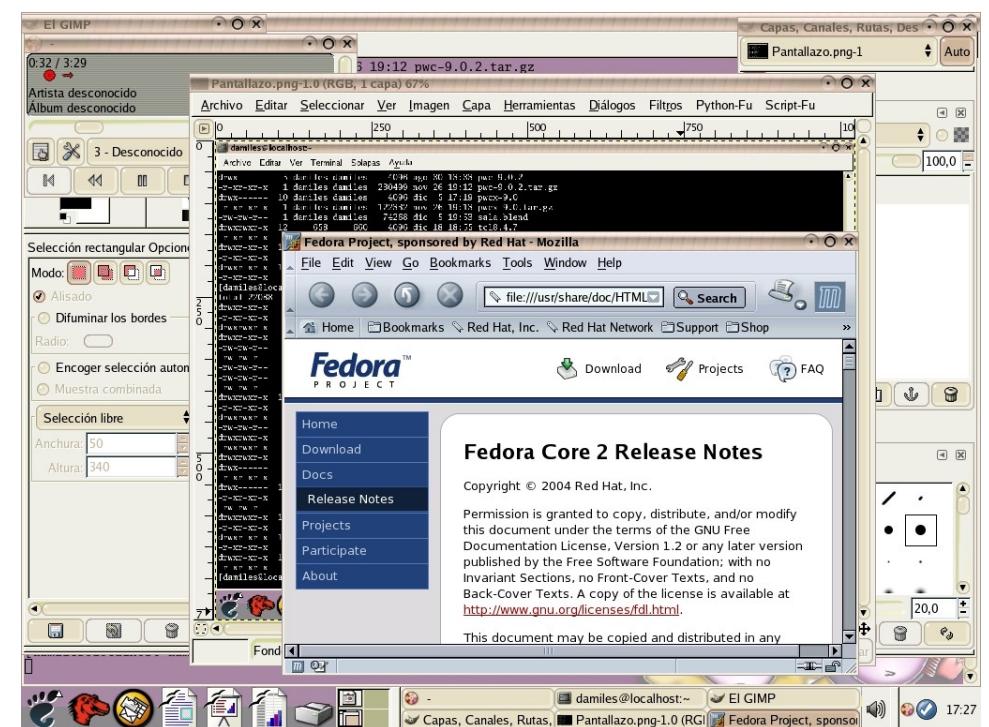
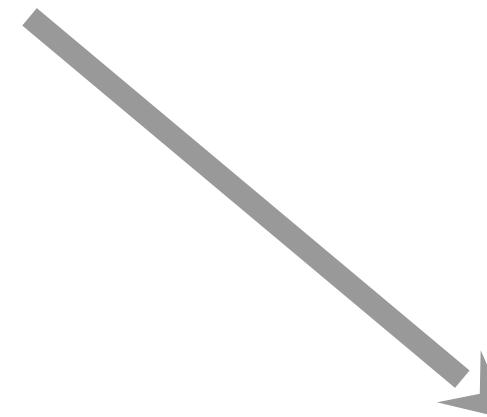
OpenCV

- La metáfora del escritorio
 - Extensión *al mundo real*



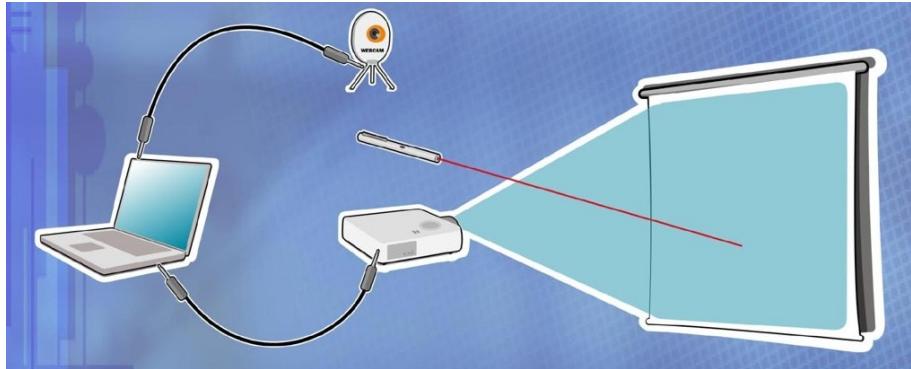
La metáfora del escritorio

```
drwx----- 5 damiles damiles 4096 ago 30 13:33 pwc-9.0.2
-r-xr-xr-x 1 damiles damiles 230499 nov 26 19:12 pwc-9.0.2.tar.gz
drwx----- 10 damiles damiles 4096 dic 5 17:19 pwcx-9.0
-r-xr-xr-x 1 damiles damiles 122332 nov 26 19:13 pwcx-9.0.tar.gz
-rw-rw-r-- 1 damiles damiles 74268 dic 5 19:53 sala.blend
drwxrwxr-x 12 658 660 4096 dic 18 18:55 tc18.4.7
-r-xr-xr-x 1 root root 3496364 dic 18 18:45 tc18.4.9-src.tar.gz
drwxr-xr-x 13 15399 18810 4096 jul 30 00:03 tk8.4.7
-r-xr-xr-x 1 root root 3186531 dic 15 21:04 tk8.4.7-src.tar.gz
drwxr-xr-x 18 damiles damiles 4096 dic 5 13:52 xawtv-3.94
-r-xr-xr-x 1 damiles damiles 553118 dic 5 13:51 xawtv-3.94.tar.gz
[damiles@localhost damiles]$ cat pepa
total 22088
drwxr-xr-x 5 damiles damiles 4096 dic 8 19:13 blender
-r-xr-xr-x 1 damiles damiles 4365186 nov 26 19:24 blender-2.35b-linux-glibc2.2.5-i386-static.tar.gz
drwxrwxr-x 6 damiles damiles 4096 dic 8 22:22 CvCam
drwxr-xr-x 3 damiles damiles 4096 dic 5 19:48 Desktop
-rw-rw-r-- 1 damiles damiles 80904 dic 8 19:32 gota.blend
-rw-rw-r-- 1 damiles damiles 80904 dic 8 19:32 gota.blendl
-rw-rw-r-- 1 damiles damiles 127280 dic 11 20:03 instantaneal.jpg
-rw-rw-r-- 1 damiles damiles 73160 dic 11 20:07 instantaneal_retocada.jpg
-rw-rw-r-- 1 damiles damiles 428109 dic 8 16:35 manualVision
drwxrwxr-x 11 damiles damiles 4096 dic 18 19:04 OpenCV-0.9.5
-r-xr-xr-x 1 damiles damiles 9653883 oct 31 16:56 OpenCV-0.9.5.tar.gz
drwxrwxr-x 2 damiles damiles 4096 dic 8 15:36 parchesOpenCvcam
-rw-rw-r-- 1 damiles damiles 0 eñe 8 17:24 pepa
drwxrwxr-x 4 damiles damiles 4096 dic 8 18:38 Proyecto
-rwxrwxr-x 1 damiles damiles 0 dic 8 19:32 prueba
drwxrwxr-x 6 damiles damiles 4096 dic 6 11:46 PruebaQT
drwxr-xr-x 5 damiles damiles 4096 ago 30 13:33 pwc-9.0.2
-r-xr-xr-x 1 damiles damiles 230499 nov 26 19:12 pwc-9.0.2.tar.gz
drwxr-xr-x 10 damiles damiles 4096 dic 5 17:19 pwcx-9.0
-r-xr-xr-x 1 damiles damiles 122332 nov 26 19:13 pwcx-9.0.tar.gz
-rw-rw-r-- 1 damiles damiles 74268 dic 5 19:53 sala.blend
drwxrwxr-x 12 658 660 4096 dic 18 18:55 tc18.4.7
-r-xr-xr-x 1 root root 3496364 dic 18 18:45 tc18.4.9-src.tar.gz
drwxr-xr-x 13 15399 18810 4096 jul 30 00:03 tk8.4.7
-r-xr-xr-x 1 root root 3186531 dic 15 21:04 tk8.4.7-src.tar.gz
drwxr-xr-x 18 damiles damiles 4096 dic 5 13:52 xawtv-3.94
-r-xr-xr-x 1 damiles damiles 553118 dic 5 13:51 xawtv-3.94.tar.gz
[damiles@localhost damiles]$
```

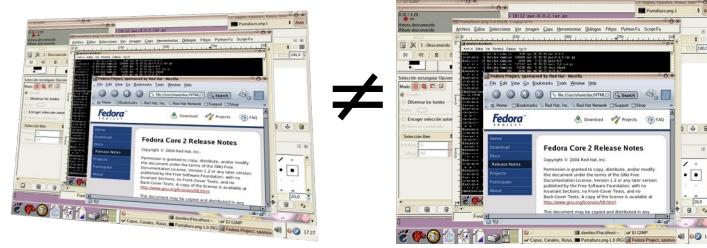


Cuestiones de implementación (II)

- Desarrollo propio sobre OpenCV
 - Esquema



- Características
 - Multiplataforma
 - Tiempo real
 - Parametrizable por el usuario



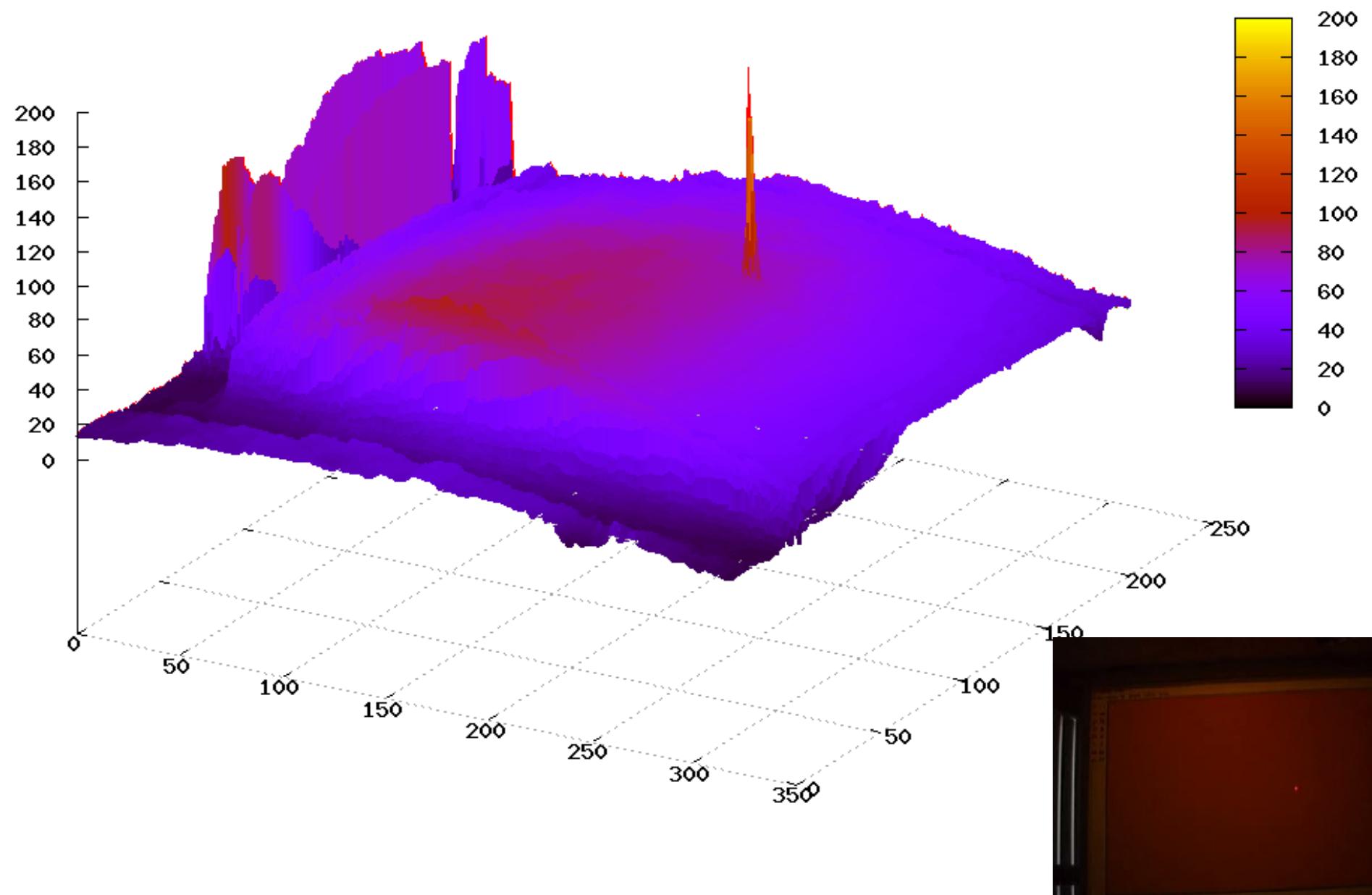
- Sobre la imagen ya corregida:
 - Se busca el objeto de interés (el punto de láser proyectado).
 - Se generan los eventos correspondientes en el sistema.

Más detalles en presentación externa

Del problema a la solución

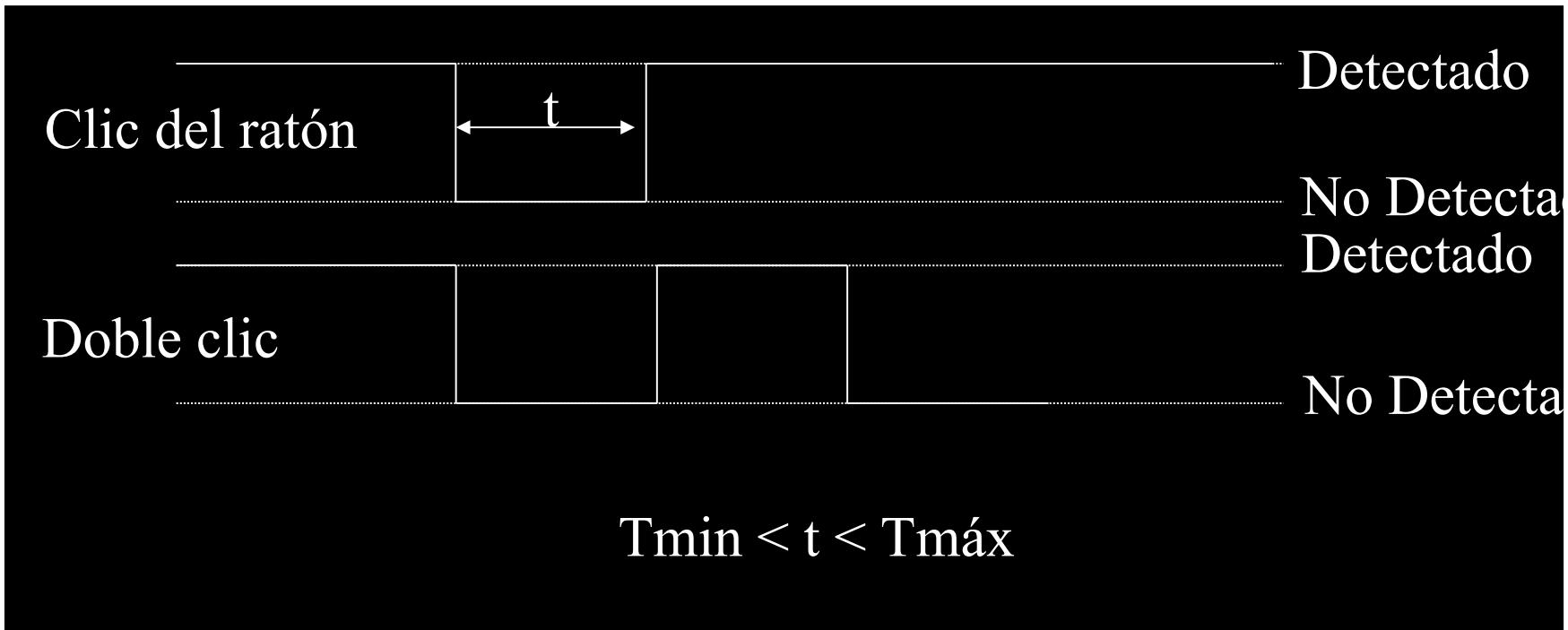
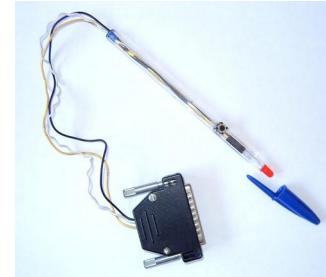
- ¿Porqué un láser es fácilmente detectable y es prácticamente imposible confundirlo con el escritorio proyectado?
- Veamos la relación de iluminancia.
 - Proyector: Iluminancia= $E=Lumen/m^2$
 $E=2000 \text{ Lumen} / (3x2) \text{ m}^2 = 333,3 \text{ lux}$
 - Puntero Láser Cat II con 1mW max= $0,683 \text{ Lumen}$
 $E=0,683 \text{ Lumen} / (0,01x0,01) \text{ m}^2 = 6830 \text{ lux}$

$333,3 << 6830$



■ Eventos: hardware y software

- Diagrama de estados



- Error total \approx E. humano + E. de calibración
(resolución y perspectiva)

Error de precisión total

- Debido entonces al error humano y al tamaño de captura, como de la calibración existen unos errores de precisión
- No se puede entonces llegar a todas las zonas.
- Error total \approx E. humano + E. de calibración
(resolución y perspectiva)

Índice

1. Introducción

1. Introducción a la VxC

2. Ejemplos de aplicaciones interactivas basadas en VxC

3. Operaciones básicas de análisis de imagen

1. Análisis de imagen

2. Extracción de información

2. Uso de OpenCV: operaciones básicas

3. Aplicaciones

Análisis de la imagen

- Relaciones entre los puntos de la imagen
 - Descripciones a partir de los valores de los puntos, considerados:
 - Aislados
 - Vecindad
- Distribución de una característica de los puntos de la imagen
 - Caracterizaciones debidas a variables estadísticas:
 - Ámbito
 - Globales ó locales (texturas)
 - Tipo de resultado
 - Escalares o vectoriales

Análisis de la imagen (II)

- Relaciones entre los puntos de la imagen

- Espacialmente

- Aislados

- Remapear los valores asociados a cada punto

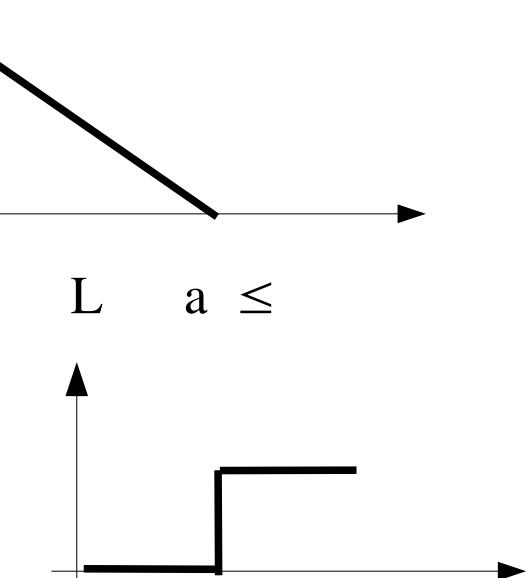
$$v = f(u) \quad / \quad u, v \in [0, L]$$

- Ejemplos:

Negativo: $dst(x,y) = L - fte(x,y)$

Umbralizar: $dst(x,y) = \begin{cases} 0 & 0 \leq fte(x,y) < a \\ fte(x,y) & \leq b \end{cases}$

- Vecindad



Análisis de la imagen (III)

- Relaciones entre los puntos de la imagen
 - Espacialmente

- Aislados
- Vecindad
 - Generar nueva información
 - No precisamente información de color
 - ¿Cuántos? 4-conectados y 8-conectados

$p_{i-1,j}$

$p_{i,j-1}$

$p_{i,j}$

$p_{i,j+1}$

$p_{i+1,j+1}$

$p_{i-1,-1j}$

$p_{i-1,j}$

$p_{i-1,j+1}$

$P_{i,j-1}$

$p_{i,j}$

$P_{i,j+1}$

$P_{i+1,j-1}$

$p_{i+1,j}$

$p_{i+1,j+1}$

- Operación: convolución
- Operador: máscara o filtro

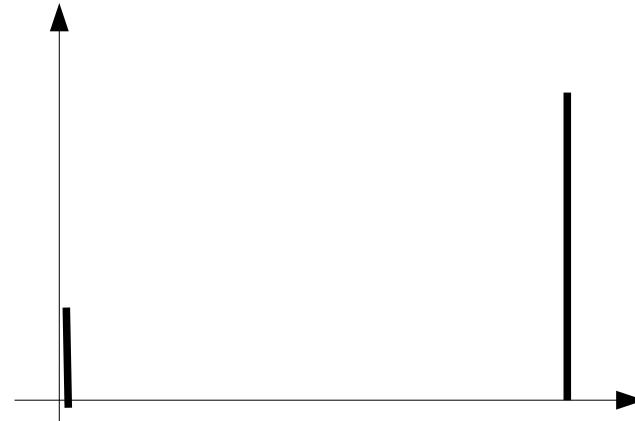
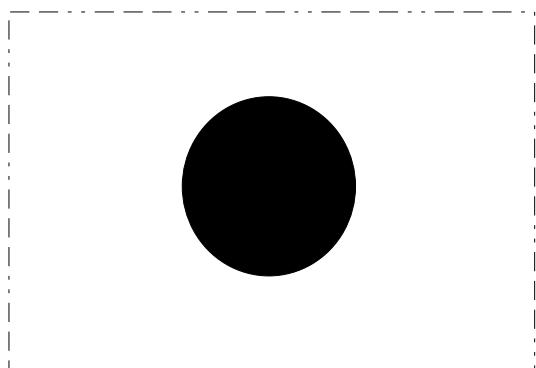
Análisis de la imagen (IV)

- Distribución de una característica de los puntos de la imagen
 - De la frecuencia de aparición de los niveles de gris (o de color).
 - En función de la dimensión
 - Escalares
 - Medias, covarianzas, ...
 - Vectoriales
 - Histograma
 - Función de distribución de probabilidad de los niveles de gris
 - En valor absoluto: entre 0 y MxN
 - En porcentaje: normalizar entre 0 .. 1

Análisis de la imagen (V)

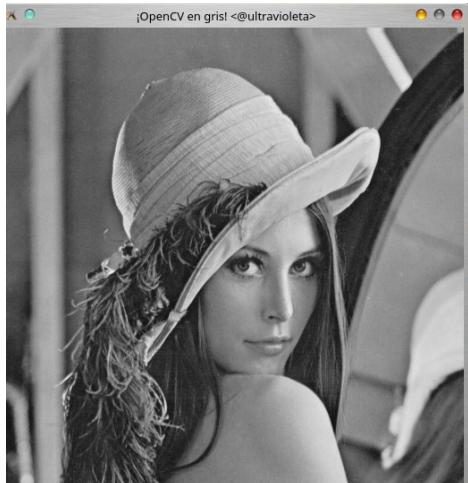
- Distribución de una característica de los puntos de la imagen
 - Caracterizaciones globales
 - Operadores estadísticos: el Histograma
 - De niveles de gris: $h \sim [0, 1, 2, \dots, L - 1]$

$h(k) = \text{número de puntos de la imagen que cumplen}$
 $I(x,y) = k \text{ con } k \in [0, L-1]$



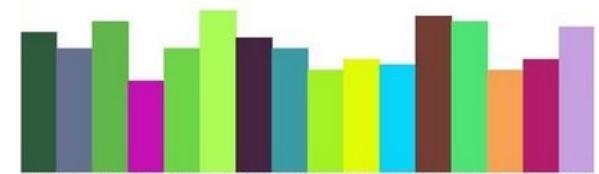
Análisis de la imagen (y VI)

- Histogramas 1D (**dim = 1**)
 - Función de densidad o función de distribución de la probabilidad de una variable
 - Ej. intensidad de la imagen



- Se suelen agrupar en bandas (*bins*)

- P. ej $[0,255] = [0,15] \cup [16,31] \cup \dots \cup [240,255]$
 $\text{range} = \text{bin1} \cup \text{bin2} \cup \dots \cup \text{binn}=15$



254	143	203	170	100	229	177	220	162	0	229	142	130	04	0	03	20	8	90	82
27	68	231	75	141	107	149	210	13	230	141	35	69	242	110	208	241	0	33	88
54	42	17	215	239	254	47	+1	98	180	05	230	235	77	182	206	70	110	192	100
0	108	192	71	104	199	09	17	37	230	18	147	174	143	211	179	180	192	69	
170	20	239	192	190	132	61	248	22	134	03	130	117	254	176	239	189	234	51	204
232	25	0	163	174	129	61	30	110	189	0	173	197	183	133	43	22	87	88	119
235	25	151	180	159	01	239	179	159	04	39	21	97	101	99	37	109	149	02	154
159	242	54	0	151	169	47	138	254	225	154	91	181	131	15	135	55	232	205	
220	114	79	129	16	0	201	68	89	107	58	44	230	14	39	1	65	5	231	216
05	168	237	189	80	101	131	241	65	130	124	151	111	25	190	4	249	26	117	145
152	195	229	76	93	217	210	103	118	77	38	49	2	9	314	181	205	118	130	33
150	34	178	196	22	149	57	223	232	113	32	45	17	15	31	179	100	119	209	01
224	118	124	172	75	29	69	160	167	195	41	4	8	170	158	101	131	25	112	
238	63	38	7	93	69	173	183	98	257	67	227	18	218	248	237	73	192	201	146
09	195	224	227	140	22	31	110	234	54	182	116	23	47	98	242	189	192	110	248
140	37	101	239	246	145	122	64	27	50	229	1	220	49	91	109	98	90	195	
221	1	178	139	131	89	97	174	249	192	77	115	223	186	182	82	65	220	93	196
179	189	223	239	97	102	148	78	170	19	17	4	184	176	183	102	83	81	132	226
173	137	185	242	181	181	214	49	74	238	197	37	98	102	15	217	148	0	102	198
03	9	17	22	16	21	70	21	78	241	194	216	93	93	208	102	152	215	118	47

Índice

1. Introducción

1. Introducción a la VxC

2. Ejemplos de aplicaciones interactivas basadas en VxC

3. Operaciones básicas de análisis de imagen

1. Análisis de imagen

2. Extracción de información

2. Uso de OpenCV: operaciones básicas

3. Aplicaciones

Extracción de información

- Relaciones entre los puntos de la imagen

- Aislados
- Vecindad

- Modo de aplicación: operación

- Convolución

$$\text{dst}(x,y) = \sum_k \sum_l (a(k,l) * fte(x-k, y-l)) ; (k,l) \in W_{\text{fte}}$$

- Pesos:

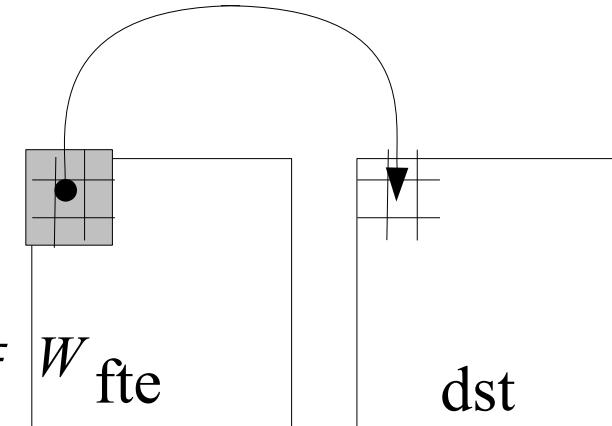
- $a(k,l)$ (gnralm., ≤ 1)

- es la aportación de cada punto vecino: promediado.

- Operación de filtrado:

- Decidir la máscara

- Aplicación



$a_{0,0}$	$a_{0,1}$	$a_{0,2}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$
$a_{2,0}$	$a_{1,2}$	$a_{2,2}$

Extracción de información (II)

- Relaciones entre los puntos de la imagen

- Aplicación: escalado

- Descripción: ampliación por interpolación
 - Algoritmo:

$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$
$\frac{1}{2}$	1	$\frac{1}{2}$
$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$

¿Inicialización?

Función de convolución

```
funcion convolucion(mascara, imgFte, x, y, imgDst)
```

para i = 0 hasta nFilasMascara

para j = 0 hasta nColsMascara

imgDst(x,y) += mascara(i, j) * imgFte(x+i, y+j)

fpara

fpara

ffuncion

Aplicar convolución

para i = 1 hasta nFilas - 1

para j = 1 hasta nCols - 1

convolucion(mascara, F, i, j, D)

fpara

fpara

Extracción de información (III)

- Relaciones entre los puntos de la imagen

- Aplicación:

- Escalado: ampliación por interpolación
 - Detección de

- Puntos

-1	-1	-1
-1	8	-1
-1	-1	-1

- Líneas

- Horizontales
 - Verticales
 - ¿Diagonales?, ...

-1	-1	-1	-1	2	-1
2	2	2	-1	2	-1
-1	-1	-1	-1	2	-1

Extracción de información (IV)

- Relaciones entre los puntos de la imagen

- Aplicación:

- Escalado: ampliación por interpolación
 - Detección de puntos, líneas, ...
 - Extracción de bordes
 - Roberts

$$\begin{matrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{0,1} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,0} & a_{1,2} & a_{2,2} \end{matrix}$$

$$\begin{matrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{matrix}$$

$$\begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix}$$

$$\begin{matrix} 0 & +1 \\ -1 & 0 \end{matrix}$$

- Prewitt

$$\begin{matrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{matrix}$$

$$\begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix}$$

- Sobel

$$\begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix}$$

$$\begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$$

Extracción de información (y V)

- Distribución de una característica de los puntos de la imagen
 - Operador: estadísticos
 - Aplicación: segmentación ← umbralización
 - Determinar un valor umbral (T) que separe las clases de objetos presentes en la imagen

$$dst(x,y) = \begin{cases} 0 & \text{si } fte(x,y) < T \\ 1 & \text{si } fte(x,y) \leq T \end{cases}$$

- A partir de aquí:
 - Etiquetado
 - Detección del centro de masas
 - Caracterización de los objetos
 - Rectángulo, ejes, elongación, ...

Índice

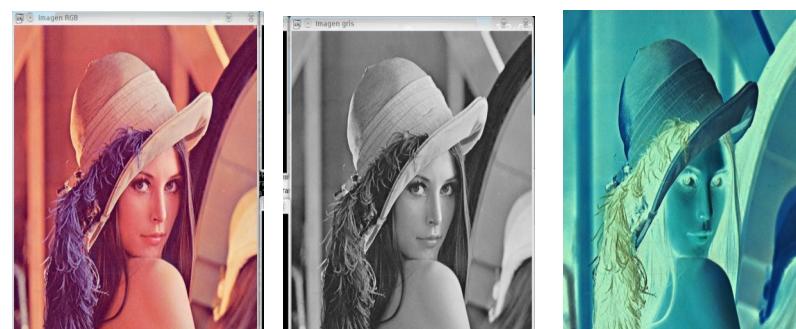
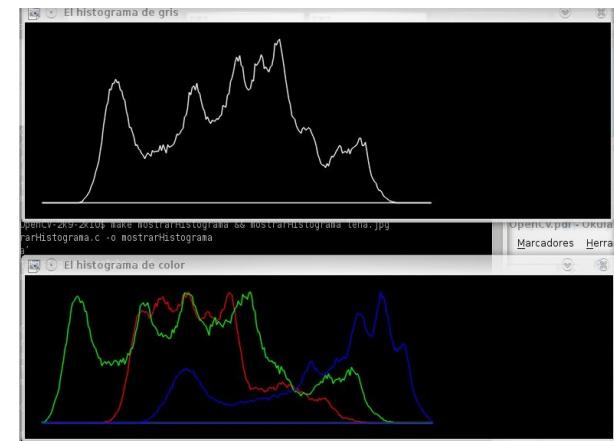
1. Introducción a la Visión por Computador
2. Uso de OpenCV. Operaciones básicas

1. Introducción

2. Ejemplos

1. Definición de una imagen
2. Acceso a las componentes
3. Primitivas de dibujo
4. Manejo de eventos
5. Imágenes y ficheros
6. Procesar una secuencia
7. Histogramas
8. Procesamientos puntuales
9. Procesos de vecindad

3. Aplicaciones



Introducción a OpenCV

- Ejemplos de APIs para VxC: bibliotecas de funciones
 - NetPBM
(~ 1988) <<http://netpbm.sourceforge.net/>>
 - *ImageMagick Studio*: *convert, montage, display, animate, ...*
(~ 1990) <<http://www.imagemagick.org>>
 - ***Open Computer Vision Library (OpenCV)***
<(~1999) <<http://opencv.org>>, <<http://sourceforge.net/projects/opencvlibrary>>>
 - *ARToolkit*
(~ 2003) <<http://www.hitl.washington.edu/artoolkit/>>
- *Tendencias actuales: servicios en la red.*
 - Comparativa *Cloudy Vision Output*
 - *Clarifi, Rekognition | Amazon,*
Visual Recognition | IBM Watson,
Computer Vision. < Cognitives Services | Microsoft Azure,
Cloud Vision. AI & Machine Learning Products | Google Cloud,
Cloud Sight

Introducción a OpenCV

- Tendencias actuales

Comparison of Cloud APIs for CV						
	Google	Azure	Rekognition	clarifai	IBM	KAIROS
FACE DETECTION	✓	✓	✓	✓	✓	✓
FACE RECOGNITION	✗	✓	✓	✗	✗	✓
FACIAL LANDMARKS	✓	✓	✓	✗	✗	✓
FEATURE DETECTION	✓	✓	✓	✓	✗	✓
SIMILAR FACES	✗	✓	✓	✗	✗	✓
EMOTION	✓	✓	✓	✗	✗	✓
LABEL DETECTION	✓	✓	✓	✓	✓	✗
LANDMARKS	✓	✓	✗	✗	✓	✗
CELEBRITIES	✗	✓	✓	✓	✗	✗
LOGO DETECTION	✓	✗	✗	✓	✗	✗
OCR	✓	✓	✓	✗	✓	✗
NSFW	✓	✓	✓	✓	✓	✗
IMAGE ANALYSIS	✓	✓	✓	✓	✓	✗
VIDEO ANALYSIS	✓	✓	✓	✗	✗	✓ for faces
CUSTOM MODEL CREATION	✗	✓	✗	✓	✓	✗

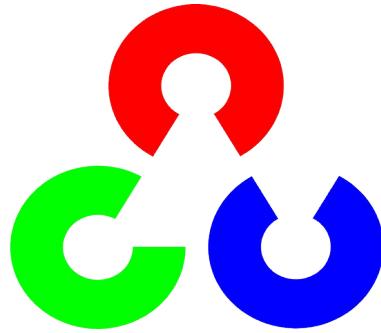
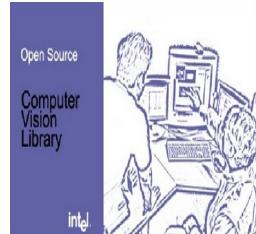
Created by ActiveWizards

Fuente: Igor Bobriakov. (2018). Comparison of Top 6 Cloud APIs for Computer Vision

<<https://medium.com/activewizards-machine-learning-company/comparison-of-top-6-cloud-apis-for-computer-vision-ebf2d299be73>>

Introducción a OpenCV

- Evolución histórica de OpenCV



- Últimas versiones/revisiones
 - 4-0.0-alpha (20-09-2018)
 - 2.4.13.6 (26-02-2018)
 - 3.4.5 (22-12-2018)
 - 4.0.1 (22-12-2018)

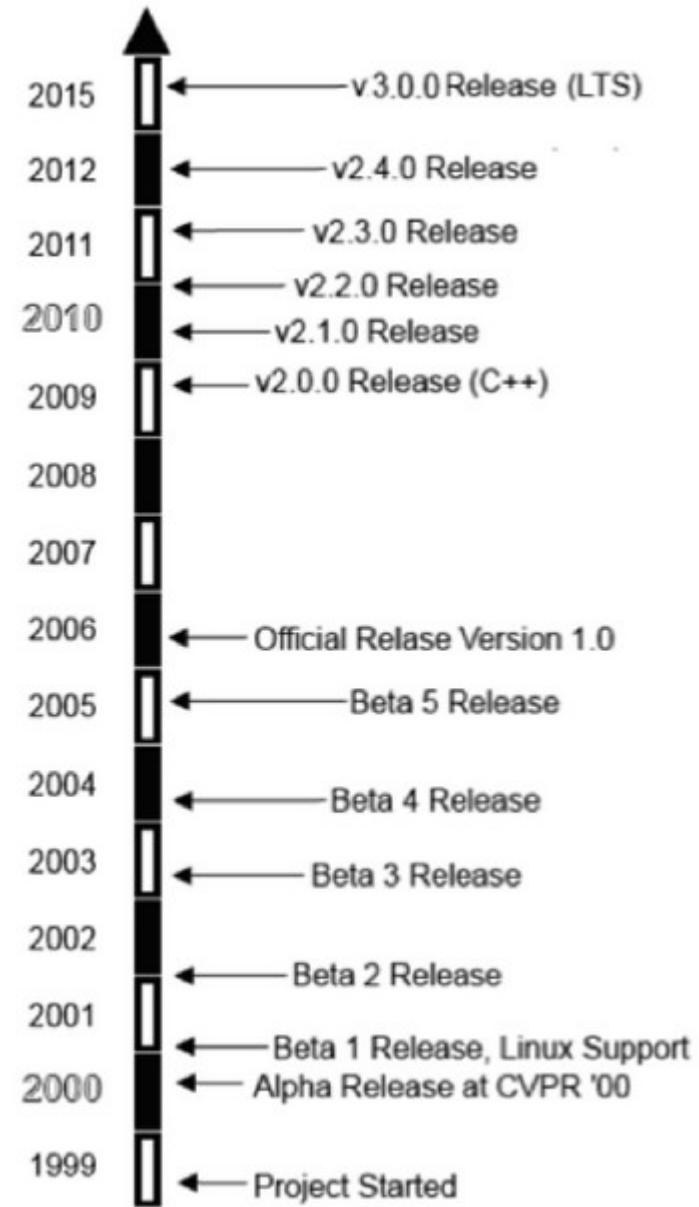


Figure 1-3. OpenCV timeline

Fuente: A. Kaehler y G. Bradski. (2017). Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library.

Y “Releases – OpenCV Library” <<https://opencv.org/releases.html>>

Introducción

- Arquitectura de OpenCV

OpenCV Block Diagram

OpenCV is built in layers. At the top is the OS under which OpenCV operates. Next comes the language bindings and sample applications. Below that is the contributed code in *opencv_contrib*, which contains mostly higher-level functionality. After that is the core of OpenCV, and at the bottom are the various hardware optimizations in the *hardware acceleration layer* (HAL). **Figure 1-4** shows this organization.

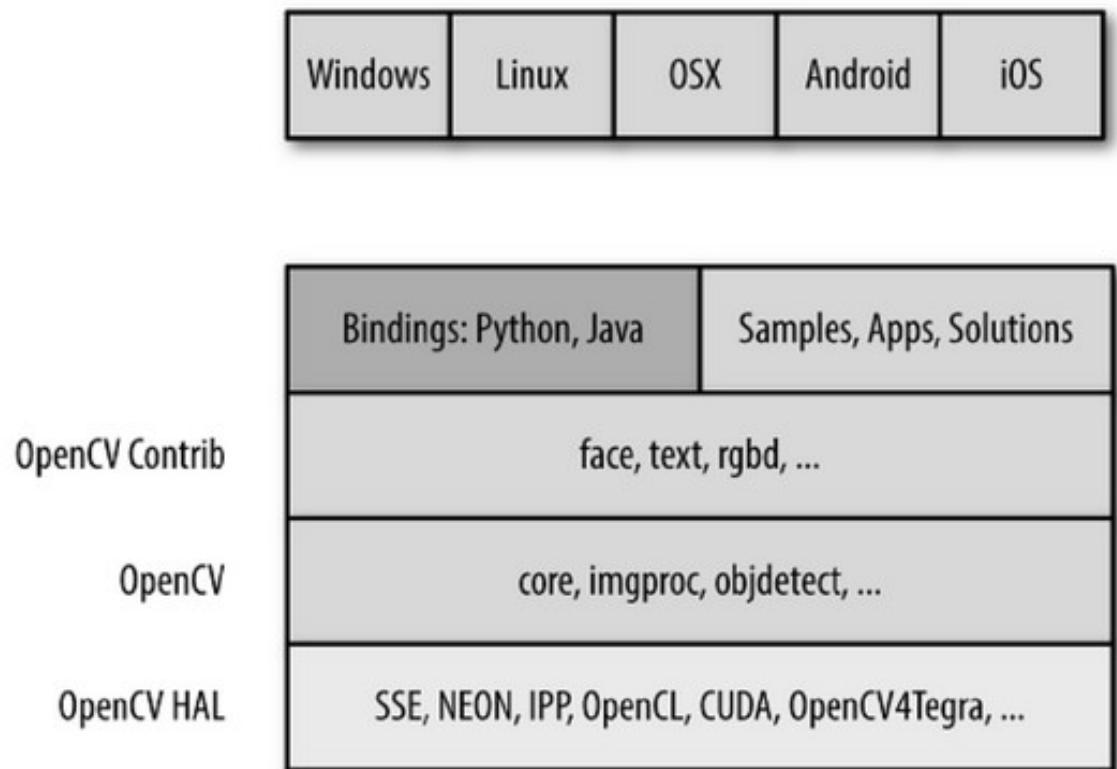


Figure 1-4. Block diagram of OpenCV with supported operating systems

OpenCV: introducción

- Open Computer Vision Library
 - <http://sourceforge.net/projects/opencvlibrary/>
- OpenCV
 - *Home* <<http://opencv.org>>
 - Documentación <<http://docs.opencv.org/>>
 - Tutoriales <<http://docs.opencv.org/>>
- Biblioteca de código abierto para tareas de VxC.
- Originalmente : Intel. Publicada bajo licencia BSD.
- Multiplataforma.
- Módulos

OpenCV: introducción (y II)

- Open Computer Vision Library
- Módulos
 - *cv*: *funciones básicas*
 - *cxCore*: estructuras de datos (memoria y disco), álgebra (vectores y matrices), primitivas gráficas, XML
 - *cvReference*: procesado de imágenes, movimiento, reconocimiento de patrones, calibración.
 - *cvAux*: estéreo, morphing, ...
 - *HighGui*: ventanas, E/S (usuario, fichero, vídeo)
 - *ML*: *machine learning*

Ej. OpenCV 3.2: definición de una imagen

```
cv::Mat // <https://docs.opencv.org/3.2.0/d3/d63/classcv\_1\_1Mat.html#details>

// Public Member Functions

_tp & at( int row, int col ) // Ejs.: _Tp->at<uchar>(y, x) = x; ó _Tp-> at<Vec3b>(y, x)[R] = y;
int channels() // Generalmente 1,2,3 o 4 canales
void create( Size size, int type )
int depth() // pixel depth in bits: CV_8U ... CV_64F

// Static Public Member Functions

...
static MatExpr ones( int rows, int cols, int type )
static MatExpr zeros( int rows, int cols, int type )
...

// Public attributes

int cols // image width in pixels
uchar * data
int dims
int flags
int rows // image height in pixels
MatSize size;
...
```

Ej. OpenCV 3.2: cargar y crear imágenes

```
#include <opencv2/opencv.hpp>

cv::Mat imgOrg, imgDst;

main( int argc , char *arg[], char *env[] ) {

    imgOrg = cv::imread( "nombreFichero" [, type] );

        // type:
        // CV_LOAD_IMAGE_ANYDEPTH - If set, return 16-bit/32-bit image ..., otherwise convert it to 8-bit.
        // CV_LOAD_IMAGE_COLOR - If set, always convert image to the color one
        // CV_LOAD_IMAGE_GRAYSCALE - If set, always convert image to the grayscale one

    imgDst=cv::Mat::create( imgOrg.rows, imgOrg.cols, imgOrg.depth() );
                    // OpenCV >= 3.0 ya no es necesario

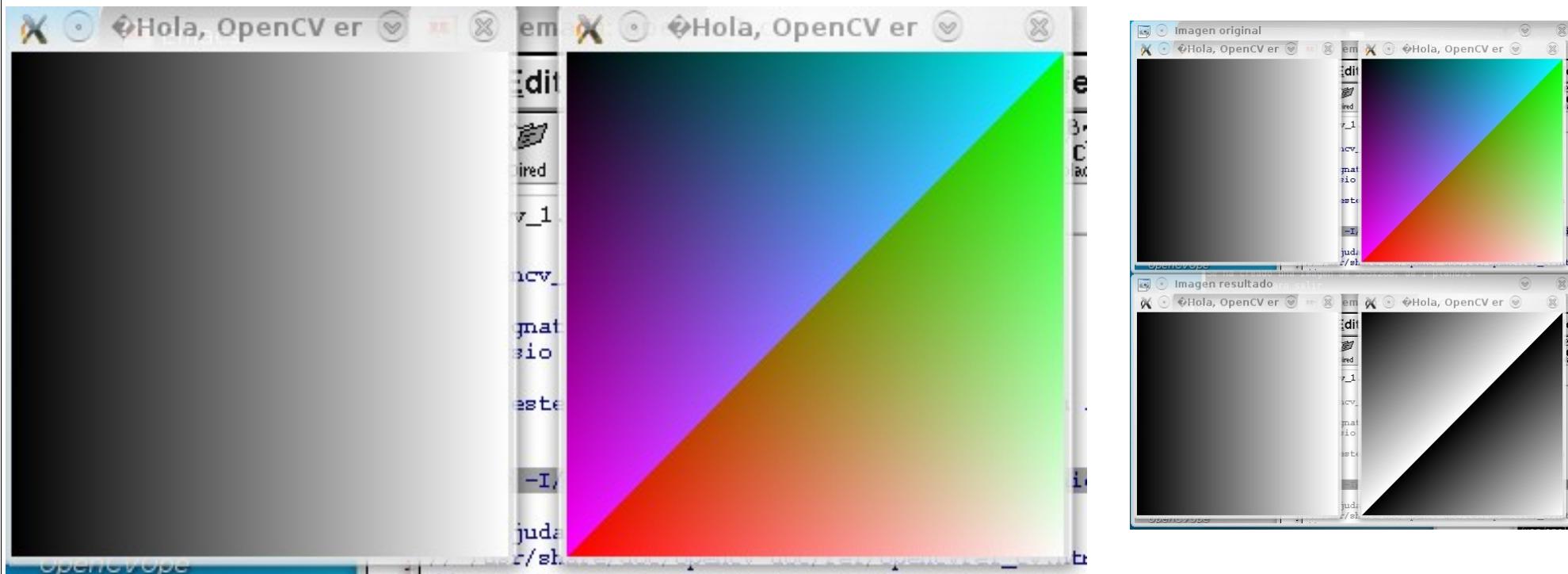
    imgDst = Mat::zeros(imgDst.rows, imgDst.cols, imgDst.depth());

    ...

    imgOrg.release();           // explícito vs implícito (recolector de C++)
    imgDst.release();          // OpenCV >= 3.0 ya no es necesario

    ...
}
```

Ej. OpenCV: acceso a las componentes (y III)



Ej. OpenCV: acceso a las componentes (II)

```
#define fRGB "¡Hola, OpenCV en color!"
```

```
...
```

```
cv::Mat imgOrgRGB,
```

```
...
```

```
imgOrgRGB.cv::Mat::create( imgOrgGris.rows, imgOrgGris.cols, cv::CV_8UC3 );
```

```
...
```

```
inicIALIZARIMGRGB( imgOrgRGB );
```

```
cv::namedWindow( fRGB, CV_WINDOW_AUTOSIZE );
```

```
cv::imshow( fRGB, imgOrgRGB );
```

```
cv::moveWindow( fRGB, AMPLE+30, 0 );
```

```
...
```

```
imgOrgRGB.release();
```

```
fRGB.release();
```



```
int inicializarImgRGB( Mat *imgOrg ) {  
    for ( x = 0; x < imgOrg->cols; x++ )  
        for ( y = 0; y < imgOrg->rows; y++ ) {  
            imgOrg->at<Vec3b>(y, x)[R] = x;  
            imgOrg->at<Vec3b>(y, x)[G] = y;  
            imgOrg->at<Vec3b>(y, x)[B] = (uchar)((y+x) % (int)L);  
        } // Fin de for ( y ...  
}
```

ó

```
imgOrg = cv::Mat::zeros(imgOrg->rows, imgOrg->cols,  
                      CV_8UC3);
```

ó

```
imgOrg = cv::Scalar( 255, 0, 0);
```

Ej. OpenCV: manejo de eventos

```
#define ESC 27

int key;

void on_mouse(int event, int x, int y, int flags, void *parametros ) {
    ...
} // on_mouse

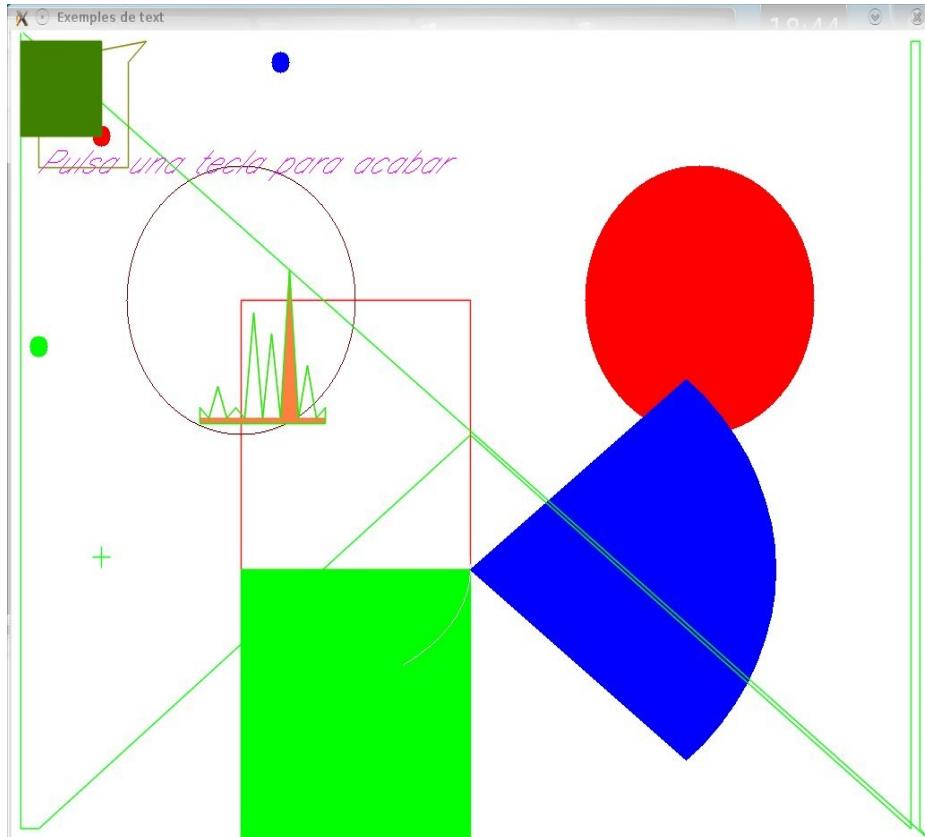
int main( ... ) {
    ...
    cv::setMouseCallBack( nombreVentana, on_mouse, NULL );
    for(;;) {
        key = cv::waitKey(0) & 255;      if ( key == ESC ) break;
    }
    ...
}

switch ( event ) {
    case EVENT_LBUTTONDOWN: // mouseUp
        ...
        break;
    } // switch
    ...
}
```

Ej. OpenCV: primitivas de dibujo

```
cv::Mat *image;  
...  
fila = (int)( image->rows / 3 );  
columna = (int)( image->cols / 4 );  
cv::line(image, cv::cvPoint(0,0), cv::cvPoint(columna, fila),  
         cv::CV_RGB(0,255,0), thickness, line_type, shift );  
  
cv::rectangle( image, cv::cvPoint(columna, fila), cv::cvPoint(2*columna, 2*fila),  
                 cv::CV_RGB(255,0,0), thickness, line_type, shift);  
  
cv::circle( image, cv::cvPoint(100, 100), 10, cv::CV_RGB(255,0,0), cv::CV_FILLED, line_type, shift );  
  
cv::ellipse( image, cv::cvPoint(2*columna, fila), axes, angle, start_angle, end_angle,  
              cv::CV_RGB(0,0,255), cv::CV_FILLED, line_type, shift );  
  
  
cv::cvInitFont( &font, cv::CV_FONT_HERSHEY_SIMPLEX, hScale,vScale, italicScale, lineWidth, 8);  
  
cv::putText( image,"Pulsa una tecla para acabar", cvPoint(columna, fila ), &font,  
             cv::CV_RGB(255,0,255));  
  
cv::imshow( nomFinestra, image );  
...
```

Ej. OpenCV: primitivas de dibujo (y II)



Ej. OpenCV: mostrar y guardar imágenes

```
char *nombreVentana;  
...  
cv::Mat::zeros(imgOrg→:cv::namedWindow( nombreVentana, cv::WINDOW_AUTOSIZE );  
                                // cv::WINDOW_NORMAL  
...  
cv::imshow( nombreVentana, imgOrg );  
printf( "Imagen de %dx%d, con %d canales", imgOrg->cols, imgOrg→rows, imgOrg->channels() );  
...  
std::vector<int> compression_params;          // ImwriteFlags : JPEG, PNG  
compression_params.push_back( IMWRITE_JPG_COMPRESSION );  
compression_params.push_back(50);  
compression_params.push_back( IMWRITE_JPEG_PROGRESSIVE );  
compression_params.push_back(1);  
compression_params.push_back( IMWRITE_JPEG_OPTIMIZE );  
compression_params.push_back(1);  
cv::imwrite( "nombreFichero.jpg", imgOrg, compression_params );  
...
```

Ej. OpenCV: imágenes y ficheros

```
cv::Mat *imgOrg, *dst[3];  
...  
imgOrg = cv::imread( rutalmagen, cv::CV_LOAD_IMAGE_ANYDEPTH);  
  
dst[0]. createImage( imgOrg.rows, imgOrg.cols, imgOrg.depth() );  
dst[1]. createImage( imgOrg.rows, imgOrg.cols, imgOrg.depth() );  
dst[2]. createImage( imgOrg.rows, imgOrg.cols, imgOrg.depth() );  
cv::split( imgOrg, dst[0], dst[1], dst[2], NULL );
```

```
std::vector<int> compression_params;  
  
compression_params.push_back( cv::IMWRITE_PNG_COMPRESSION );  
  
compression_params.push_back( 9 );  
  
cv::imwrite( nomFDst.png, dst[0], compression_params );  
  
cv::imshow( nomFinestra, imgOrg ); cv::imshow( "Un camal", dst[0]);
```

```
cv::Mat planes[] = {dst[1], dst[0], dst[2]};  
  
cv::merge(planes, 3, imgDst);  
  
cv::imshow( "Recombinada", imgDst);
```

Formatos (actualmente):

Windows bitmaps - *.bmp, *.dib
JPEG files - *.jpeg, *.jpg, *.jpe
JPEG 2000 files - *.jp2
Portable Network Graphics - *.png
WebP - *.webp
Portable image format - *.pbm, *.pgm, *.ppm
*.pxm, *.pnm
Sun rasters - *.sr, *.ras
TIFF files - *.tiff, *.tif
OpenEXR Image files - *.exr
Radiance HDR - *.hdr, *.pic
Raster and Vector geospatial data supported by
Gdal (see the Notes section)

*"In the case of color images, the decoded images
will have the channels stored in B G R order".*

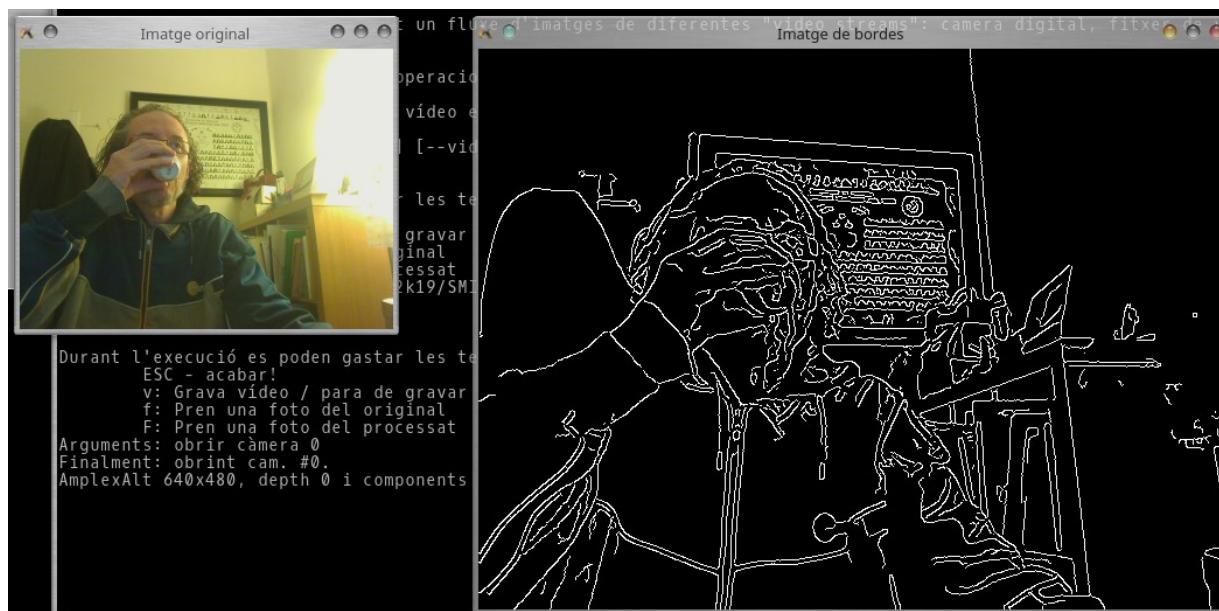
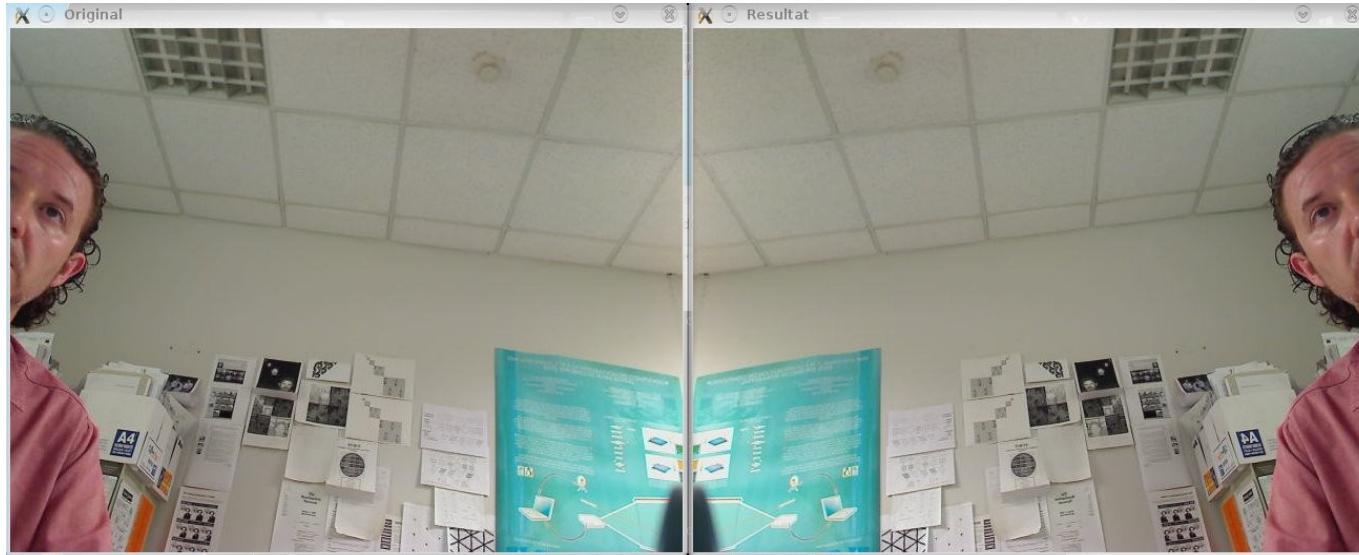
Ej. OpenCV: procesar una secuencia

```
cv::VideoCapture *laCamara;  
  
cv::Mat frame, imDst;  
  
...  
  
laCamara.open( CAP_ANY ); // Cámara por defecto ó laCamara.open( 0 )  
  
if( ! capture.isOpened() ) { printf("--(!)Error opening video capture\n"); return -1};  
  
...  
  
while ( capture.read(frame) ) { // == capture >> frame  
    if( frame.empty() ) { printf(" --(!) No captured frame – Break!"); break; }  
  
    // Procesar la imagen  
  
    cv::flip( frame, imgDst, 1); // flipCode: 0 --> filas / > 0 cols / < 0 filas y cols  
  
    //-- Show what you got  
  
    cv::imshow( window_name, frame );  
  
    cv::imshow( ventana_proceso, imgDst );  
  
    int c = cv::waitKey(10);  if( (char)c == 27 ) { break; } // escape  
  
}  
  
...
```

Alternativa 1

Ej. OpenCV: procesar una secuencia (II)

- Uso de la cámara



Ej. OpenCV: procesar una secuencia

```
cv::VideoCapture *laCamara;  
  
cv::Mat frame, imDst;  
  
...  
  
laCamara.cv::open( CAP_ANY ); // Cámara por defecto ó laCamara.open( 0 )  
  
if( ! capture.cv::isOpened() ) { printf("--(!)Error opening video capture\n"); return -1};  
  
while ( ... ) {  
  
    // cap >> frame; // get a new frame from camera → "graba i decodifica == grab + retrieve"  
  
    if( !cap.cv::grab() ) { fiDeVideo = true; }                                // Congela la imagen en la cámara  
    else {  
  
        cap.cv::retrieve( frame, 0 ); // Decodes and returns the grabbed video frame.          // Decodifica  
  
        // Procesar  
  
        cv::cvtColor(frame, edges, cv:: CV_BGR2GRAY );  
  
        cv::GaussianBlur(edges, edges, Size(7,7), 1.5, 1.5);  
  
        cv::Canny(edges, edges, 0, 30, 3);  
  
        ... // Mostrar  
  
    }  
}.
```

Alternativa 2

// Congela la imagen en la cámara

// Decodifica

Ej. OpenCV: procesar una secuencia (y IV)

- Desde vídeo

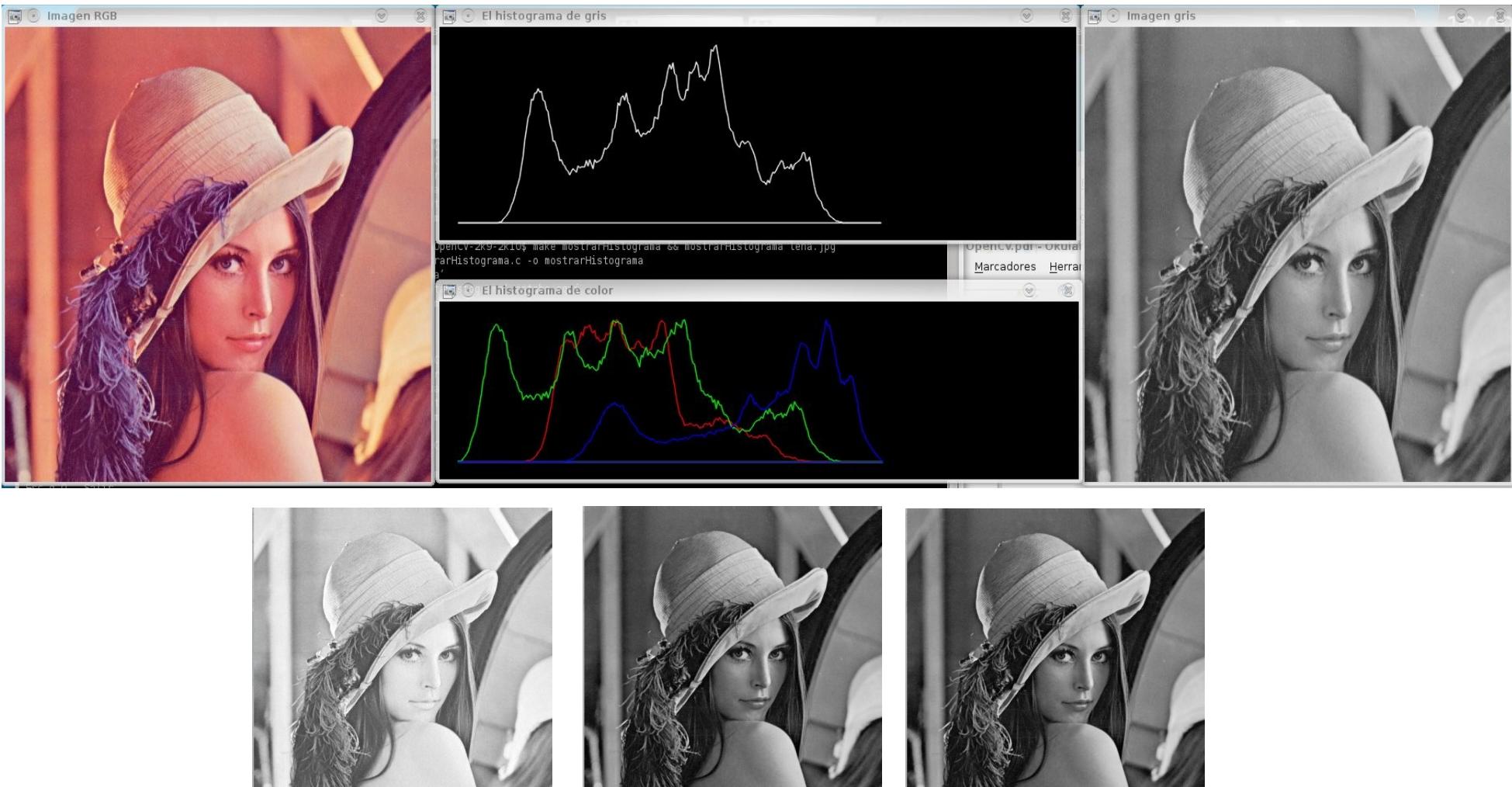
```
cv::VideoCapture *laCamara;  
  
laCamara.open( CAP_ANY ); // Cámara por defecto ó laCamara.open( 0 )  
  
if (!laCamara.isOpened()) {  
  
    std::cout << "Could not open the input video: " << source << std::endl; return -1; }  
  
...  
  
outputVideo.open(NAME, ex=-1, laCamara.get(cv::CAP_PROP_FPS), S, true);  
  
if (!outputVideo.isOpened()) {  
  
    std::cout << "Could not open the output video for write: " << source << std::endl; return -1; }  
  
...  
  
cv::Mat src, res;  
  
for( ... ) {  
  
    laCamara >> src;          // ó laCamara.read( src )  
  
    // Procesar src → res  
  
    outputVideo.write(res);    // ó save ó outputVideo << res;  
  
}
```

Ej. OpenCV: histogramas (I)

```
void calcularHistogramaGris( cv::Mat src ) {  
    int histSize = 256; float range[] = { 0, 256 } ; const float* histRange = { range };  
    bool uniform = true; bool accumulate = false; cv::Mat histImage;  
    cv::calcHist( &src, 1, 0, Mat(), hist, 1, &histSize, &histRange, uniform, accumulate );  
}  
  
void pintarHistogramaGris( cv::Mat hist, int histSize, cv::Mat histImage ) {  
    int hist_w = 512; int hist_h = 256; int bin_w = cvRound( (double) hist_w/histSize );  
    histImage.cv::Mat::create( hist_h, hist_w, CV_8UC1); histImage = Scalar( 0,0,0);  
    cv::normalize(hist, hist, 0, histImage.rows, NORM_MINMAX, -1, Mat() );  
    for( int i = 1; i < histSize; i++ ) {  
        cv::line( histImage, cv::Point( bin_w*(i-1), hist_h - cvRound(hist.at<float>(i-1)) ),  
                 cv::Point( bin_w*(i), hist_h - cvRound(hist.at<float>(i)) ), cv::Scalar( 255, 255, 255), 2, 8, 0 );  
    }  
    cv::namedWindow("Histograma de grises", WINDOW_AUTOSIZE );  
    cv::imshow("Histograma de grises", histImage );  
}
```

Ej. OpenCV: histogramas (y II)

- Canales



```
cv::split(imgOrg, dst[0], dst[1], dst[2], NULL);
```

```
cv::merge(dst[0], dst[1], dst[2], ImgDst)
```

Ej. OpenCV: procesamientos puntuales

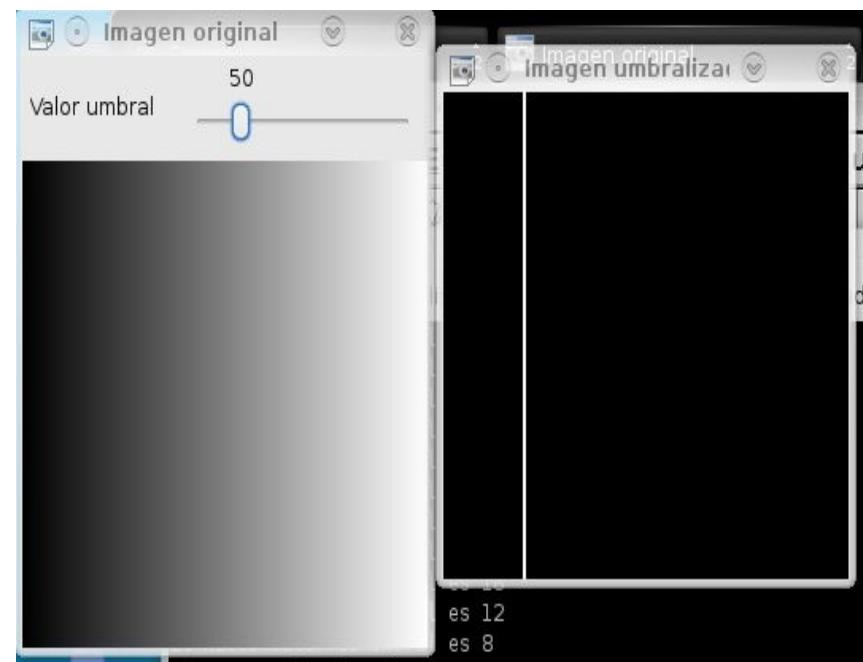
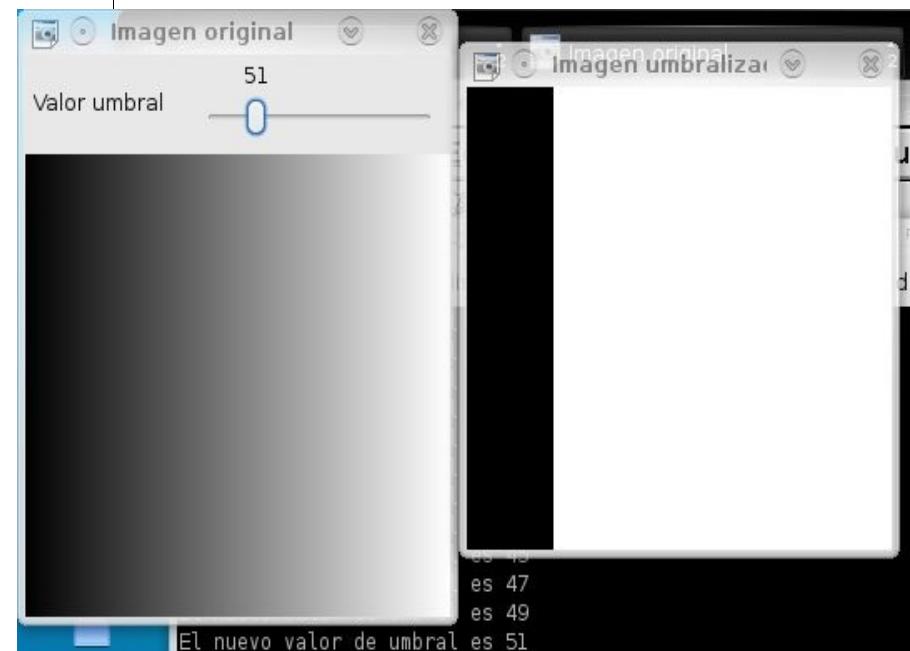
- Negativo



```
void Op_NEGATIVO( cv::Mat *img ) {  
    ...  
    for ( x = 0; x < img->cols; x++ ) {  
        for ( y = 0; y < img->rows; y++ ) {  
            img->at<Vec3b>(y, x)[R] = L - img->at<Vec3b>(y, x)[R];  
            img->at<Vec3b>(y, x)[G] = L - img->at<Vec3b>(y, x)[G];  
            img->at<Vec3b>(y, x)[B] = L - img->at<Vec3b>(y, x)[B];  
        }  
    }  
    ...
```

Ej. OpenCV: procesamientos puntuales

- Imágenes binarias: umbralización



```
void Op_UMBRALIZAR( cv::Mat *imgOrg, cv::Mat *imgDst) {  
    for ( x = 0; x < imgOrg->cols; x++ ) {  
        for ( y = 0; y < imgOrg->rows; y++ ) {  
            if (imgOrg->at<Vec3b>(y, x) >= valor)  
                imgDst->at<Vec3b>(y, x)[R] = L - img->at<Vec3b>(y, x)[R];  
            else  
                imgDst->at<Vec3b>(y, x)[G] = L - img->at<Vec3b>(y, x)[G];  
        }  
    } ...
```

```
if (imgOrg->at<Vec3b>(y, x) == valor)  
  
void Op_UMBRALIZAR( cv::Mat *imgOrg, int valor,  
                     cv::Mat *imgDst) {  
    cv::threshold( imgOrg, imgDst, valor, 255, THRESH_BINARY );  
}
```

Ej. OpenCV: procesos de vecindad

- Convolución

```
void Op_FILTRO3x3( cv::Mat* imgOrg, cv::Mat *imgDst )  
  
cv::Mat *src, *filtro;  
  
float kernel[3][3]={0, 0, 0, 0, 1, 0, 0, 0, 0};  
  
...  
  
src.create( cvSize(imgOrg->cols,imgOrg->rows), CV_8UC1 );  
  
cv::cvtColor( imgOrg, src, CV_BGR2GRAY );  
  
...  
  
imgDst.create( cvSize(imgOrg->cols,imgOrg->rows), CV_8UC1 );  
  
...  
  
filtro = cv::Mat::create( nFiles, nCols, CV_32FC1 );  
  
cvSetData( filtro, kernel, filtro->step); ??????  
  
cv::filter2D( src, imgDst, filtro, cvPoint(-1,-1) );  
  
...  
  
} // Op_FILTRO3x3
```

Ej. OpenCV: procesos de vecindad (II)

- Combinaciones de máscaras: Sobel

```
void sobel( cv::Mat *imgOrg, cv::Mat *imgDst )  
{  
    cv::Mat *df_dx, *df_dy, *dest_dx, *dest_dy, *src;  
    ...  
    src = cvCreateImage( cv::cvSize(imgOrg->cols,imgOrg->rows), CV_8UC1 );  
    cvCvtColor( imgOrg, src, CV_BGR2GRAY );  
  
    //create temp images  
    df_dx=cv::create(cvGetSize(src), CV_16SC1);  
    df_dy. cv::create(cvGetSize(src),CV_16SC1);  
  
    // use sobel to find derivatives  
    cv::cvSobel( src, df_dx, 1, 0, 3);  
    cv::cvSobel( src, df_dy, 0, 1, 3);
```

Ej. OpenCV: procesos de vecindad (III)

- Combinaciones de máscaras: Sobel

```
void sobel( IplImage* imgOrg, IplImage *imgDst )
```

```
{
```

```
...
```

[Continuación]

```
//Convert signed to unsigned 8
```

```
cv::convertScaleAbs( df_dx , dest_dx, 1, 0);
```

```
cv::convertScaleAbs( df_dy , dest_dy, 1, 0);
```

```
...
```

```
ImgDst->cv::Mat;create( c::cvGetSize(src), IPL_DEPTH_8U, 1 );
```

```
cv::add( dest_dx, dest_dy, imgDst, NULL );
```

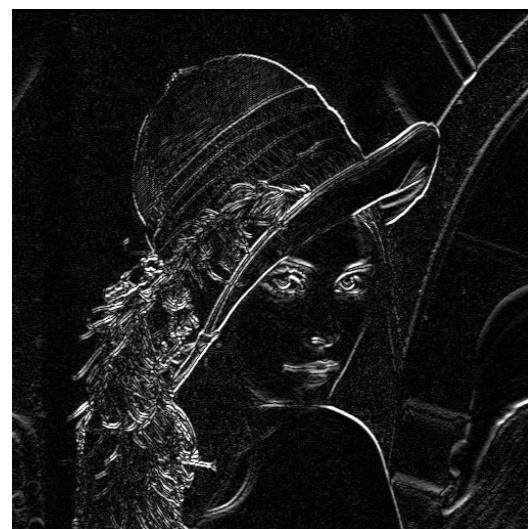
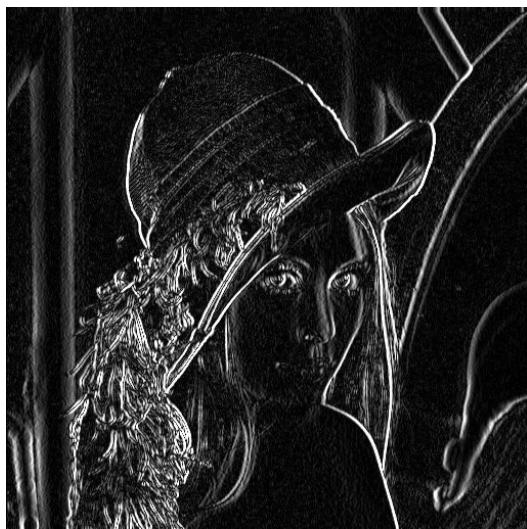
```
...
```

```
...
```

```
} // Fi de sobel
```

Ej. OpenCV: procesos de vecindad (y IV)

- Convolución
- Ejemplos de uso de una máscara
- Combinaciones de máscaras: Sobel



Índice

1. Introducción a la Visión por Computador
2. Uso de OpenCV: operaciones básicas
3. Aplicaciones: ejemplos de integración de medios
 1. Interacción por eventos visuales
 1. Definición.
 1. Detección del puntero láser
 2. Salvapantallas activo
 2. Detección de objetos por color
 1. Segmentación básica. *Umbrales*
 2. Segmentación robusta. *Camshift*
 3. Detectores de forma. *Square*
 4. Detector de objetos a partir de modelos. *Facedetect*
 5. Detección de movimiento. *Motion template*
 6. *Detección de objetos a partir de patrones*
 1. De mapas de bits. *Template matching*
 2. De características. *find_object*

Interacción por eventos visuales

- “*Activo: Que obra o tiene virtud de obrar*”.
- *Generar respuestas por partes del computador en función de lo que sucede en su entorno de trabajo exterior*
 - *Objetivo*
 - *Reaccionar a estímulos, tener actividad en función de lo que pasa delante del computador*
 - *Tipos de respuesta*
 - *Ejecutar aplicaciones*
 - *Generar respuestas ← imágenes y/o sonidos*
- *Interacción Persona-Computador (IPO, HCI)*

Interacción por eventos visuales

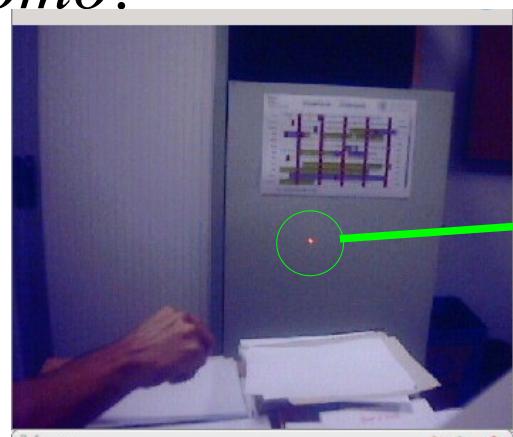
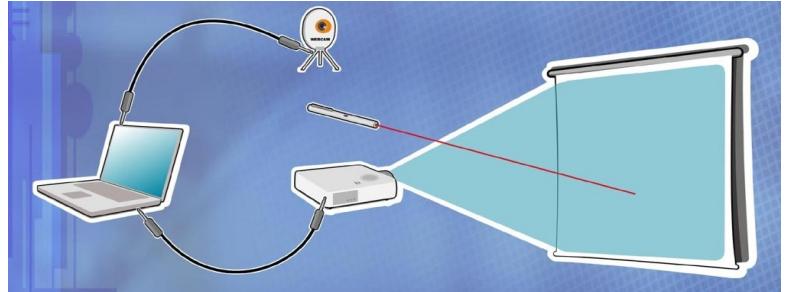
- *Ejemplo: Detección del puntero láser*

- *David (2K4)*

- *Idea*

- *Las acciones del cursor en pantalla deben corresponder con acciones externas al escritorio tradicional*
 - *El punto de luz láser que se utiliza sobre un proyector, ...*

- *¿Cómo?*



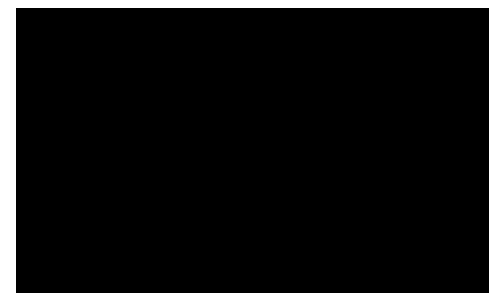
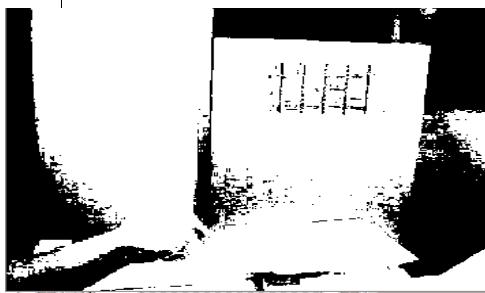
$\dot{x}(x,y)_{\text{Pantalla}}$?

Interacción por eventos visuales

- *Algoritmo*
 - *Estudiar comportamiento color*



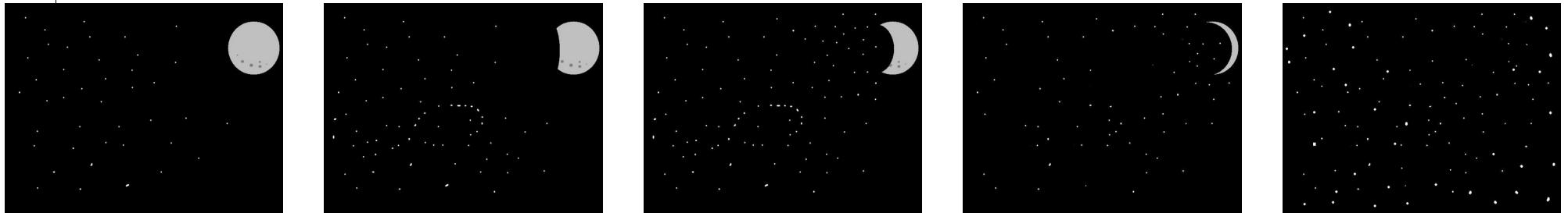
- *Segmentar objeto vs fondo*



- *Establecer coordenadas en la imagen → CdG*
- *Generar evento, en coordenadas de pantalla, en el sistema → dependiente del SO*

Interacción por eventos visuales

- *Ejemplo: “Salvapantallas activo”*
- *Algoritmo*
 - *Para cada imagen*
$$RGB_{media} = (RGB_{pixel1} + RGB_{pixel2} + \dots + RGB_{pixeln})/n.$$
 - *¿Día/Noche? → cambiar de imágenes*



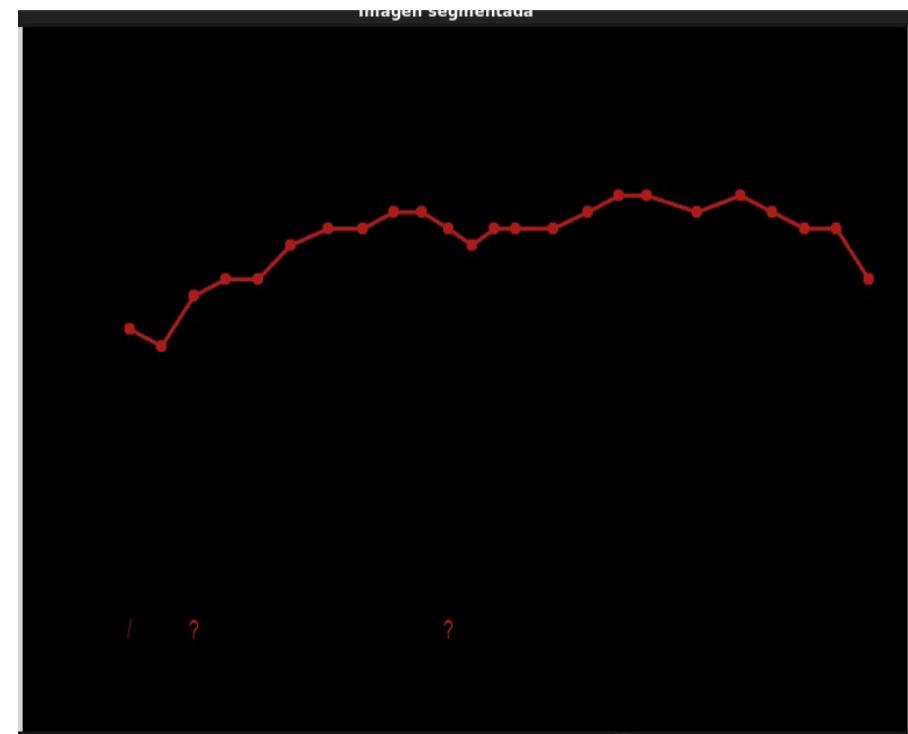
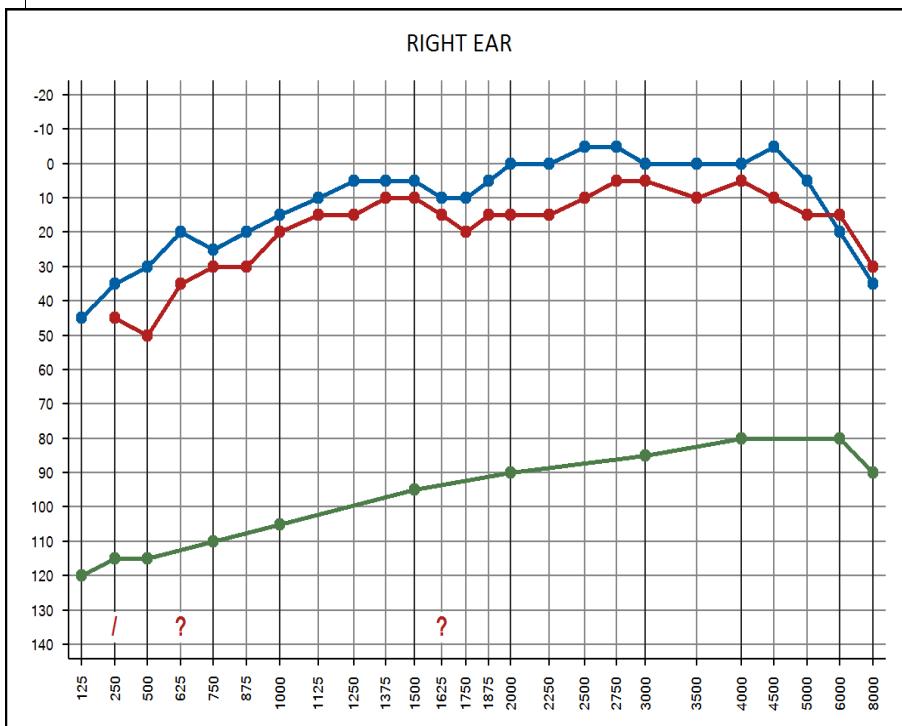
- *Sonido de fondo → volumen controlado con “Square Detector” (squares.cpp)*

Índice

1. Introducción a la Visión por Computador
2. Uso de OpenCV: operaciones básicas
3. Aplicaciones: ejemplos de integración de medios
 1. Interacción por eventos visuales
 1. Definición.
 1. Detección del puntero láser
 2. Salvapantallas activo
 2. Detección de objetos por color
 1. *Segmentación básica. Umbrales*
 2. *Segmentación robusta. Camshift*
 3. *Interpretación de gestos. Pintando con el dedo*
 3. Detectores de forma. square, circle
 4. Detector de objetos a partir de modelos. *Facedetect*
 5. Detección de movimiento. *Motion template*
 6. *Detección de objetos a partir de patrones*
 1. De características. *find_object*
 2. De texturas. *Template matching*

Detección de objetos por color

- *Segmentación básica por selección de color*
 - Umbrales para la selección de color: *InRange*
 - *cvInRangeS(imgOrg, colorMin, colorMax, mascara);*
 - *cvAnd(imgOrg, imgOrg, imgDst, mascara);*



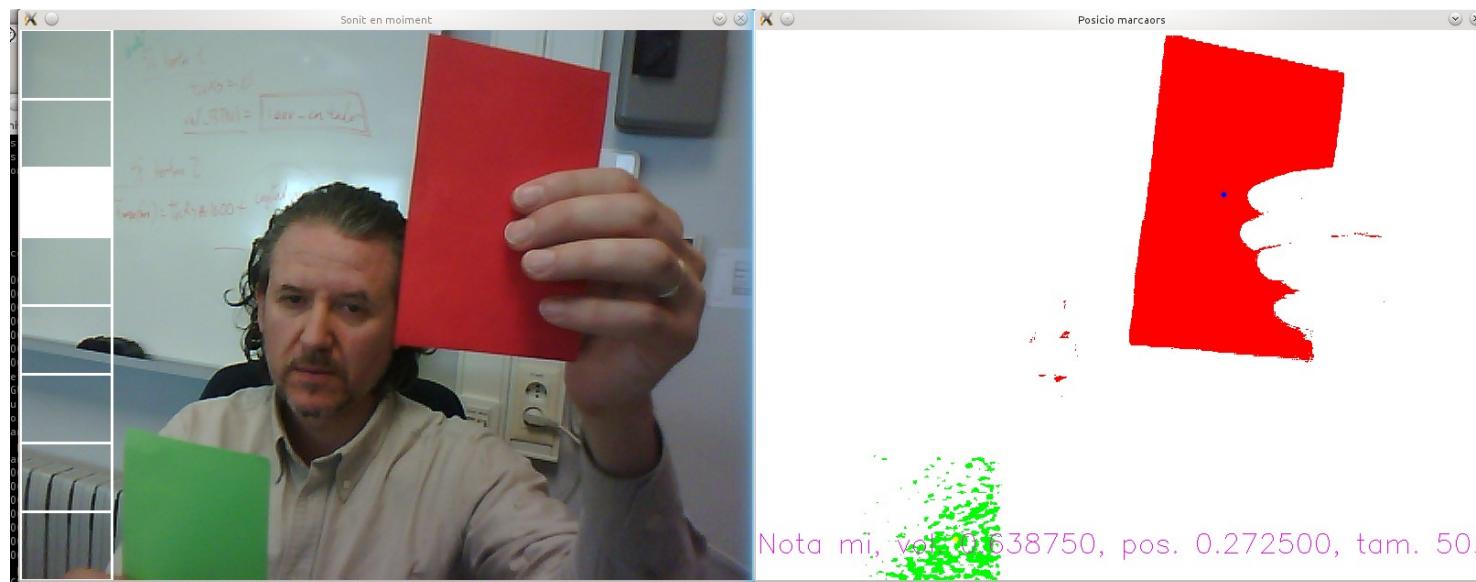
Detección de objetos por color (II)

- *Segmentación básica por selección de color*
 - Umbrales para la selección de color: *InRange*
 - *cvInRangeS(imgOrg, colorMin, colorMax, mascara);*
 - *cvAnd(imgOrg, imgOrg, imgDst, mascara);*



Detección de objetos por color (III)

- *Interacción por color con OpenCV y OpenAL*
 - Umbrales para la selección de color: *InRange*
 - *cvInRangeS(imgOrg, colorMin, colorMax, mascara);*
 - *cvAnd(imgOrg, imgOrg, imgDst, mascara);*



Detección de objetos por color (IV)

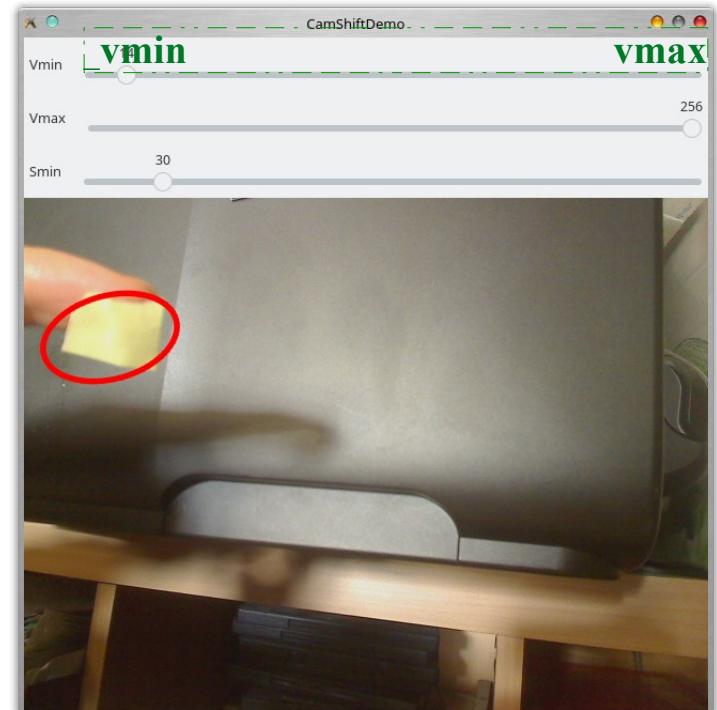
- *Segmentación robusta*

- Ej. de OpenCV: *camshiftdemo* (*examples/cpp/camshift.cpp*)

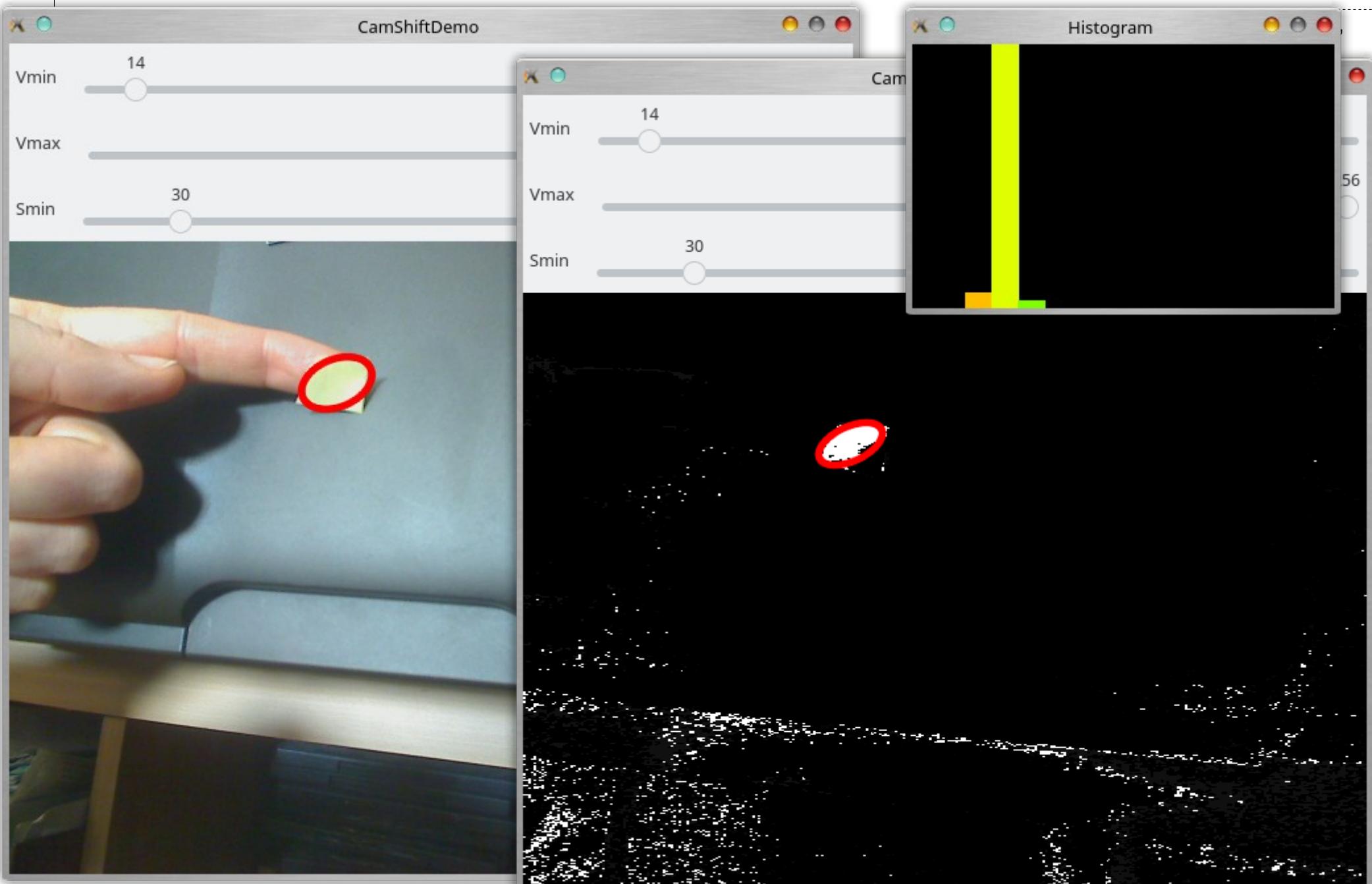
```
./camshiftdemo
Hot keys:
    ESC - quit the program
    c - stop the tracking
    b - switch to/from backprojection view
    h - show/hide object histogram
To initialize tracking, select the object with mouse
```

- *Algoritmo:*

1. Seleccionar el área de interés. → color
2. Para cada imagen RGB obtenida de la cámara
 1. Convertir a HSV.
 2. Segmentar la componente Hue con los parámetros Vmin, Vmax, Smin.
 3. Etiquetar los puntos que están alrededor del color indicado por el usuario.
 4. Calcular X, Y (centro de masas), Z (tamaño del área) y ángulo del eje mayor con la horizontal.
 5. Dibujar un ellipse centrada en X, Y con diámetro el ancho del área de puntos de ese color.
3. fPara



Detección de objetos por color (V)



Detección de objetos por color (VI)

- *camshiftdemo.cpp*

```
...
frame = cvQueryFrame( capture );
...
cvCopy( frame, image, 0 );
cvCvtColor( image, hsv, CV_BGR2HSV );
...
cvInRangeS( hsv, cvScalar(0,smin,MIN(_vmin,_vmax),0),           ←
            cvScalar(180,256,MAX(_vmin,_vmax),0), mask );
cvSplit( hsv, hue, 0, 0, 0 );
...
cvCalcHist( &hue, hist, 0, mask );
cvGetMinMaxHistValue( hist, 0, &max_val, 0, 0 );
...
cvCalcBackProject( &hue, backproject, hist );
cvAnd( backproject, mask, backproject, 0 );

// The CAMSHIFT object tracking algorithm [Bradski98]: finds an object center, size, and orientation.
cvCamShift( backproject, track_window,
            cvTermCriteria( CV_TERMCRIT_EPS | CV_TERMCRIT_ITER, 10, 1 ),
            &track_comp, &track_box );
...
cvEllipseBox( image, track_box, CV_RGB(255,0,0), 3, CV_AA, 0 );
```

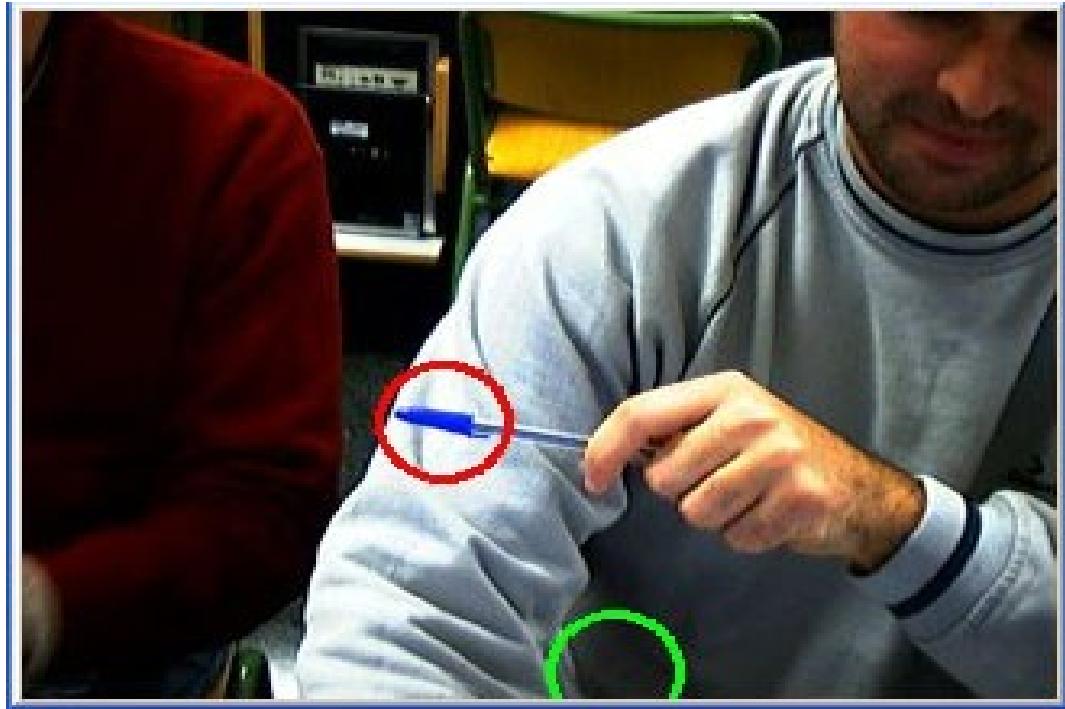
```
void on_mouse( int event, int x, int y,
int flags, void* param )
{
    ...
    switch( event )
    {
        case CV_EVENT_LBUTTONDOWN:
            ...
            select_object = 1;
            break;
        case CV_EVENT_LBUTTONUP:
            select_object = 0;
            ...
    }
}
```

Detección de objetos por color (y VII)

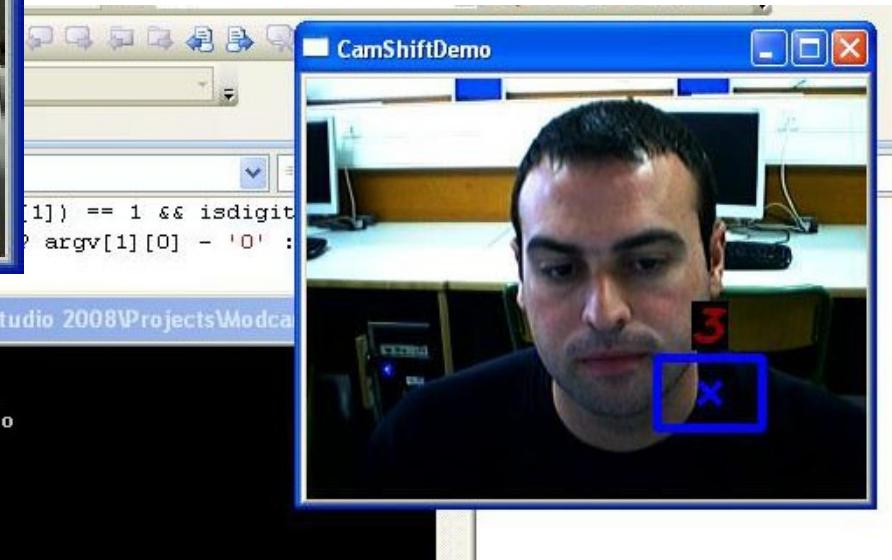
- *Interacción por color*

- ¡Atrápalo!

$$d(X, Y) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$



```
c:\Documents and Settings\joargu\Mis documentos\Visual Studio 2008\Projects\Modca>
Hot keys:
    ESC - Salir del programa
    c - Para de rastrear el objeto
    p - Para volver a rastrear un nuevo objeto
    b - Cambiar a modo backprojection
```



Fuente: Trabajo de Jose (2K10) y

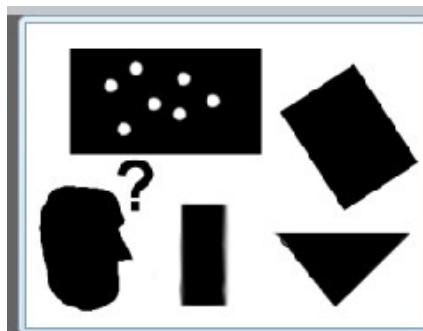
M. Agustí y J. A. (2011). Interacción con OpenCV: Seguimiento de un objeto por color <<https://riunet.upv.es/handle/10251/12682>>

Detectores de forma

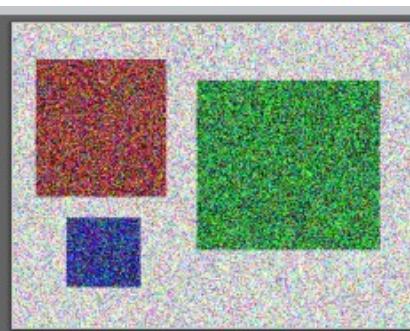
- *Formas básicas: primitivas geométricas*
 - Detector de líneas
 - De esquinas
 - De cuadrados
 - De círculos
- *Formas complejas:*
 - Búsqueda de patrones
 - Búsqueda de códigos 1D y 2D
 - Búsqueda de descriptores

Detectores de forma

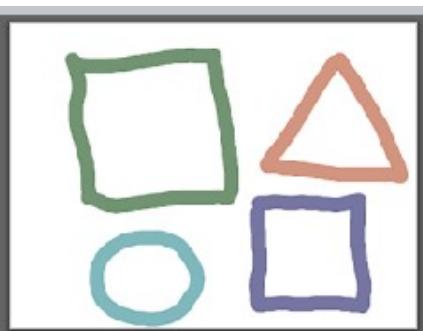
- *Formas básicas: primitivas geométricas*
 - Detector de líneas
 - De cuadrados
 - De círculos
 - De esquinas



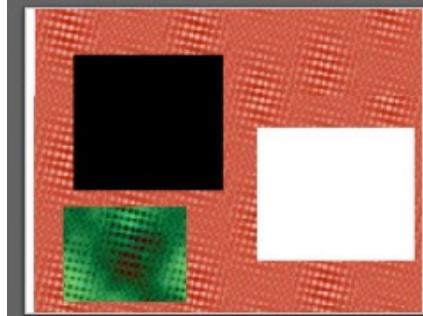
pic1.png



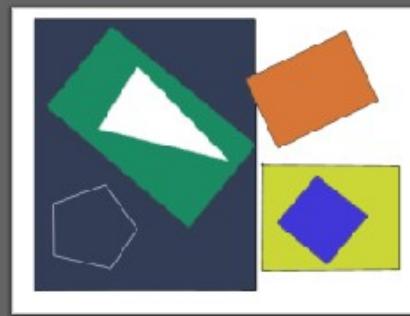
pic2.png



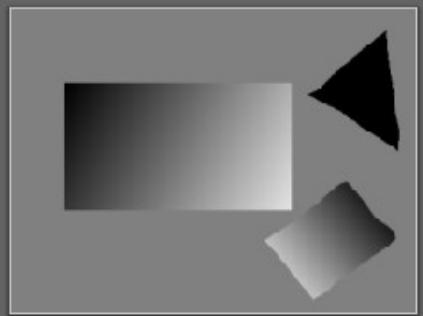
pic3.png



pic4.png



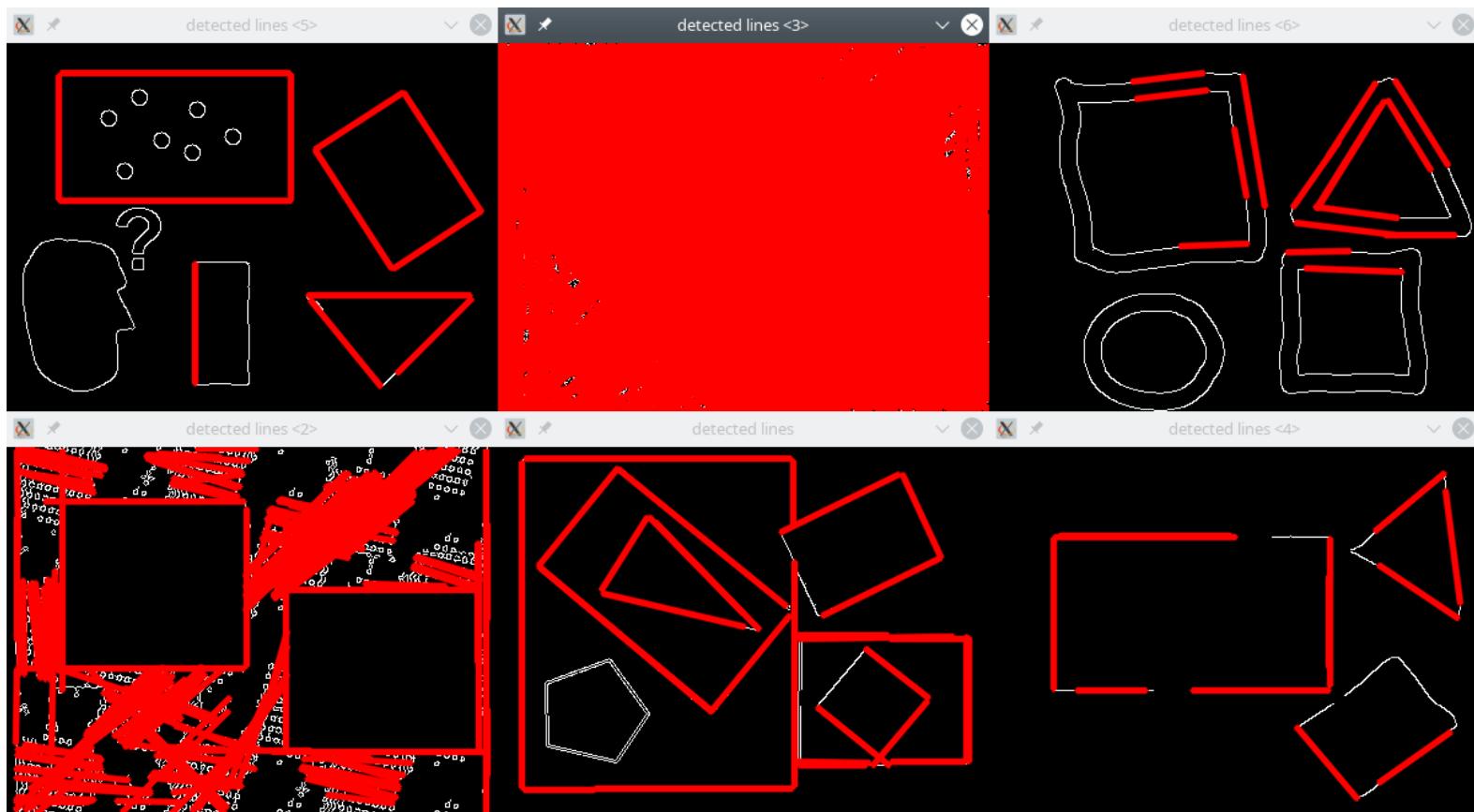
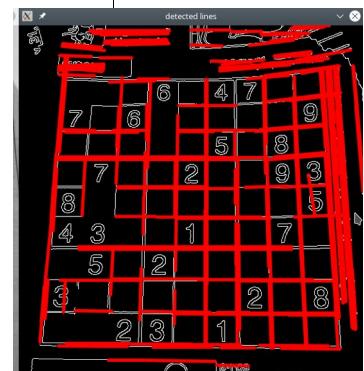
pic5.png



pic6.png

Detectores de forma

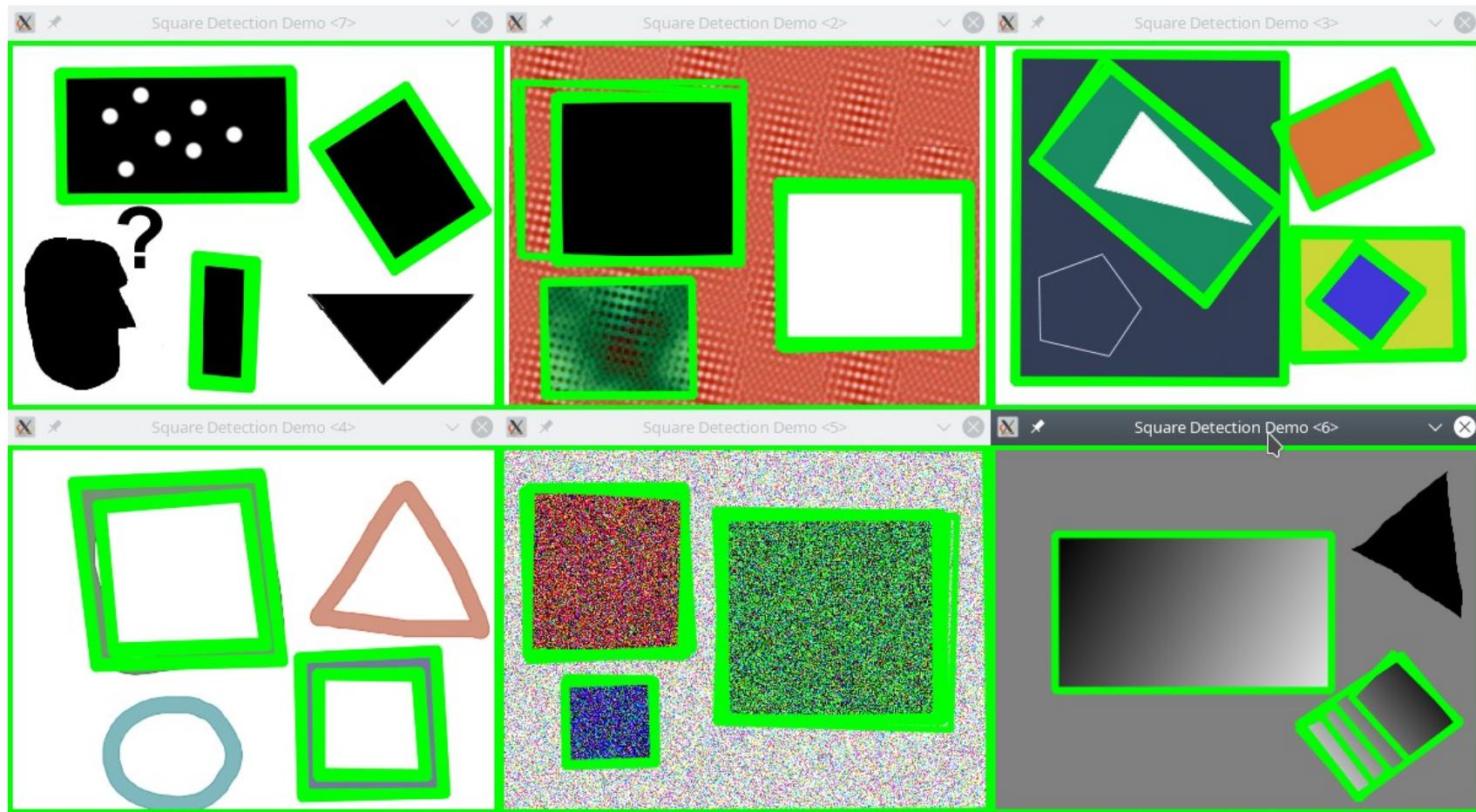
- *Formas básicas: primitivas geométricas*
 - Detector de líneas



- Ejemplo
 - houghlines ..//data/pic1.png

Detectores de forma

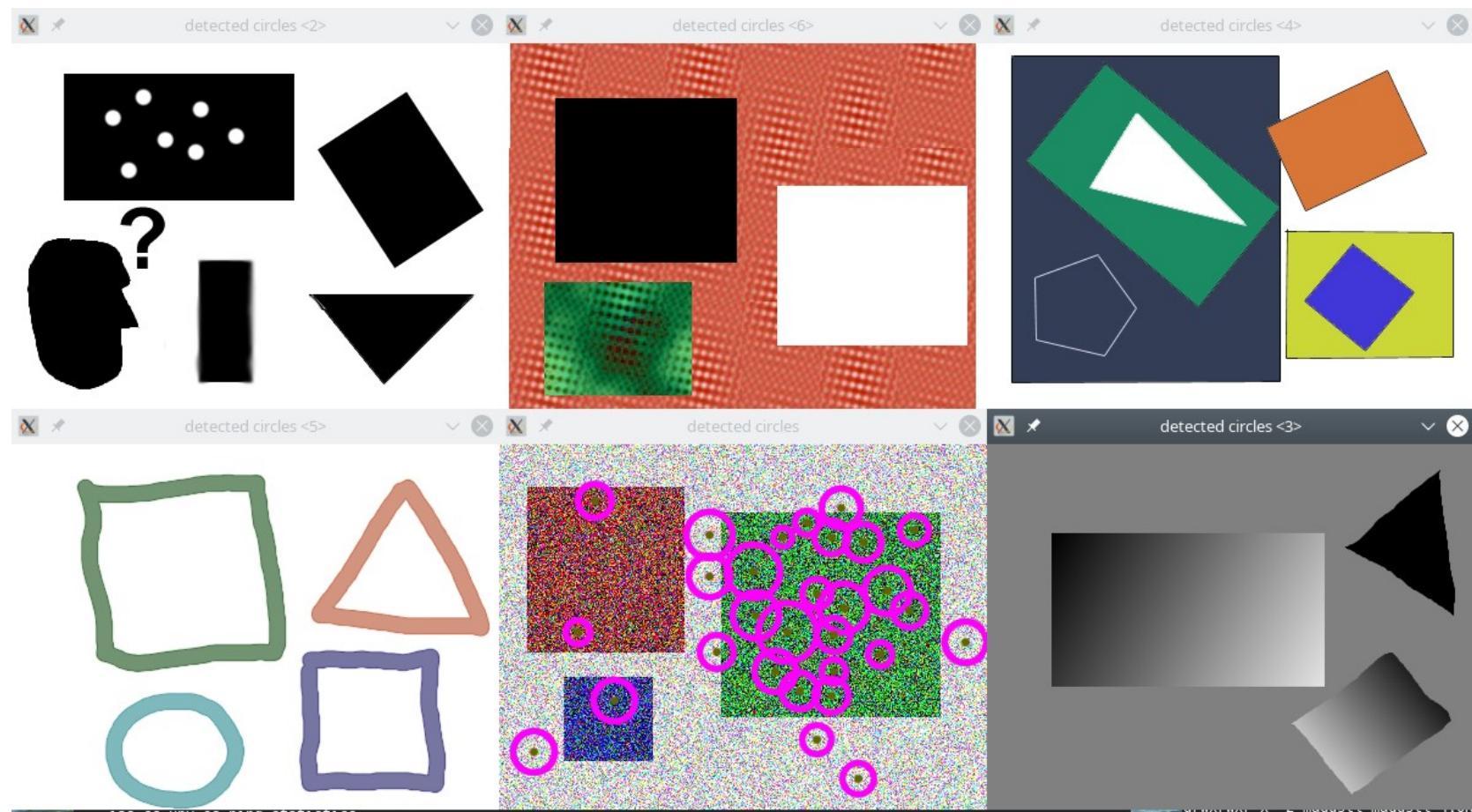
- *Formas básicas: primitivas geométricas*
 - De cuadrados → square.cpp



- Ejemplo
 - squares/data/pic1.png

Detectores de forma

- *Formas básicas: primitivas geométricas*
 - De círculos: [HoughCircles](#)



- Ejemplo <https://docs.opencv.org/4.2.0/d1/de6/samples_2cpp_2tutorial_code_2ImgTrans_2houghcircles_8cpp-example.html>
 - houghcircles ../data/pic1.png

Detectores de forma

- *Formas básicas: primitivas geométricas*
 - De esquinas: Harris

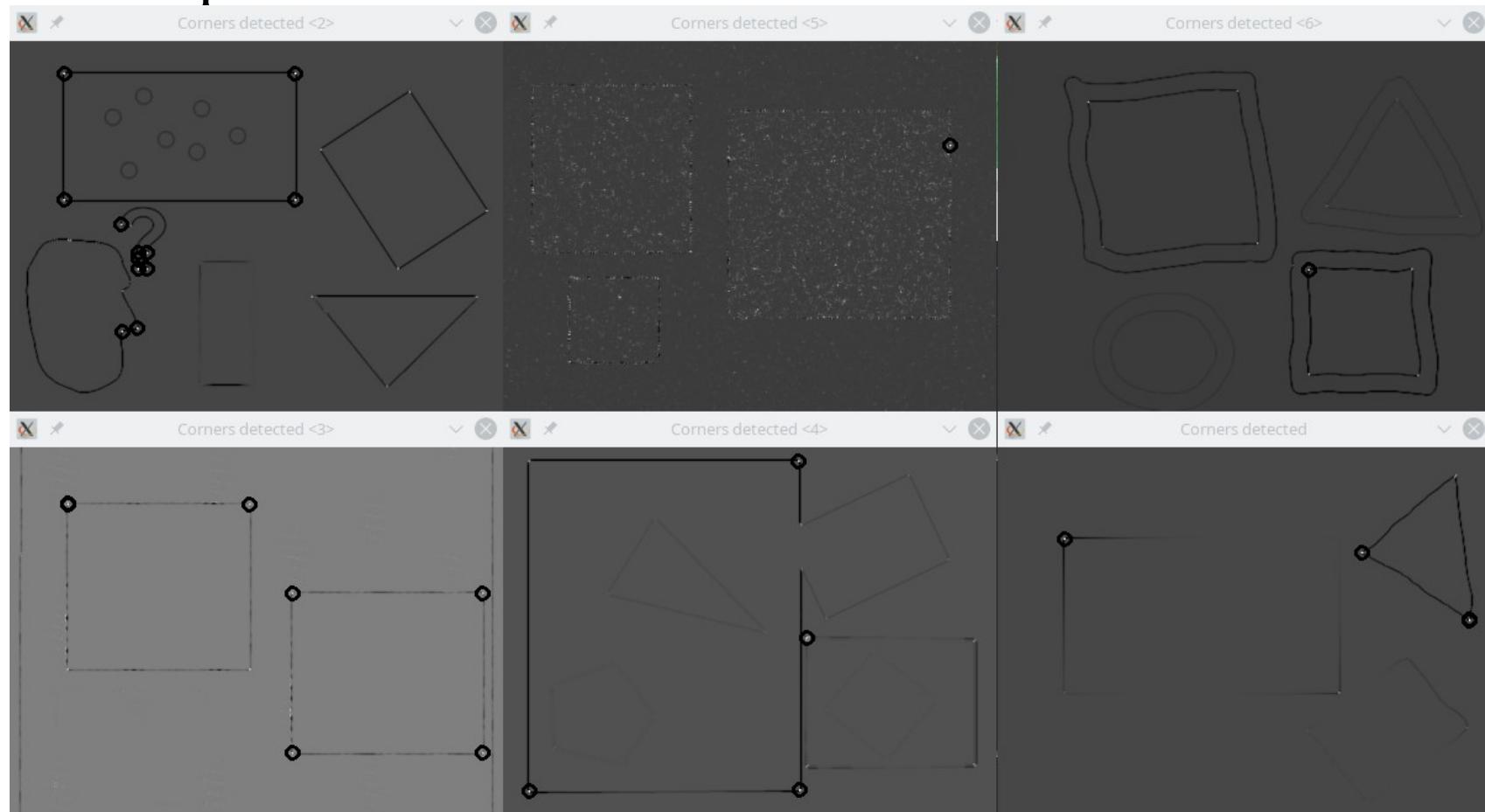


- Ejemplo

- cornersHarris_Demo building.jpg

Detectores de forma

- *Formas básicas: primitivas geométricas*
 - De esquinas: **Harris**



- Ejemplo
 - cornersHarris_Demo building.jpg

Detección de objetos por modelo

- *Detector de caras*
 - *Síntesis / Aprendizaje → Modelo*
 - *Multiresolución → haarcascade*
 - <http://docs.opencv.org/trunk/doc/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html>
 - *facedetect*
`--cascade="data/haarcascades/haarcascade_frontalface_alt.xml" --nested-cascade="data/haarcascades/haarcascade_eye.xml" --scale=1.3 lena.jpg`

Detección de objetos por modelo (II)

- *Ejemplos de modelos*

- Cara:
 - haarcascade_frontalface_default.xml,*
 - haarcascade_frontalface_alt.xml,*
 - haarcascade_frontalface_alt2.xml,*
 - haarcascade_frontalface_alt_tree.xml*
 - haarcascade_profileface.xml*
- Ojos:
 - haarcascade_eye.xml,*
 - haarcascade_righteye_2splits.xml,*
 - haarcascade_lefteye_2splits.xml,*
 - haarcascade_eye_tree_eyeglasses.xml,*
 - haarcascade_mcs_righteye.xml*
 - haarcascade_mcs_lefteye.xml*
 - haarcascade_mcs_eyepair_small.xml,*
 - haarcascade_mcs_eyepair_big.xml.*
- Oídos: *haarcascade_mcs_rightear.xml o haarcascade_mcs_leftear.xml.*
- Boca: *haarcascade_mcs_mouth.xml.*
- Nariz: *haarcascade_mcs_nose.xml*

Detección de objetos por modelo (III)

- *Ejemplo de reconocimiento de cara (1^{er} nivel)*

...
CascadeClassifier cascade, nestedCascade;

cascade.load(cascadeName)

...
capture >> frame;

if(frame.empty()) break;

Mat frame1 = frame.clone();

detectAndDraw(frame1, cascade, nestedCascade, scale, tryflip);



void detectAndDraw(Mat& img, CascadeClassifier& cascade,

 CascadeClassifier& nestedCascade, double scale, bool tryflip) {

 cvtColor(img, gray, COLOR_BGR2GRAY);

 double fx = 1 / scale; resize(gray, smallImg, Size(), fx, fx, INTER_LINEAR); equalizeHist(smallImg, smallImg);

 cascade.detectMultiScale(smallImg, faces, 1.1, 2, 0|CASCADE_SCALE_IMAGE, Size(30, 30));

...

 for (size_t i = 0; i < faces.size(); i++) {

 double aspect_ratio = (double)r.width/r.height;

 if(0.75 < aspect_ratio && aspect_ratio < 1.3) {

 center.x = cvRound((r.x + r.width*0.5)*scale); center.y = cvRound((r.y + r.height*0.5)*

 radius = cvRound((r.width + r.height)*0.25*scale);

 circle(img, center, radius, color, 3, 8, 0);

 }

else

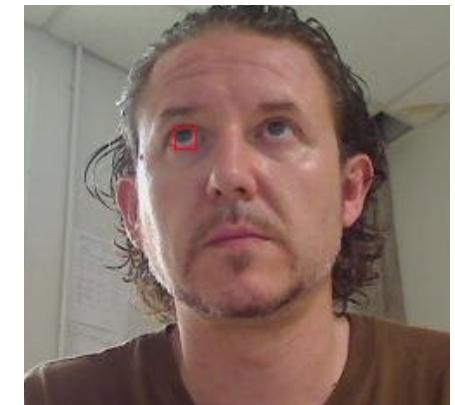
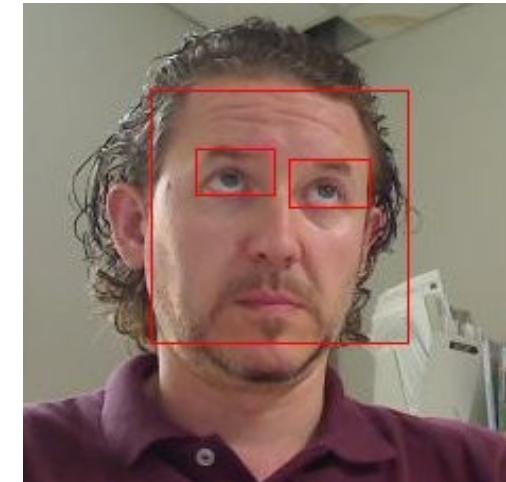


Detección de objetos por modelo (IV)

- *Ej. de reconocimiento de partes de la cara (2º nivel)*

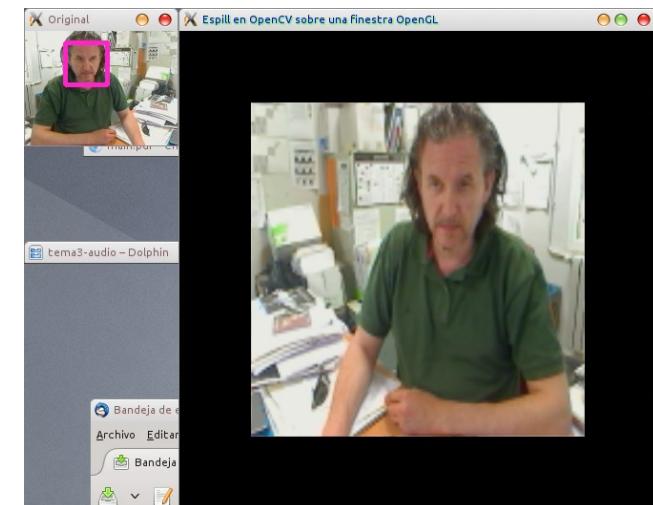
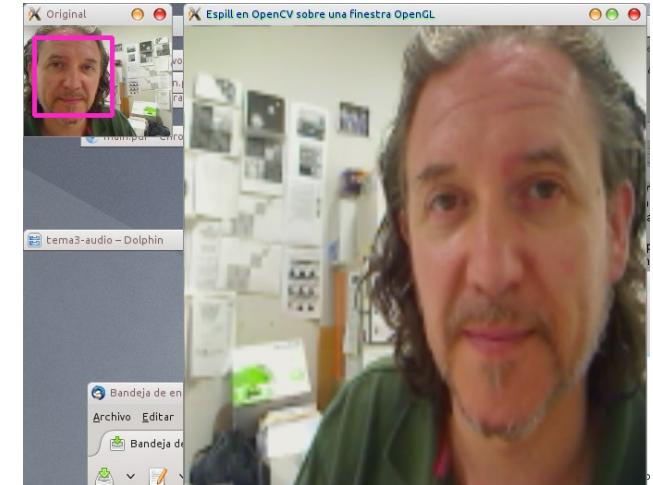
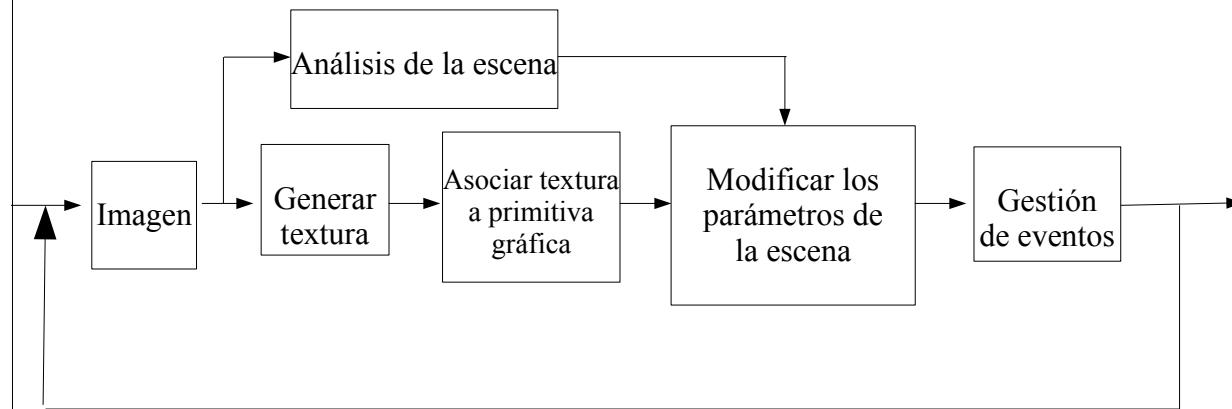
...

```
smallImgROI = smallImg( r );
nestedCascade.detectMultiScale( smallImgROI, nestedObjects,
    1.1, 2, 0 |CASCADE_SCALE_IMAGE, Size(30, 30) );
for ( size_t j = 0; j < nestedObjects.size(); j++ )
{
    Rect nr = nestedObjects[j];
    center.x = cvRound((r.x + nr.x + nr.width*0.5)*scale);
    center.y = cvRound((r.y + nr.y + nr.height*0.5)*scale);
    radius = cvRound((nr.width + nr.height)*0.25*scale);
    circle( img, center, radius, color, 3, 8, 0 );
}
```



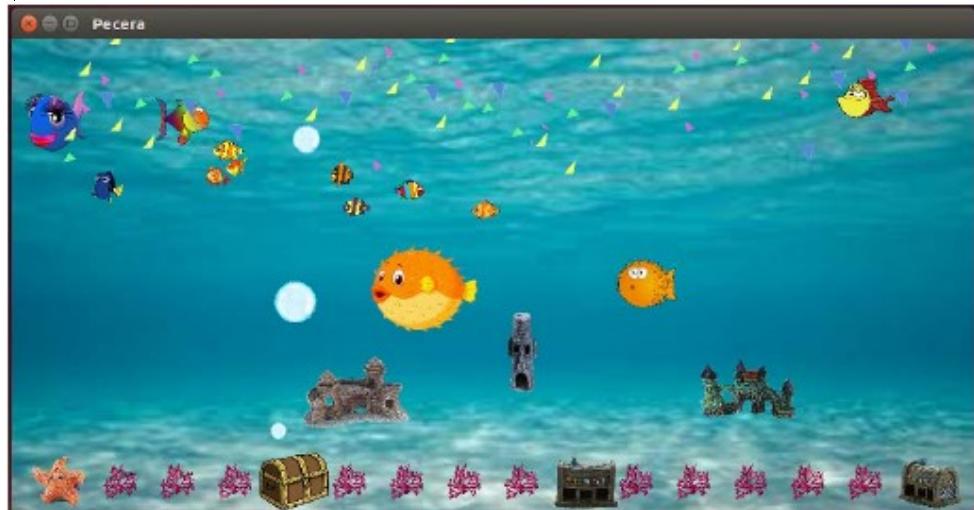
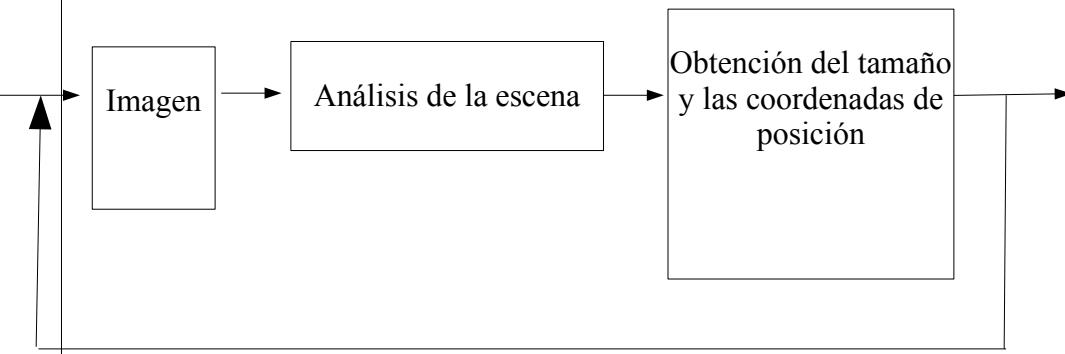
Ejemplo de interacción

- Interpretación de la distancia del usuario



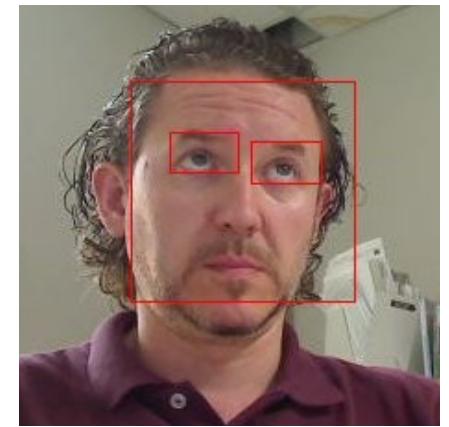
Ejemplo de interacción (y II)

- Interpretación de la distancia del usuario



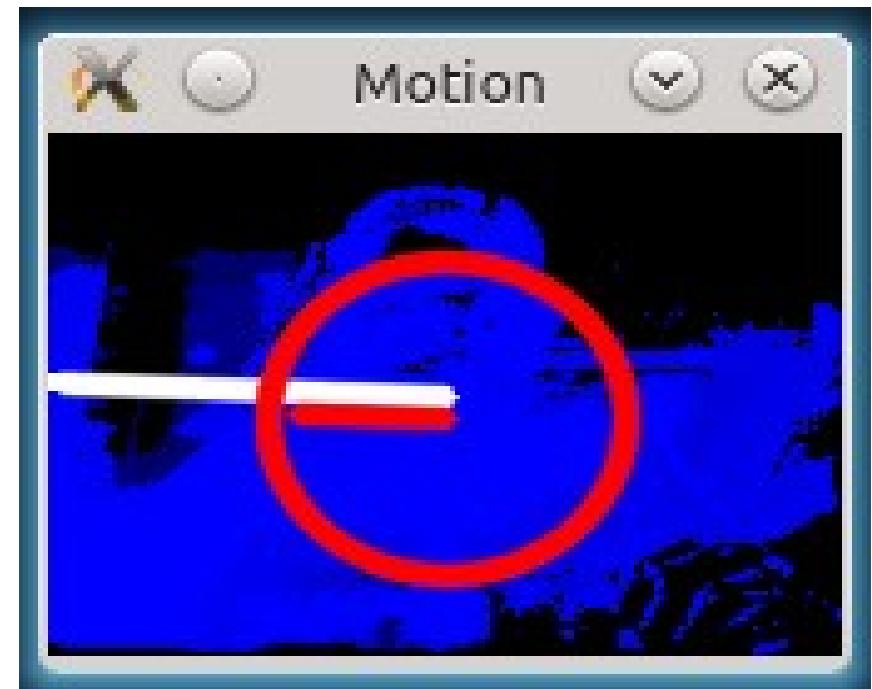
Detección de objetos por modelos (y V)

- *Detección de*
 - Caras, ojos, labios → *eyedetector*
 - Manos → *HandTracking*
 - Personas → *peopledetect*
 - *haarcascade_fullbody.xml*, *haarcascade_upperbody.xml*,
haarcascade_lowerbody.xml o *haarcascade_mcs_upperbody.xml*



Detección de movimiento

- *Ejemplo de la distribución de OpenCV*
 - Movimiento (examples/c/motempl.c)
 - Cambios importantes en los valores de la imagen
 - Agrupaciones → tendencias o direcciones



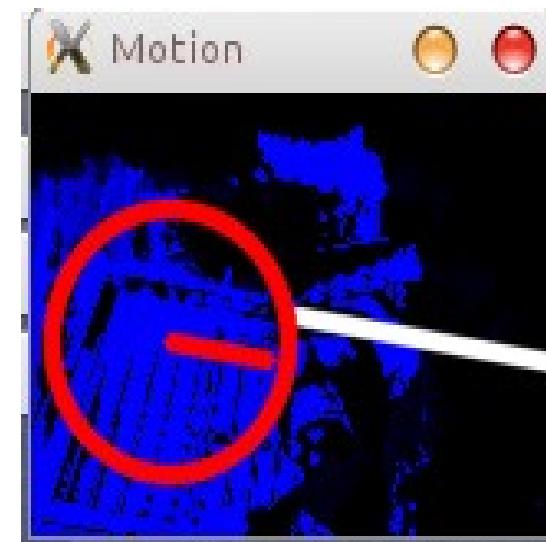
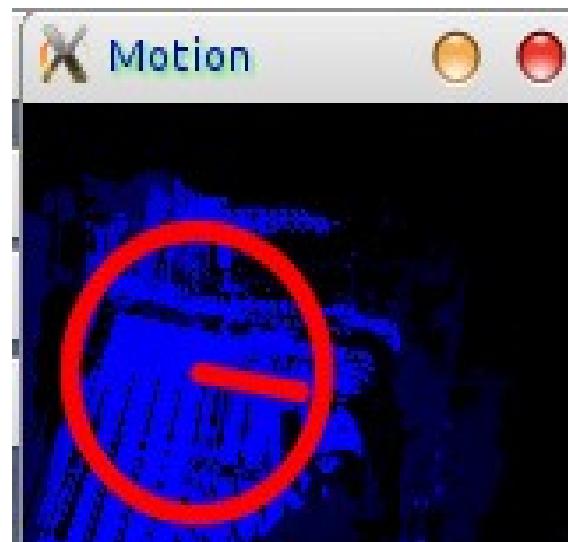
Detección de movimiento (II)

- *Bucle principal*

```
for(;;)
{
    cv::Mat* image = cvQueryFrame( capture );
    if( !image )
        break;
    ...

    update_mhi( image, motion, 30 );
    cv::imshow( "Motion", motion );

    if( cv::waitKey(10) >= 0 )
        break;
}
```



Detección de movimiento (y III)

- *Secuencia de movimientos*

```
void update_mhi( IplImage* img, IplImage* dst, int diff_threshold )  
{  
    IplImage* image = cvQueryFrame( capture );  
  
    cvCvtColor( img, buf[last], CV_BGR2GRAY ); // to grayscale  
    ...  
    cvAbsDiff( buf[idx1], buf[idx2], silh ); // get difference between frames  
    cvThreshold( silh, silh, diff_threshold, 1, CV_THRESH_BINARY ); // and threshold it  
    cvUpdateMotionHistory( silh, mhi, timestamp, MHI_DURATION ); // update MHI  
    ...  
  
    {  
        ...  
        count = cvNorm( silh, 0, CV_L1, 0 ); // calculate number of points within silhouette ROI  
        ...  
        // draw a clock with arrow indicating the direction  
        center = cvPoint( (comp_rect.x + comp_rect.width/2),  
                         (comp_rect.y + comp_rect.height/2) );  
  
        cvCircle( dst, center, cvRound(magnitude*1.2), color, 3, CV_AA, 0 );  
        cvLine( dst, center, cvPoint( cvRound( center.x + magnitude*cos(angle*CV_PI/180)),  
                                     cvRound( center.y - magnitude*sin(angle*CV_PI/180))), color, 3, CV_AA, 0 );  
    }  
}
```



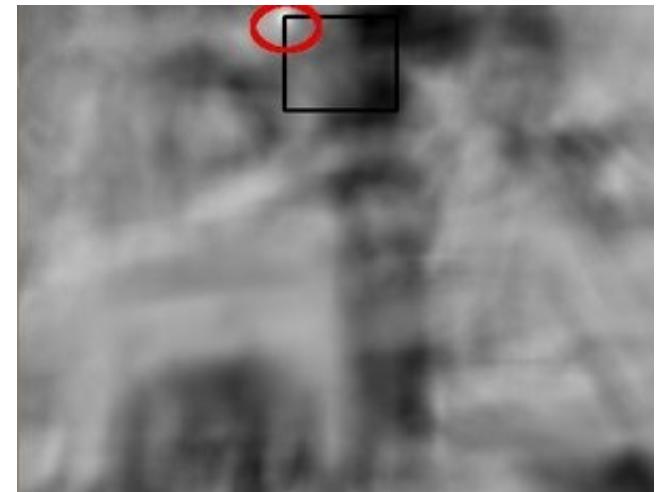
Detección de objetos a partir de patrones

- *Patrones:*
 - De texturas → *template_match*
 - Códigos 1D y 2D
 - De características → *puntos característicos o puntos de interés*

Detección de objetos a partir de patrones

- *Patrones:*

- De texturas → *matchTemplate*

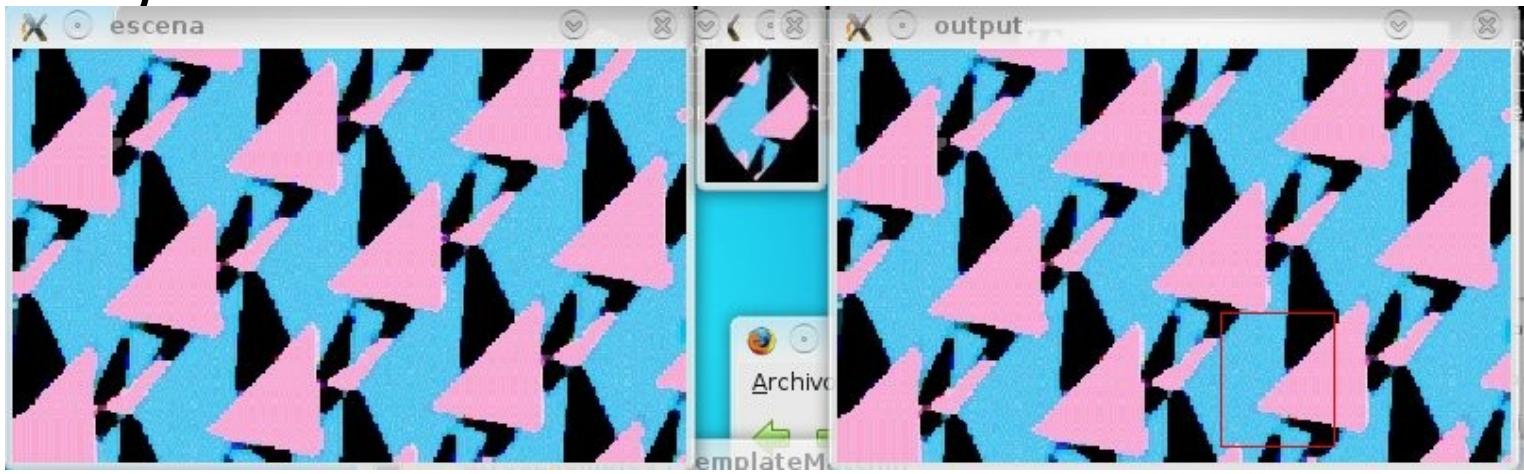


$$+ \begin{matrix} \text{template image} \\ \times 2 \end{matrix} =$$



Detección de objetos a partir de patrones

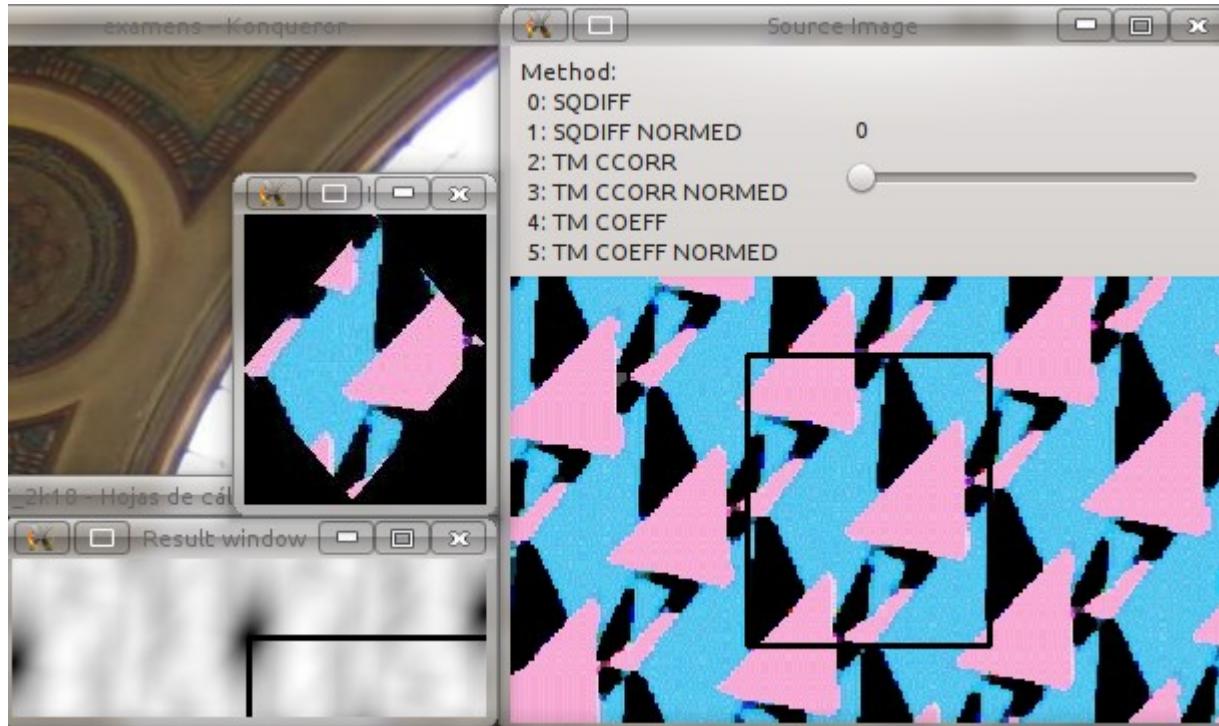
- *Búsqueda de patrones*



- *Ejemplo de uso de correlación de imágenes*

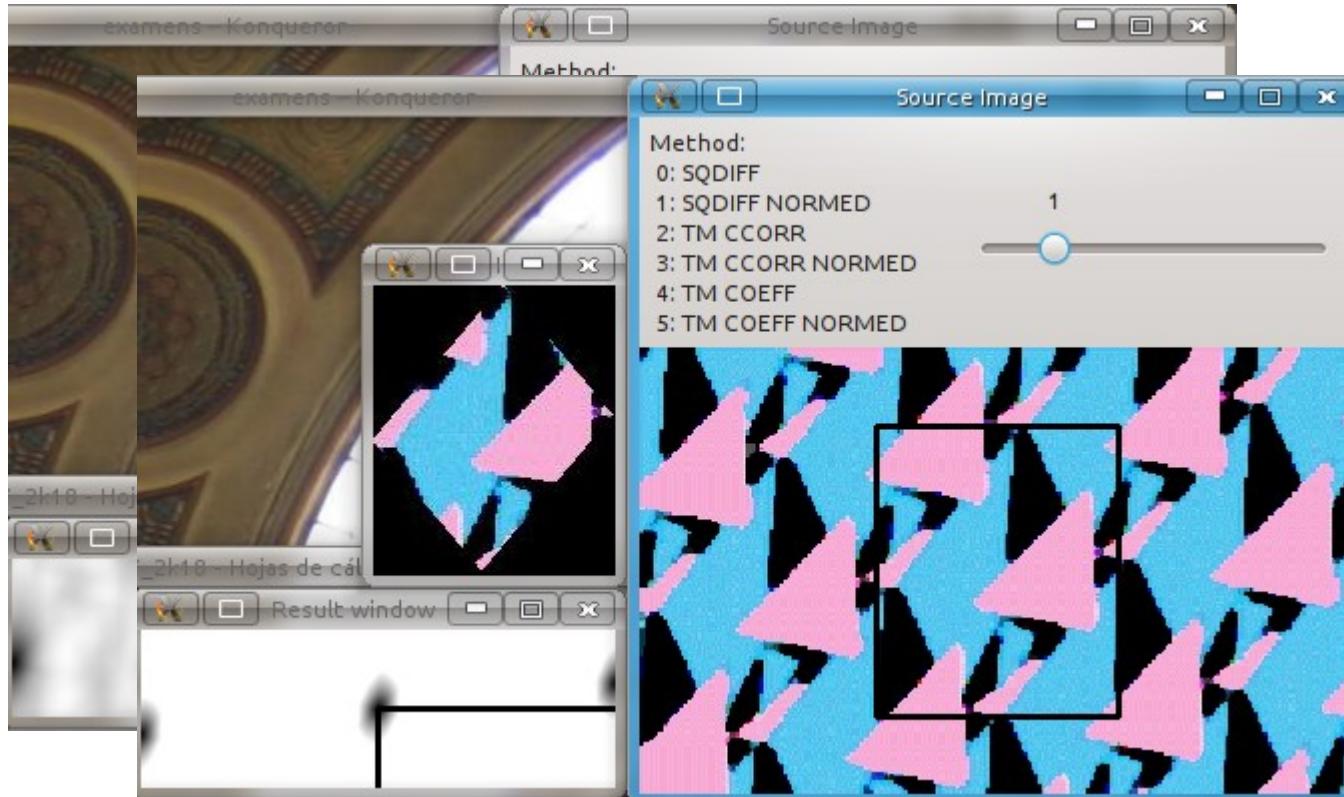
Detección de objetos a partir de patrones

- *Búsqueda de patrones*



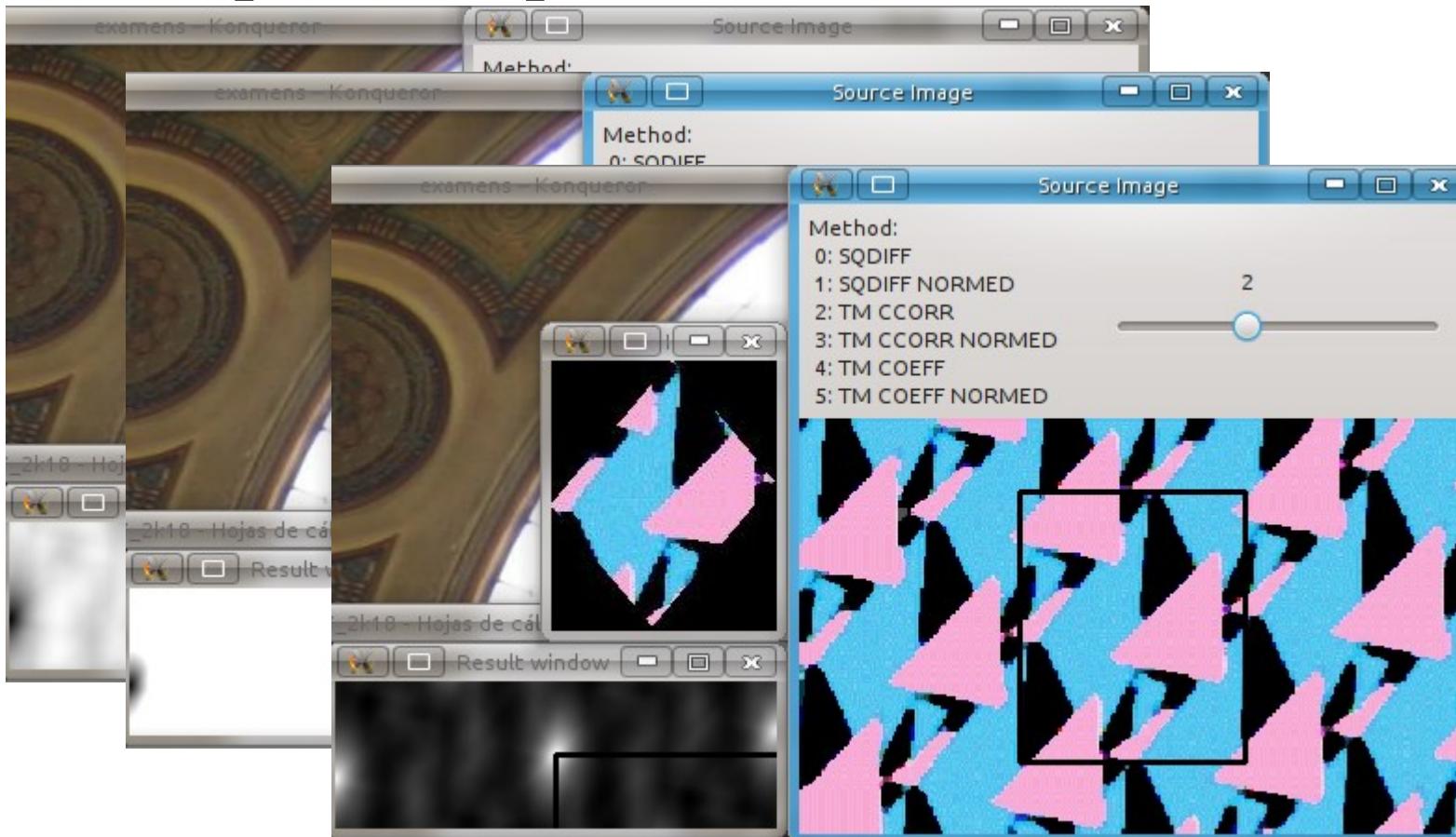
Detección de objetos a partir de patrones

- *Búsqueda de patrones*



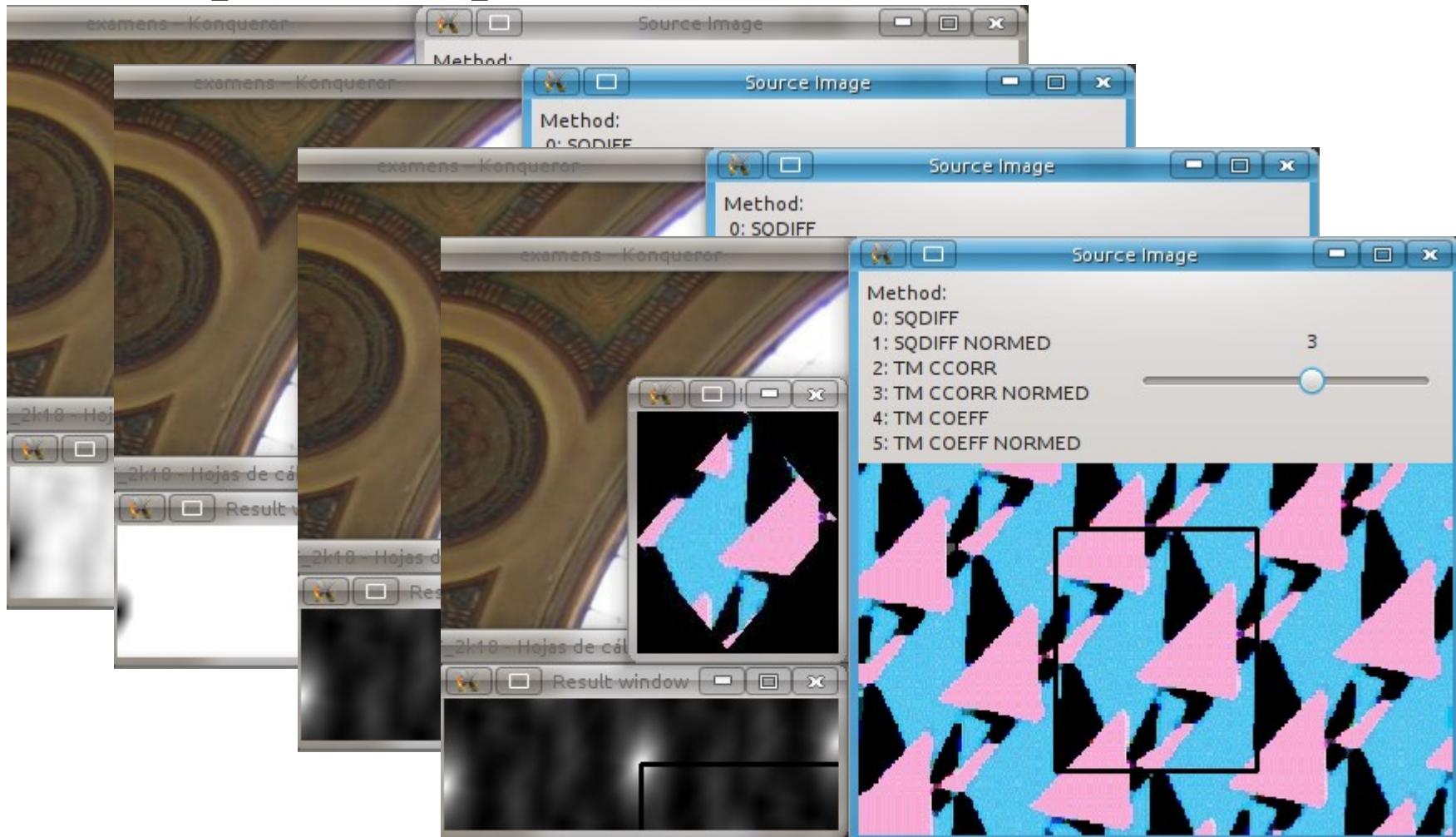
Detección de objetos a partir de patrones

- *Búsqueda de patrones*



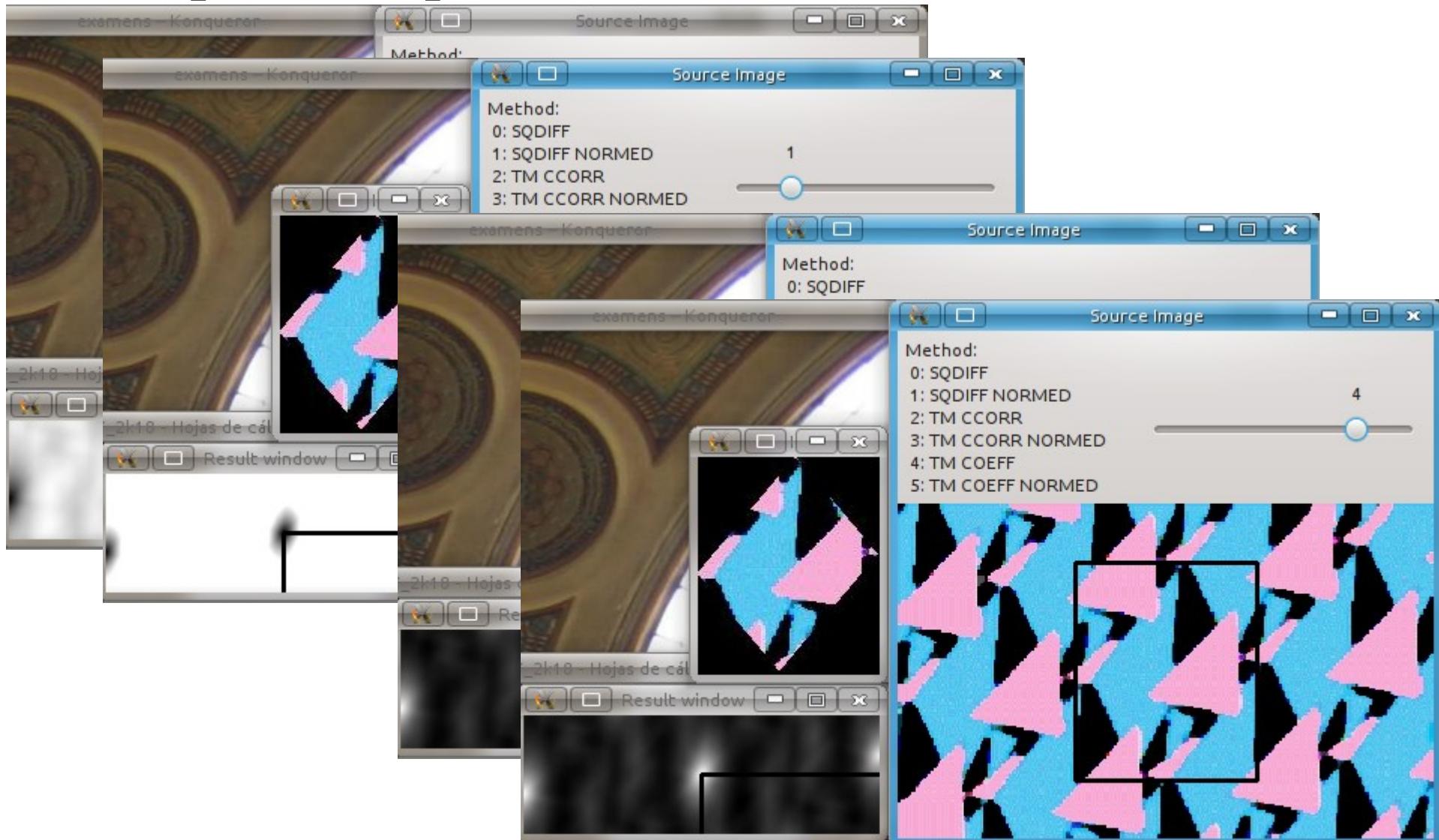
Detección de objetos a partir de patrones

- *Búsqueda de patrones*



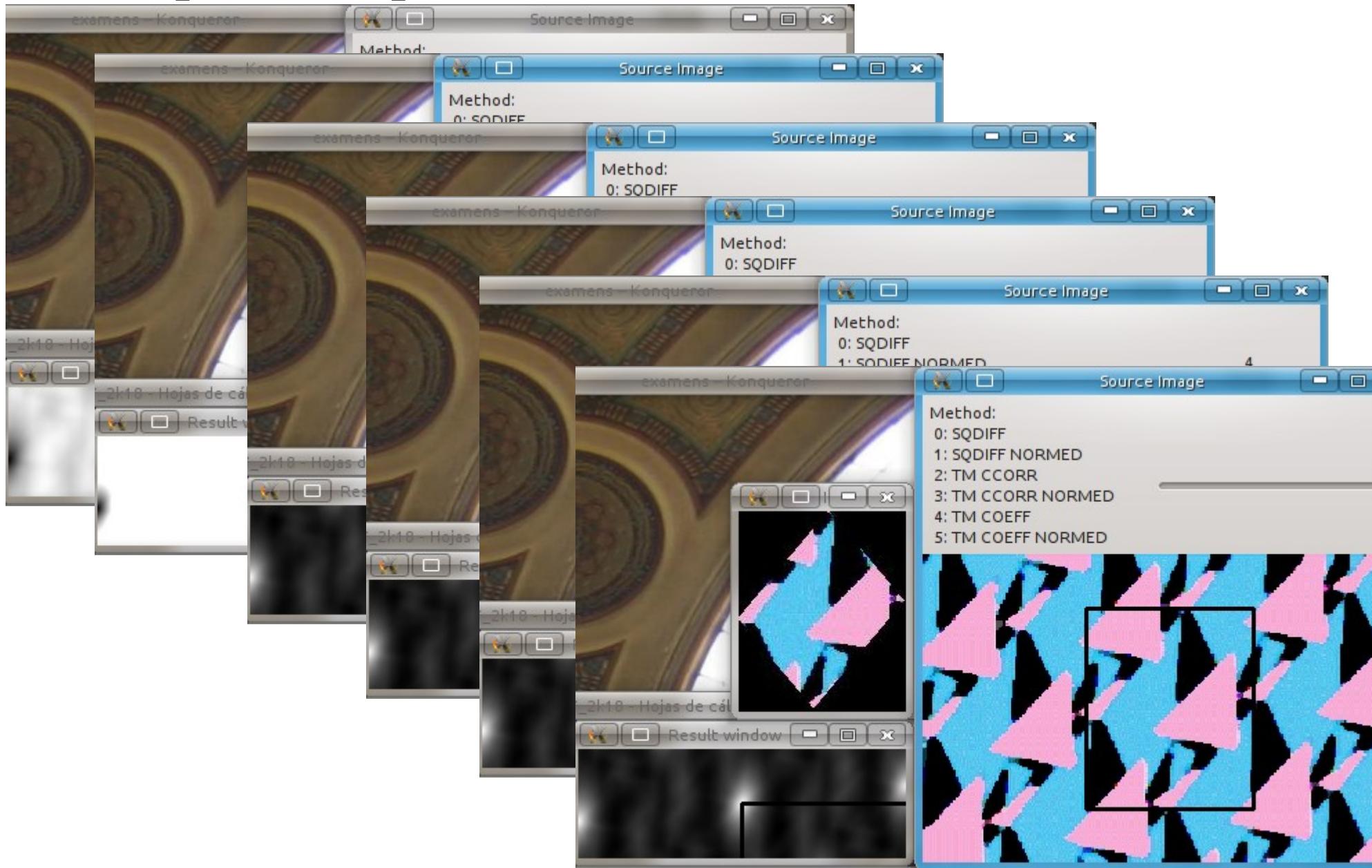
Detección de objetos a partir de patrones

- *Búsqueda de patrones*



Detección de objetos a partir de patrones

- *Búsqueda de patrones*



Detección de objetos a partir de patrones

- *Búsqueda de patrones*

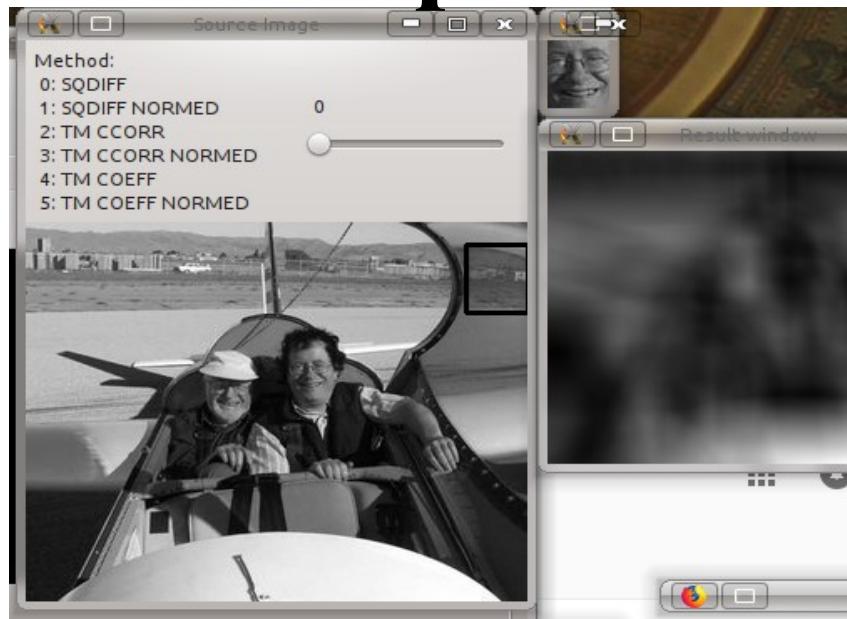
```
if(use_mask && method_accepts_mask)
{ matchTemplate( img, templ, result, match_method, mask); }
else
{ matchTemplate( img, templ, result, match_method); }

normalize( result, result, 0, 1, NORM_MINMAX, -1, Mat() );

// Localizing the best match with minMaxLoc
...
minMaxLoc( result, &minVal, &maxVal, &minLoc, &maxLoc, Mat() );

// For SQDIFF and SQDIFF_NORMED, the best matches are lower values.
// For all the other methods, the higher the better
if( match_method == TM_SQDIFF || match_method == TM_SQDIFF_NORMED )
{ matchLoc = minLoc; }
else
{ matchLoc = maxLoc; }

/// Show me what you got
rectangle( img_display, matchLoc, Point( matchLoc.x + templ.cols , matchLoc.y + templ.rows ),
Scalar::all(0), 2, 8, 0 );
```



Detectores de formas

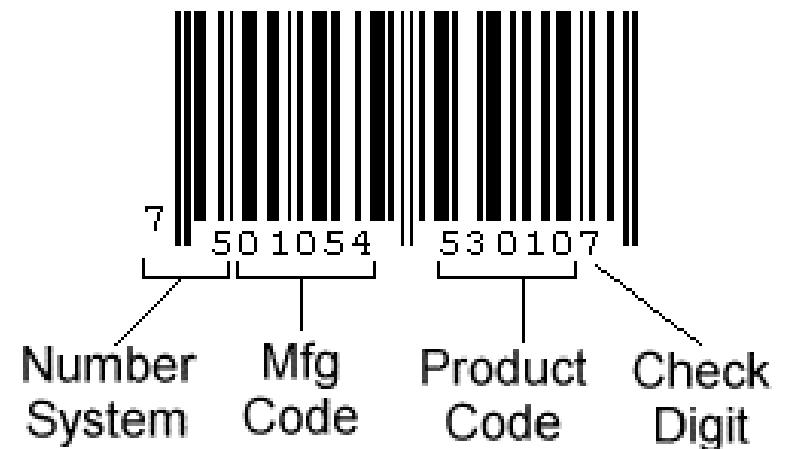
- *Formas complejas:*

- *Códigos 1D*

- *UPC-A (IBM, 1971)*



- *EAN 13 (International Article Numbering Association)*



Detectores de formas

- *Formas complejas:*
 - Códigos 1D y 2D



Figure 2

Fixed edges of UPC-A barcode shown as dashed red lines. Labels on bottom denote the guard regions and the 12 digit regions between the fixed edges.

Detectores de formas

- *Formas complejas:*
 - *Códigos 1D con OpenCV*



```
#include "opencv2/barcode.hpp"
#include "opencv2/imgproc.hpp"

using namespace cv;

Ptr<barcode::BarcodeDetector> bardet = makePtr<barcode::BarcodeDetector>();
Mat input = imread("your file path");
Mat corners; //the corners of detected barcodes, if N barcodes detected, then the shape is [N][4][2]
std::vector<std::string> decoded_info; //the decoded infos, if fail to decode, then the string is empty
std::vector<barcode::BarcodeType> decoded_format; //the decoded types, if fail to decode, then the type is
BarcodeType::NONE
bool ok = bardet->detectAndDecode(input, decoded_info, decoded_format, corners);
```



Detectores de formas

- *Formas complejas:*
 - Códigos 2D

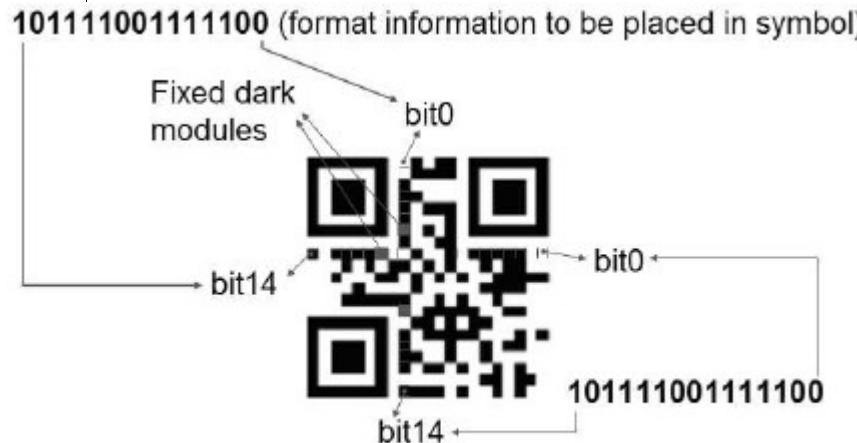


Figure 1.2 — Final 1-M symbol encoding 01234567

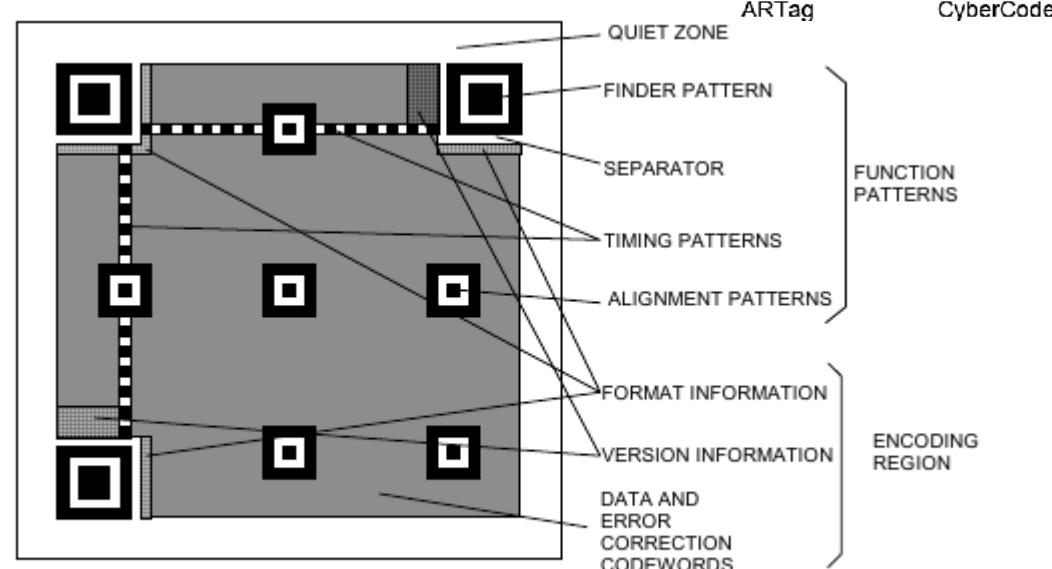
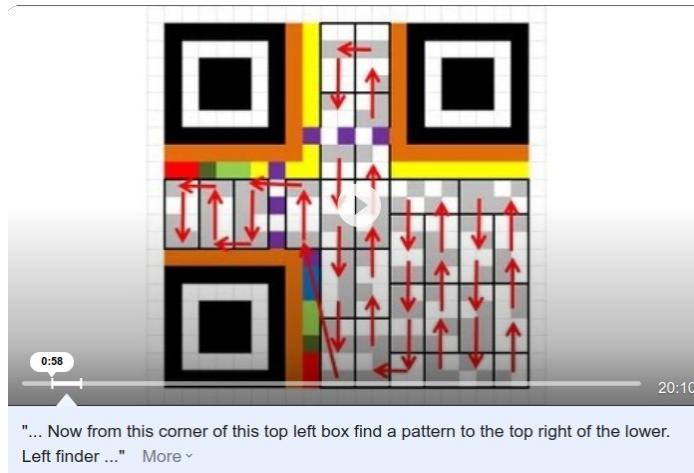
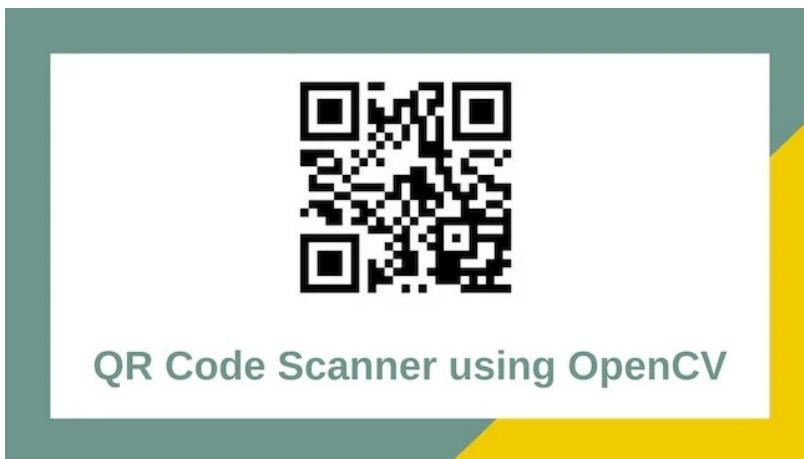


Figure 3 — Structure of a QR Code 2005 symbol

Detectores de formas

- *Formas complejas:*
 - Códigos 2D con OpenCV



- Ejemplo

- \$ barcode-QRcodeScanner

Time Taken for Detect and Decode : 0.044 seconds

Decoded Data / http://LearnOpenCV.com

```
#include <opencv2/objdetect.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <iostream>

using namespace cv;
using namespace std;

void display(Mat &im, Mat &bbox)
{
    int n = bbox.rows;
    for(int i = 0 ; i < n ; i++)
    {
        line(im, Point2i(bbox.at<float>(i,0),bbox.at<float>(i,1)),
        Point2i(bbox.at<float>((i+1) % n,0), bbox.at<float>((i+1) % n,1)),
        Scalar(255,0,0), 3);
    }
    imshow("Result", im);
}

int main(int argc, char* argv[])
{
    // Read image
    Mat inputImage;
    if(argc>1)
        inputImage = imread(argv[1]);
    else
        inputImage = imread("qrcode-learnopencv.jpg");

    QRCodeDetector qrDecoder = QRCodeDetector::QRCodeDetector();

    Mat bbox, rectifiedImage;

    std::string data = qrDecoder.detectAndDecode(inputImage, bbox,
    rectifiedImage);
    if(data.length()>0)
    {
        cout << "Decoded Data : " << data << endl;

        display(inputImage, bbox);
        rectifiedImage.convertTo(rectifiedImage, CV_8UC3);
        imshow("Rectified QRCode", rectifiedImage);

        waitKey(0);
    }
    else
        cout << "QR Code not detected" << endl;
}
```

```

#include <opencv2/opencv.hpp>
#include <zbar.h>

using namespace cv;
using namespace std;
using namespace zbar;

typedef struct
{
    string type;
    string data;
    vector<Point> location;
} decodedObject;

// Find and decode barcodes and QR codes
void decode(Mat &im, vector<decodedObject>&decodedObjects) {
    // Create zbar scanner
    ImageScanner scanner;

    // Configure scanner
    scanner.set_config(ZBAR_NONE, ZBAR_CFG_ENABLE, 1);

    // Convert image to grayscale
    Mat imGray;
    cvtColor(im, imGray,COLOR_BGR2GRAY);

    // Wrap image data in a zbar image
    Image image(im.cols, im.rows, "Y800", (uchar *)imGray.data, im.cols * im.rows);

    // Scan the image for barcodes and QRCodes
    int n = scanner.scan(image);

    // Print results
    for(Image::SymbolIterator symbol = image.symbol_begin();
        symbol != image.symbol_end(); ++symbol) {
        decodedObject obj;
        obj.type = symbol->get_type_name();
        obj.data = symbol->get_data();

        // Print type and data
        cout << "Type : " << obj.type << endl;
        cout << "Data : " << obj.data << endl << endl;

        // Obtain location
        for(int i = 0; i < symbol->get_location_size(); i++) {
            obj.location.push_back(Point(symbol->get_location_x(i),symbol->get_location_y(i)));
        }

        decodedObjects.push_back(obj);
    }
}

// Display barcode and QR code location
void display(Mat &im, vector<decodedObject>&decodedObjects)
{
    // Loop over all decoded objects
    for(int i = 0; i < decodedObjects.size(); i++) {
        vector<Point> points = decodedObjects[i].location;
        vector<Point> hull;

        // If the points do not form a quad, find convex hull
        if(points.size() > 4)
            convexHull(points, hull);
        else
            hull = points;

        // Number of points in the convex hull
        int n = hull.size();

        for(int j = 0; j < n; j++) {
            line(im, hull[j], hull[ (j+1) % n], Scalar(255,0,0), 3);
        }
    }

    // Display results
    imshow("Results", im);
    waitKey(0);
}

int main(int argc, char* argv[])
{
    // Read image
    Mat im = imread("zbar-test.jpg");

    // Variable for decoded objects
    vector<decodedObject> decodedObjects;

    // Find and decode barcodes and QR codes
    decode(im, decodedObjects);

    // Display location
    display(im, decodedObjects);

    return EXIT_SUCCESS;
}

```

Detectores de formas

- *Formas complejas:*
 - Códigos 1D y 2D con Zbar
 - Ejemplo

- \$ barcode-QRcodeScanner
Type : QR-Code
Data : <http://LearnOpenCV.com>
Type : CODE-128
Data : LearnOpenCV.com

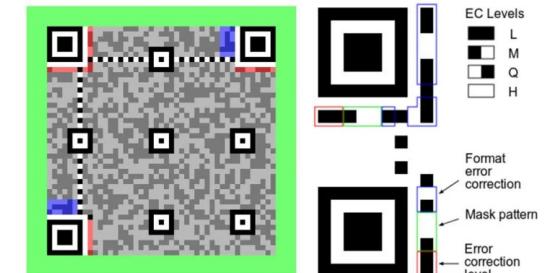


- \$ barcode-QRcodeScanner ../magusti_qrcode.png
Type : QR-Code
Data : Manuel Agustín Melchor
magusti@disca.upv.es
<http://www.disca.upv.es/magustim>
 - \$ barcode-QRcodeScanner ../magustimURL_qrcode.png
Type : QR-Code
Data : <http://www.disca.upv.es/magustim>



Detectores de formas

- *Formas complejas:*
 - *Códigos 1D y 2D con WeChat QR Scanner (CNN)*



WeChat QR Code Scanner in OpenCV



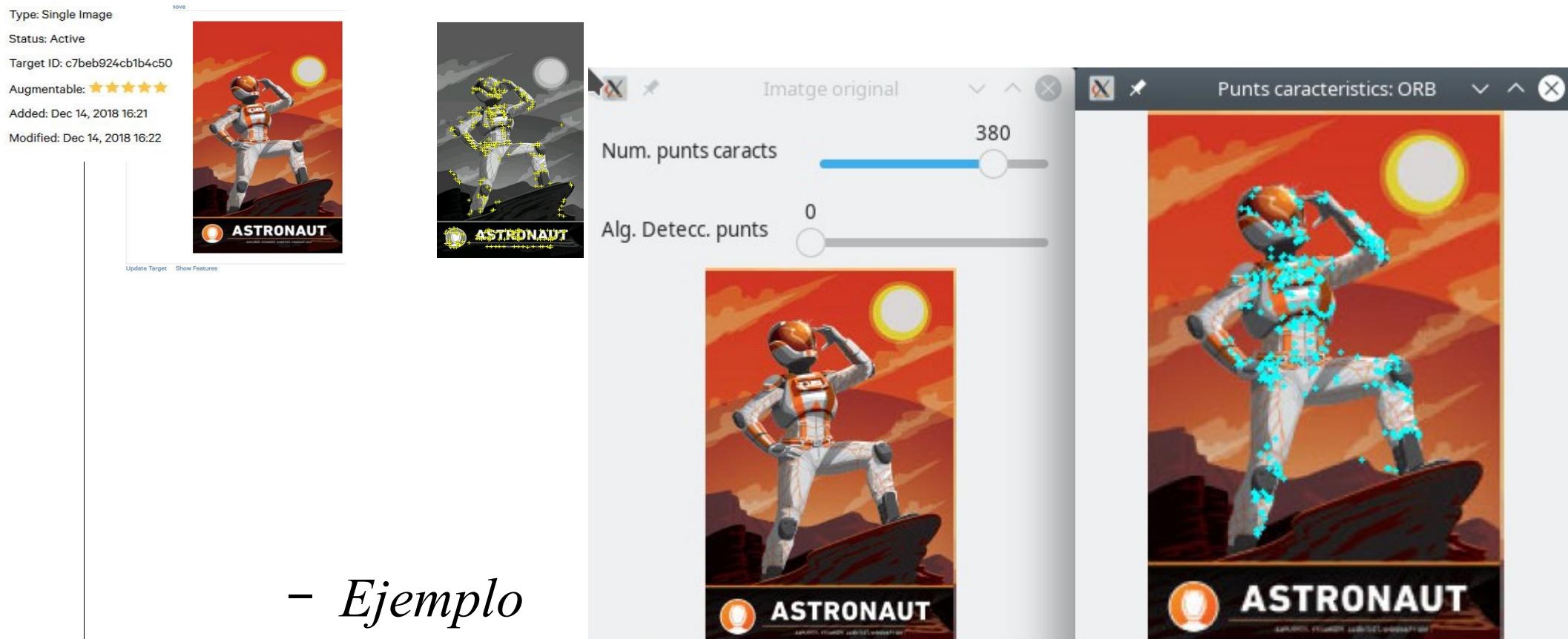
● LearnOpenCV.com



<https://www.youtube.com/watch?v=bMBHuct_zyY>

Detectores de formas

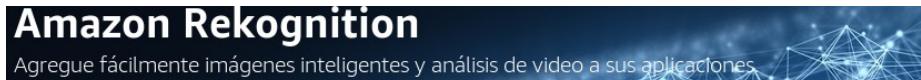
- *Formas complejas:*
 - Búsqueda de descriptores → SURF, SIFT, ORB, AKAZE, KAZE, MSER, BRISK, ...



- *Ejemplo*

- `vorePuntsCaracteristics_orb images/vuforiaImages_astronaut.png`

Bibliografía

- Open Computer Vision Library (OpenCV)
 - Sitio web <<http://opencv.org>>
 - Documentación <<https://docs.opencv.org/>>
 - Repositorio <<http://sourceforge.net/projects/opencvlibrary>> / Github <<https://github.com/opencv/opencv>>
- Kaehler, A. (2017). Cloudy Vision Output: a tool to help compare computer vision API vendors <https://goberoi.github.io/cloudy_vision/output/output.html>
- Bobriakov, I. (2018). Comparison of Top 6 Cloud APIs for Computer Vision <<https://medium.com/activewizards-machine-learning-company/comparison-of-top-6-cloud-apis-for-computer-vision-ebf2d299be73>>
- Clarifi
- Rekognition | Amazon  **Amazon Rekognition**
Agregue fácilmente imágenes inteligentes y análisis de video a sus aplicaciones
- Visual Recognition | IBM Watson  Visual Recognition
- Computer Vision. < Cognitives Services | Microsoft Azure  Computer Vision
- Cloud Vision. AI & Machine Learning Products | Google Cloud, 
- Cloud Sight 
- Emgu CV ← .Net wrapper para OpenCV

Para acabar el tema de VxC ...

- *Antes de empezar tema de Audio:*
 - *Lanzar la idea*
“Interacción natural en instrumentos musicales”
→ *Sonido e instrumentos virtuales*
 - Encargar la tarea de OpenCV + OpenAL
 - Generación de sonidos básicos a partir del movimiento en la escena
 - Presentar el uso del movimiento para implementar instrumentos musicales virtuales
 - Theremín
 - Arpa láser

Ejemplo de integración



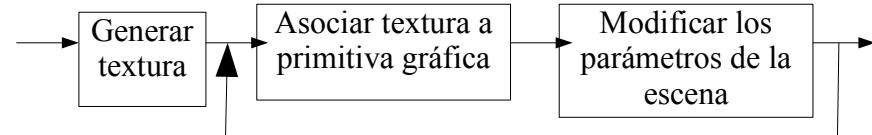
Fuente: *La prisión de cristal* (también llamada “*La zona fantasma*”), fotograma de la película *Superman* (1978).
<<https://myintuitivelife.files.wordpress.com/2012/03/superman-ii-dim.jpg>>.

Ejemplo de integración

- OpenGL + OpenCV

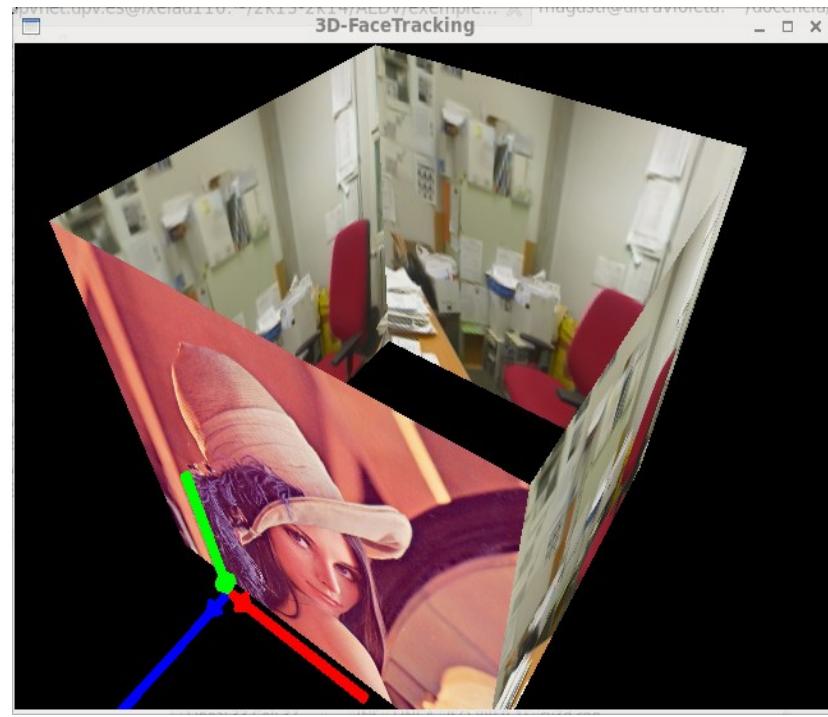
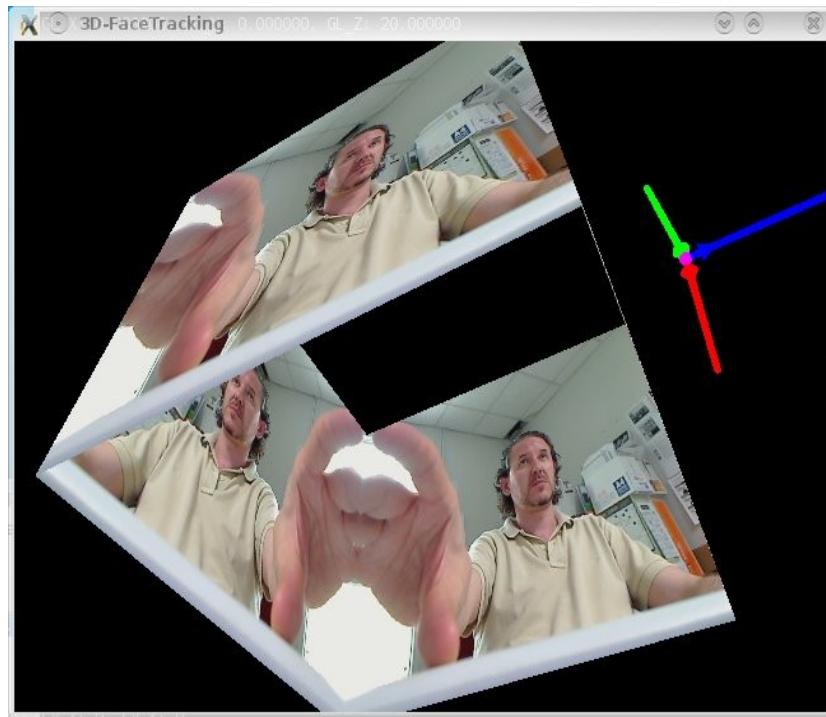
- Imagen sintética + real

```
int loadTexture_Ipl(IplImage *image, GLuint *text) {  
    ...  
    glBindTexture(GL_TEXTURE_2D, texture); //bind the texture  
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE);  
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);  
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);  
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, image->width, image->height,  
                0, GL_BGR, GL_UNSIGNED_BYTE, image->imageData);  
    ...  
}  
void plane (void) {  
    ...  
    glBegin(GL_QUADS);  
    glTexCoord2d(0.0,0.0); glVertex2d(-1.0,-1.0); //with our vertices we have to assign a texcoord  
    glTexCoord2d(1.0,0.0); glVertex2d(+1.0,-1.0); //so that our texture has some points to draw to  
    glTexCoord2d(1.0,1.0); glVertex2d(+1.0,+1.0);  
    glTexCoord2d(0.0,1.0); glVertex2d(-1.0,+1.0);  
    glEnd();  
}  
...  
}
```



Ejemplo de integración

- OpenGL + OpenCV
 - Imagen sintética + real



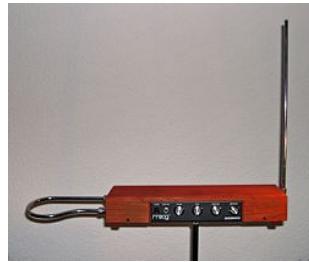
- ¿Y después?

Sonidos básicos

- *Explicando la escala musical*
- *OpenCV + OpenAL*
- *Ejemplo*
 - *Elena + Nacio. 2K10/2k11. “Sonido en movimiento”*

Sonido e instrumentos virtuales

- *Interacción natural en instrumentos*
 - *Theremin*
 - *Instrumento musical sin contacto directo*
 - “*the loop antenna on the left controls the volume while the upright antenna controls the pitch*”.



Lev Theremin, 1927



An Etherwave-Theremin, assembled from Robert Moog's kit

- *Arpa láser*

Theremin

- *Fernando*
 - *Instrumento*
 - *Ejemplo*
 - *OpenCV + OpenAL*
 - *Posición/Movimiento en los dos ejes*

Arpa láser

- *Interacción natural en instrumentos*
 - *Theremin*
 - *Arpa láser (Laser Harp)*

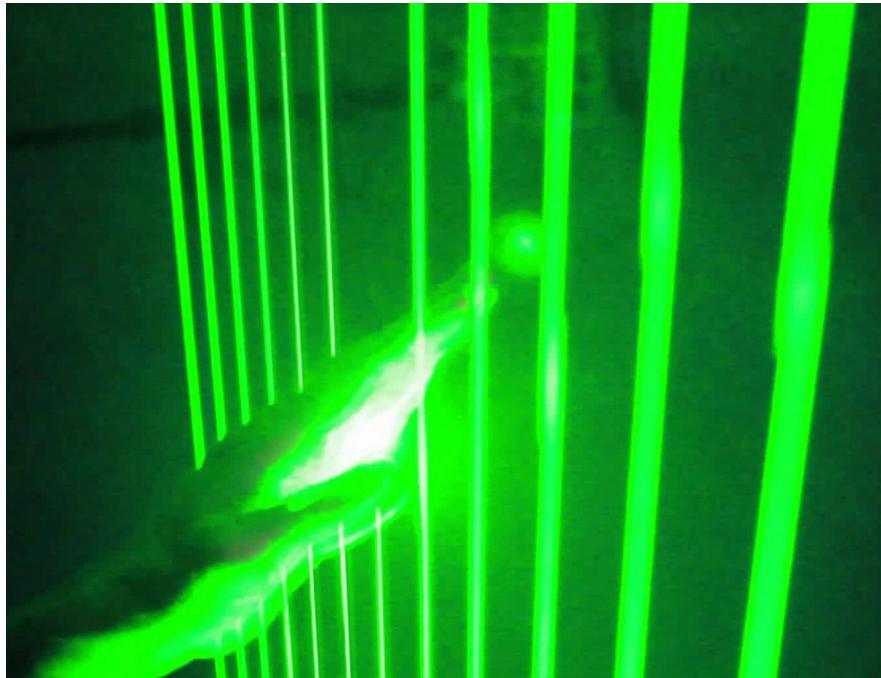


- *A light-sensitive musical instrument. It is played by moving hands over laser light sources in order to send MIDI commands when a beam is interrupted. ...*
- “*The first Laser Harp was invented and played by Bernard SZAJNER*”
 - (<www.harpelaser.com>, 1981)
- *Jean Michel Jarre*
 - *A partir de los sonidos del Elka Synthethizer Laser Harp Sound*
- *Lasertron: Laser Harp App for iPhone, iPad*



Arpa láser

- Idea
 - Crear una aplicación que simule tocar un arpa
 - OpenCV para análisis de imagen de la cámara
 - OpenAL para las fuentes de audio.
- Solución
 - Identificar qué haz de luz es tocada.
 - ¿Interacción por color con OpenCV?
 - ¿Detectar movimiento sobre el “haz de luz”?

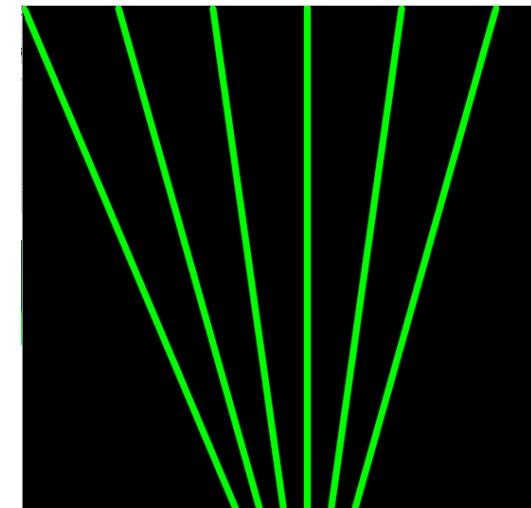


Arpa láser



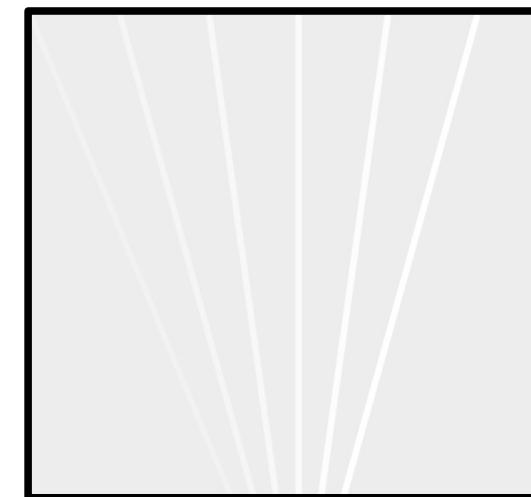
- Eventos ← OpenCV
 - Imagen del “instrumento”

```
for(int i=0;i<6;i++){  
    cvLine(imagen, cvPoint(300,800), cvPoint(i*100,0),  
    cvScalar(0,255,0, 1),5, line_type, shift );  
    cvLine(imagen2, cvPoint(300,800), cvPoint(i*100,0),  
    cvScalar(10+i*10,10+i*10,10+i*10, 1),5,  
            line_type, shift );  
}
```

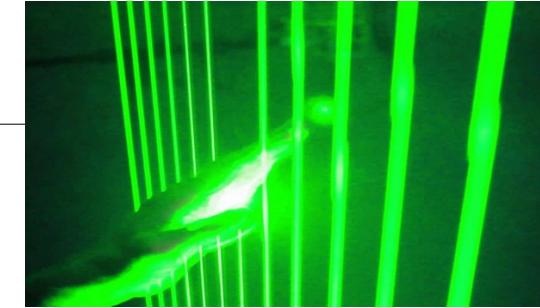


- Imagen intermedia: IDs de las “cuerdas

```
elColor = cvGet2D(imagenIntermedia, y, x);  
for(int i=0;i<6;i++){  
    if((int)elColor.val[0]==i*10+10){  
        antigua=i;  
        cvLine(imagen, cvPoint(300,800), cvPoint(i*100,0),  
        cvScalar(0,0,0, 1), 5, line_type, shift );  
        cvLine(imagen, cvPoint(300,800), cvPoint(x,y),  
        cvScalar(0,255,0, 1), 5, line_type, shift );  
  
        //sonido que tiene que reproducir  
    }  
}
```



Arpa láser



- Sonido ← OpenAL

- Inicializar sonidos

```
alGenBuffers(6,Buffer);
Buffer[0]= alutCreateBufferFromFile( "arpa.wav" );
...
Buffer[5]= alutCreateBufferFromFile( "arpa3.wav" );
// ¿Polifonía?
alGenSources(6,Source);
```

- Reproducir uno

```
alSourceStop(Source[antigua]);
alSourcei (Source[i], AL_BUFFER, Buffer[i]);
alSourcePlay(Source[i]);
```

