Visión artificial con

15 de febrero de 2017

Ricardo Samaniego
Director de Proyectos
ricardo.samaniego@imatia.com

Organización

1. Introducción a OpenCV
2. Aplicaciones (demos y vídeos)
3. Recursos

Lanzamiento de versión alfa en 1999

Software libre con licencia permisiva (BSD)

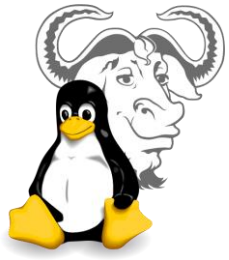
OpenCV

0.x	1.0	2.0	3.0
1999	2006	2009	2015
intel	Willow Garage	itseez	intel
C	C	C++	C++ (T-API)

OpenCV







500 programadores activos
47.000 contribuciones
9.000.000 de descargas

OpenCV 3.x

OpenCV

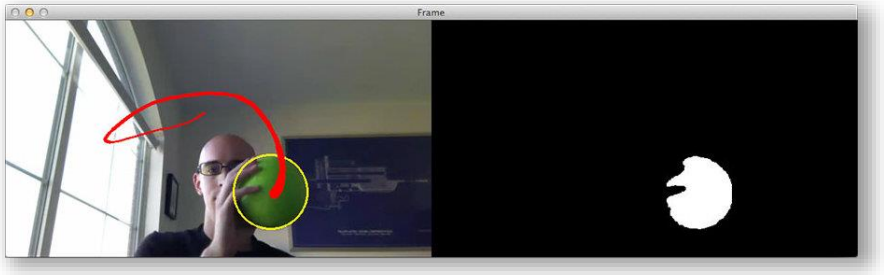
Camera Calibration
Face recognition
Deep neural networks
Biologically inspired vision models
High-level GUI
Optical Flow
Machine Learning
Background Segmentation
Video Analysis
3D visualization
Image processing
Images stitching
Tracking API
Structure from motion
Multi-dimensional clustering
Deformable Models
Object Detection
Structured light API
Video Stabilization
Stereo 3D Reconstruction
Text recognition
RGBD processing
Computational Photography
Shape matching
AR marker detection

Más información: http://docs.opencv.org/3.1.0/

Ejemplo básico

```
1.  #include <opencv2/opencv.hpp>
2.
3.  int main(int argc, char** argv)
4.  {
5.      cv::UMat img;                                   // Image object
6.      cv::VideoCapture cap(0);                        // Open first webcam found
7.
8.      while(true)                                     // Forever
9.      {
10.         cap >> img;                                 // Take image
11.
12.         cv::imshow("Meetup Vigolabs", img);         // Show on window
13.         if((uchar) cv::waitKey(40) == 27) return 0; // Exit on "esc" key
14.     }
15.
16.     return 0;                                       // End of program
17. }
```

Ejemplo en Python




```python
# import the necessary packages
from collections import deque
import numpy as np
import argparse
import imutils
import cv2
import sys

PY3 = sys.version_info[0] == 3
if PY3:
    xrange = range

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video",
    help="path to the (optional) video file")
ap.add_argument("-b", "--buffer", type=int, default=64,
    help="max buffer size")
args = vars(ap.parse_args())

# define the lower and upper boundaries of the "green"
# ball in the HSV color space, then initialize the
# list of tracked points
greenLower = (49, 107, 65)
greenUpper = (89, 216, 191)
pts = deque(maxlen=args["buffer"])

# if a video path was not supplied, grab the reference
# to the webcam
if not args.get("video", False):
    camera = cv2.VideoCapture(0)

# otherwise, grab a reference to the video file
else:
    camera = cv2.VideoCapture(args["video"])
```

```python
# keep looping
while True:
    # grab the current frame
    (grabbed, frame) = camera.read()

    # if we are viewing a video and we did not grab a frame,
    # then we have reached the end of the video
    if args.get("video") and not grabbed:
        break

    # resize the frame, blur it, and convert it to the HSV
    # color space
    frame = imutils.resize(frame, width=600)
    # blurred = cv2.GaussianBlur(frame, (11, 11), 0)
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # construct a mask for the color "green", then perform
    # a series of dilations and erosions to remove any small
    # blobs left in the mask
    mask = cv2.inRange(hsv, greenLower, greenUpper)
    mask = cv2.erode(mask, None, iterations=2)
    mask = cv2.dilate(mask, None, iterations=2)

    # find contours in the mask and initialize the current
    # (x, y) center of the ball
    cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)[-2]
    center = None

    # only proceed if at least one contour was found
    if len(cnts) > 0:
        # find the largest contour in the mask, then use
        # it to compute the minimum enclosing circle and
        # centroid
        c = max(cnts, key=cv2.contourArea)
        ((x, y), radius) = cv2.minEnclosingCircle(c)
```

```python
        M = cv2.moments(c)
        center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))

        # only proceed if the radius meets a minimum size
        if radius > 10:
            # draw the circle and centroid on the frame,
            # then update the list of tracked points
            cv2.circle(frame, (int(x), int(y)), int(radius), (0, 255, 255), 2)
            cv2.circle(frame, center, 5, (0, 0, 255), -1)

    # update the points queue
    pts.appendleft(center)

    # loop over the set of tracked points
    for i in xrange(1, len(pts)):
        # if either of the tracked points are None, ignore
        # them
        if pts[i - 1] is None or pts[i] is None:
            continue

        # otherwise, compute the thickness of the line and
        # draw the connecting lines
        thickness = int(np.sqrt(args["buffer"] / float(i + 1)) * 2.5)
        cv2.line(frame, pts[i - 1], pts[i], (0, 0, 255), thickness)

    # show the frame to our screen
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the 'q' key is pressed, stop the loop
    if key == ord("q"):
        break

# cleanup the camera and close any open windows
camera.release()
cv2.destroyAllWindows()
```

Basado en “Ball Tracking with OpenCV”, de Adrian Rosebrock

http://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/

Flujo óptico: demo

Sparse ("disperso")

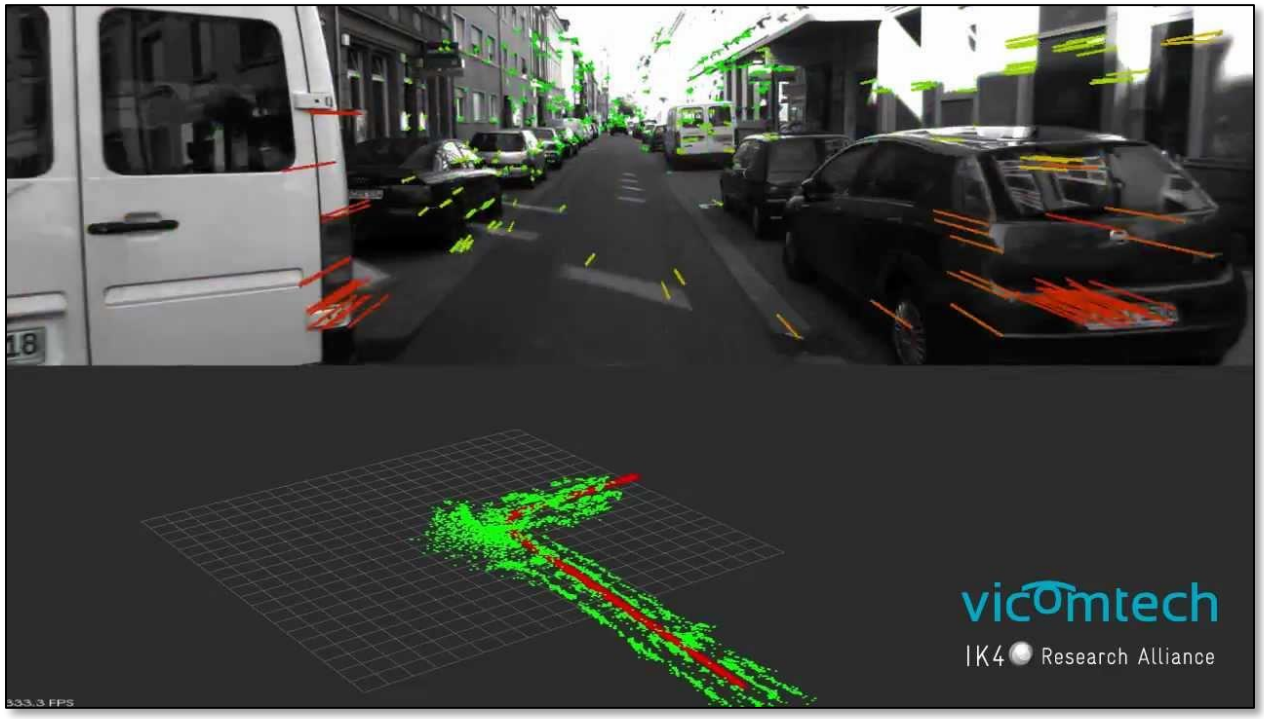
```
cv2.goodFeaturesToTrack(image, ...)
cv2.calcOpticalFlowPyrLK(prevImg, nextImg, prevPts, nextPts, ...)
```

Dense ("denso")

```
cv2.calcOpticalFlowFarneback(prevImg, nextImg, flow, ...)
```

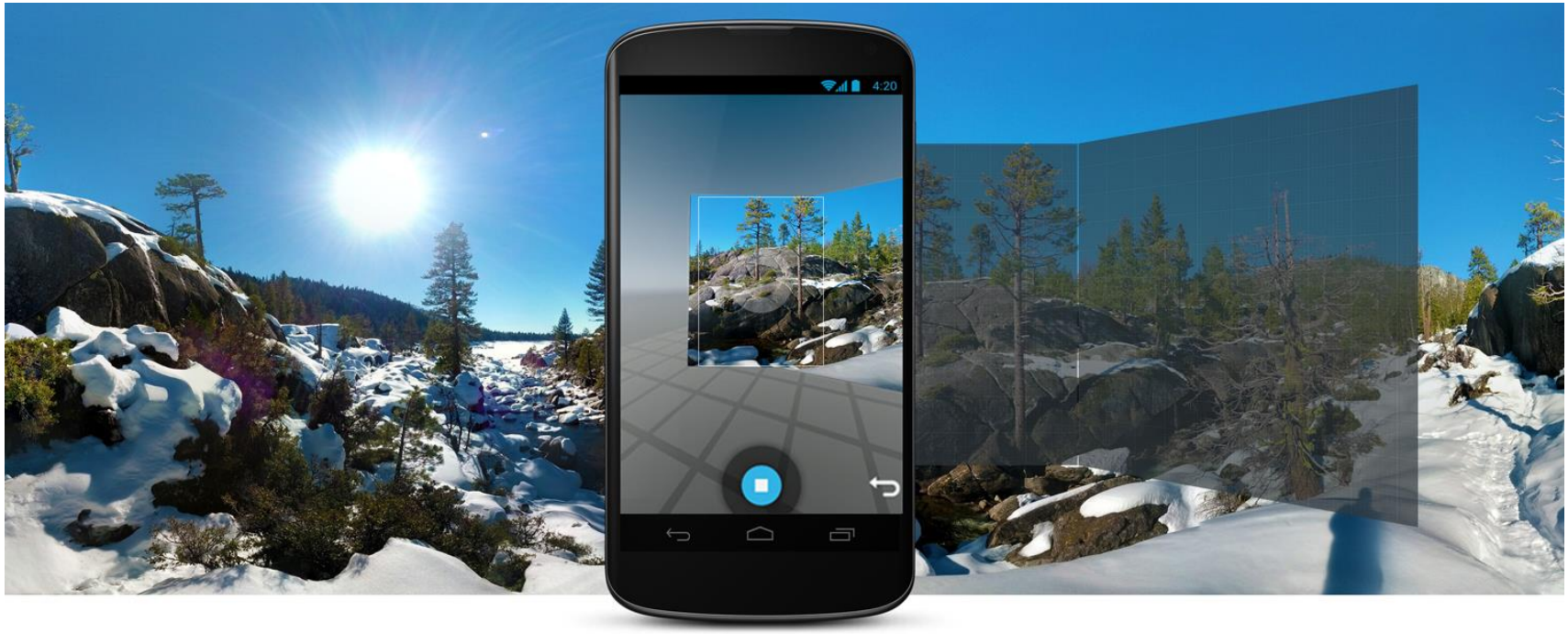
Flujo óptico: Aplicaciones (I)

Odometría visual



Flujo óptico: Aplicaciones (II)

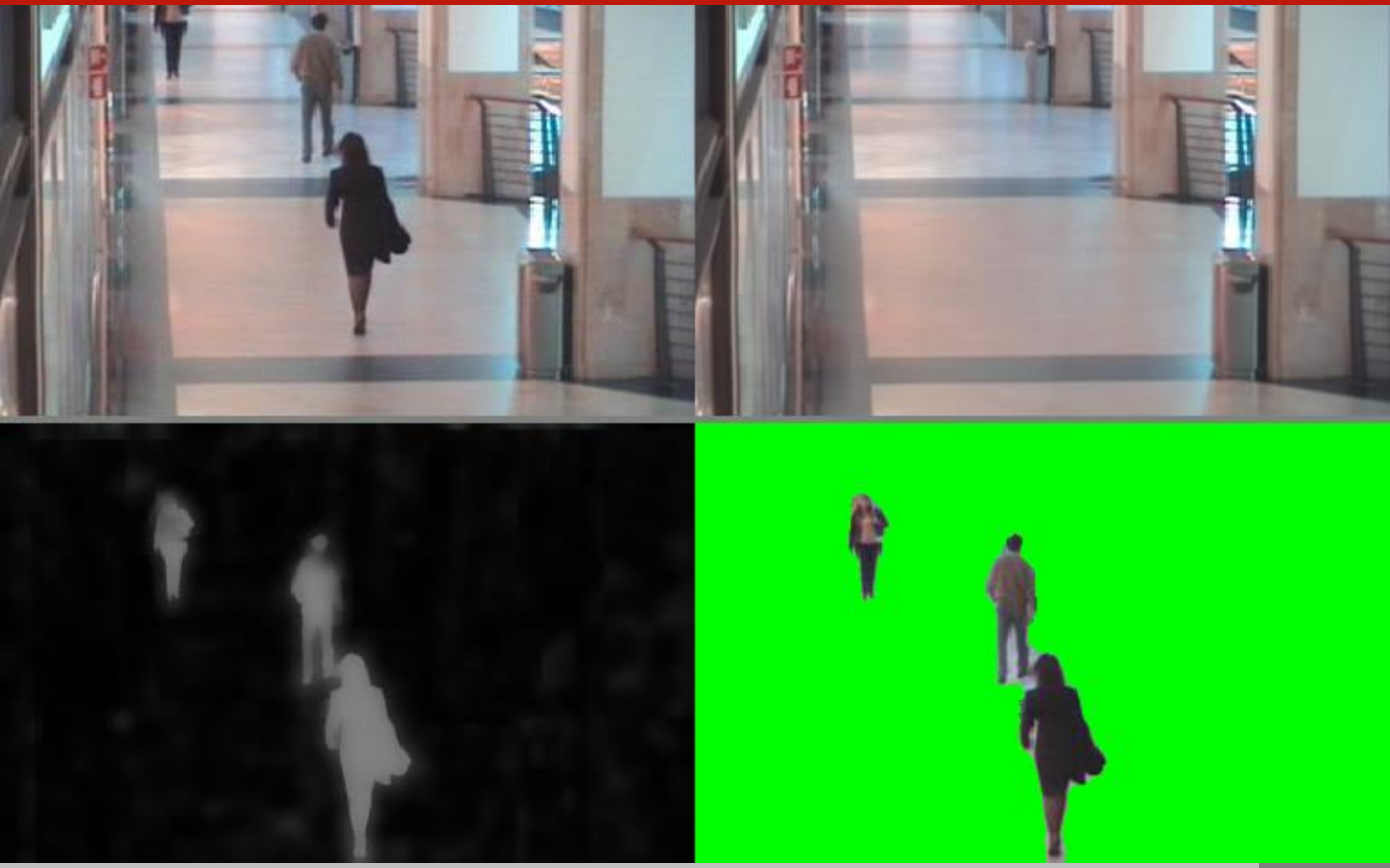
Panoramas



Flujo óptico: Aplicaciones (y III)

Structure From Motion

Supresión de fondo



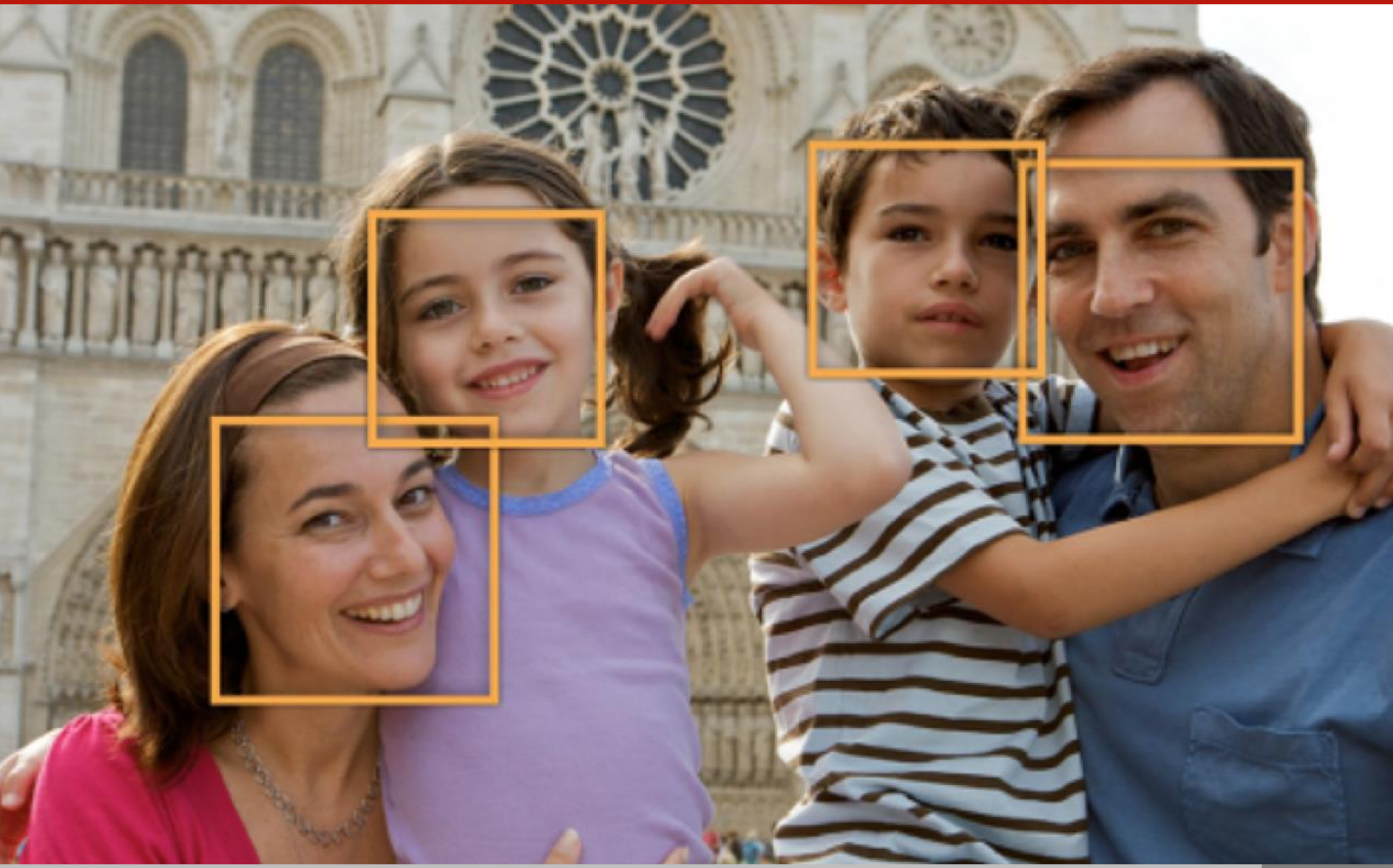
Supresión de fondo: Motion

http://www.lavrsen.dk/foswiki/bin/view/Motion/WebHome

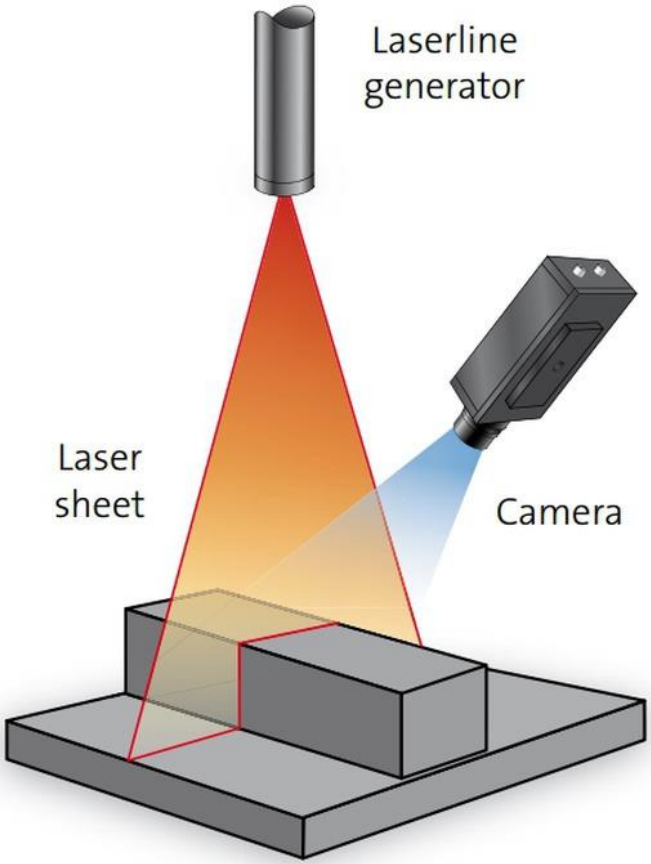
Supresión de fondo: Ejemplo

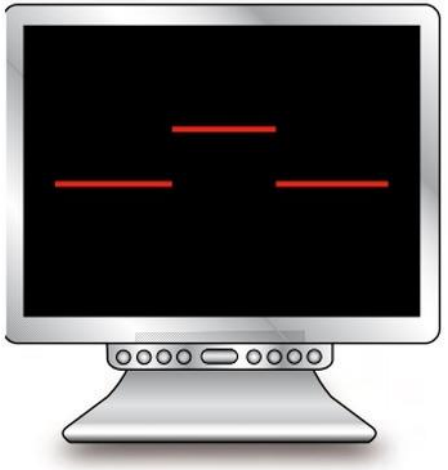
ADAS

Detección de rostros



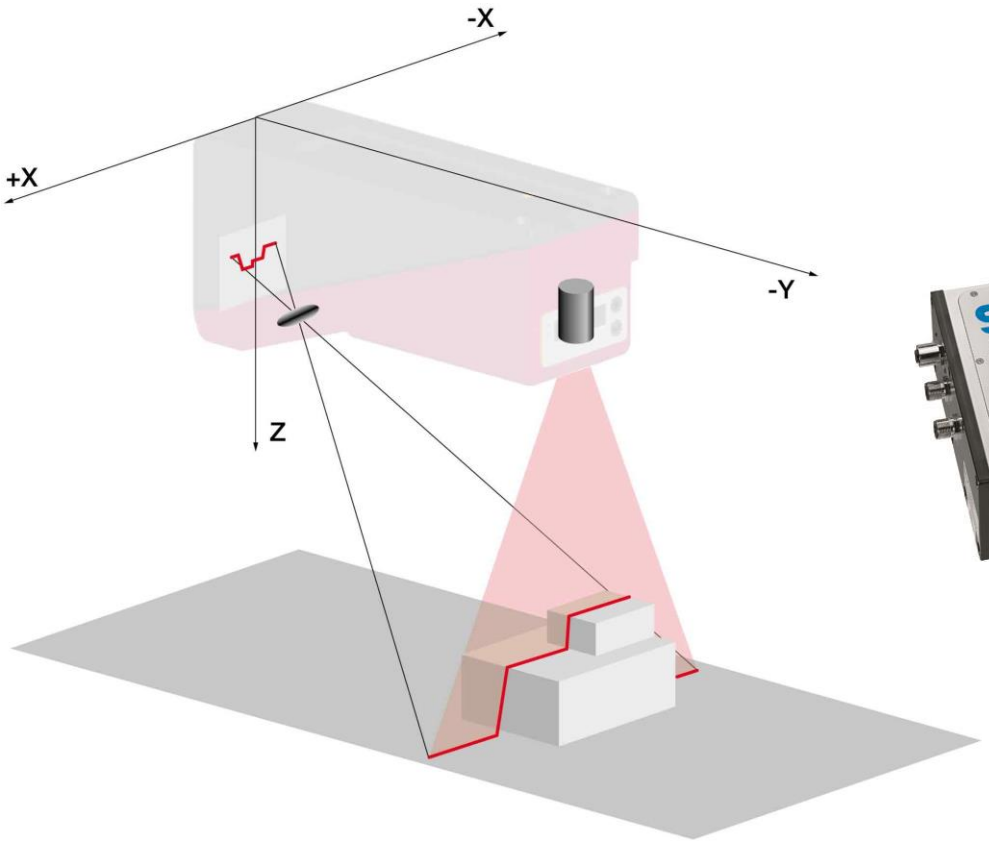
Triangulación láser

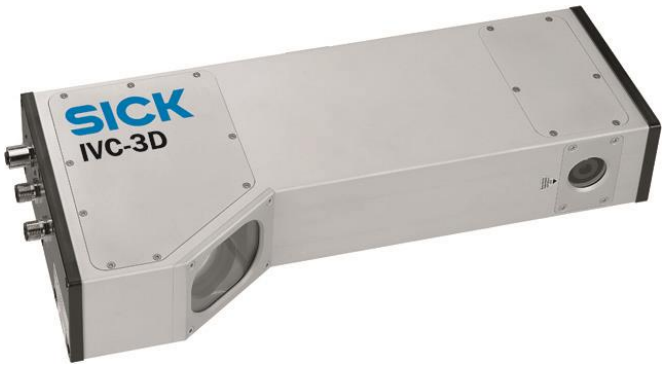




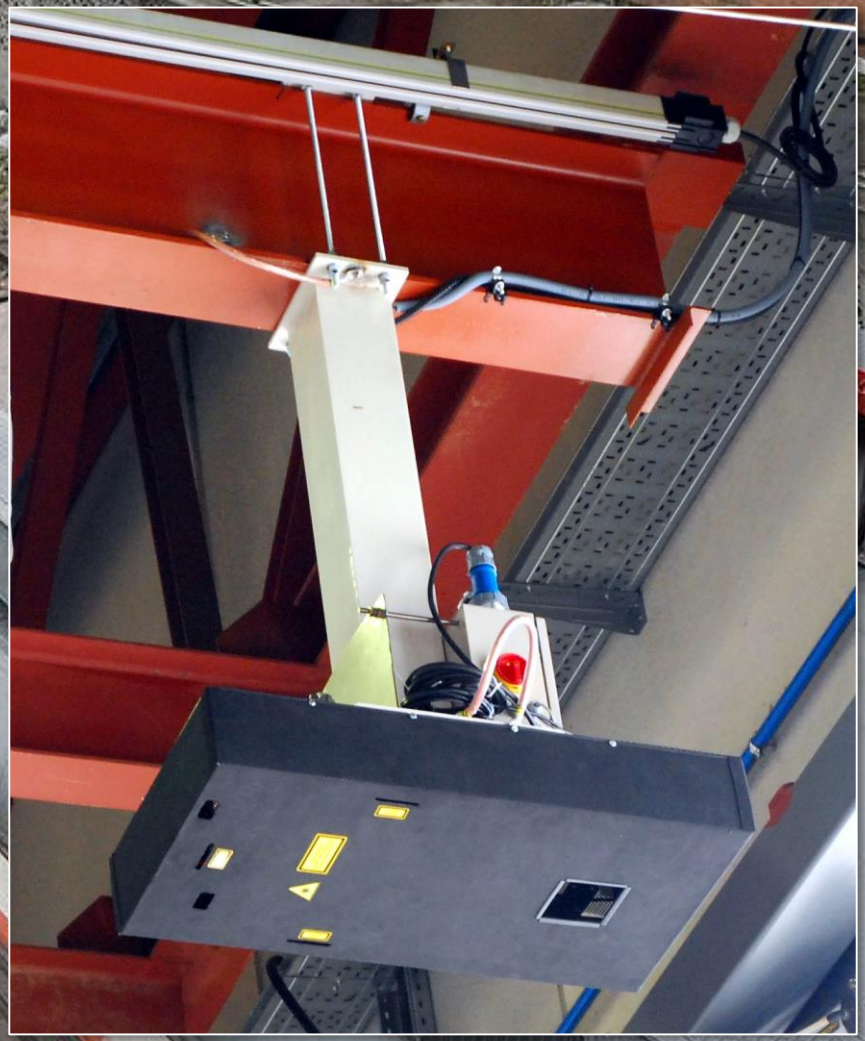
Output

Triangulación láser: aplicaciones (I)

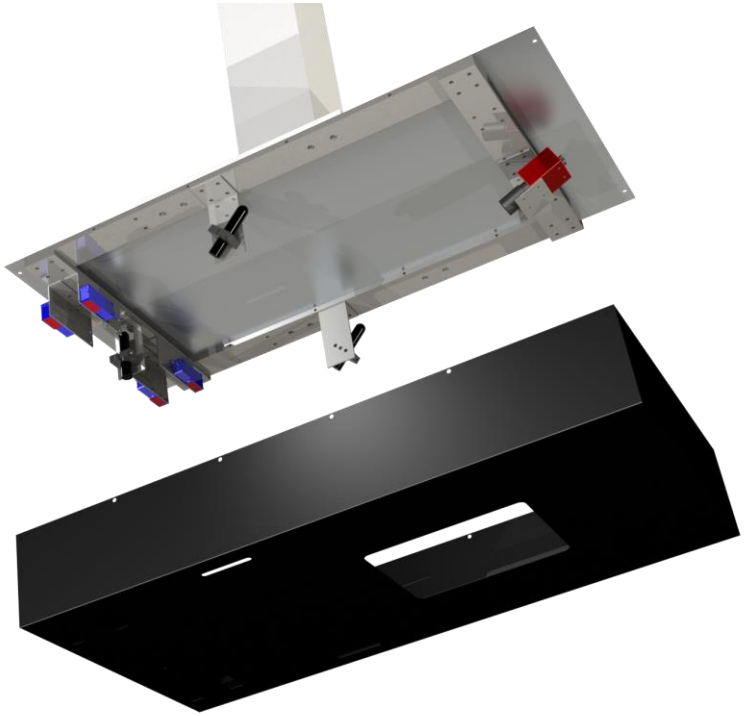


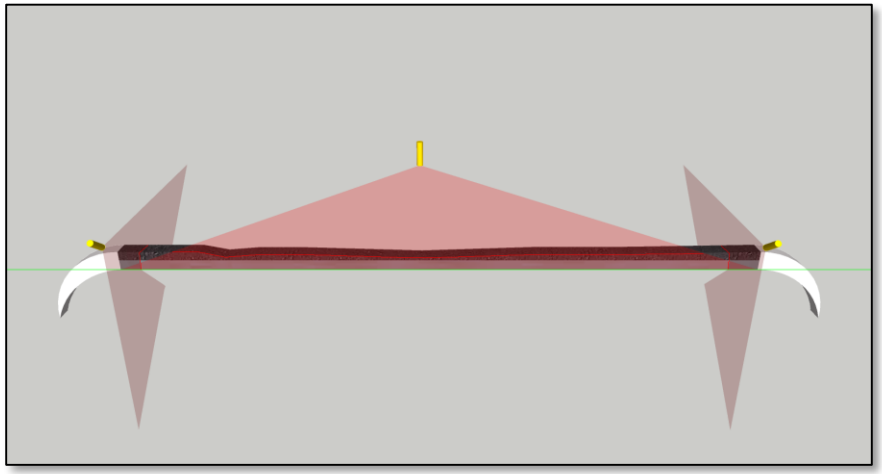


Triangulación láser: aplicaciones (II)



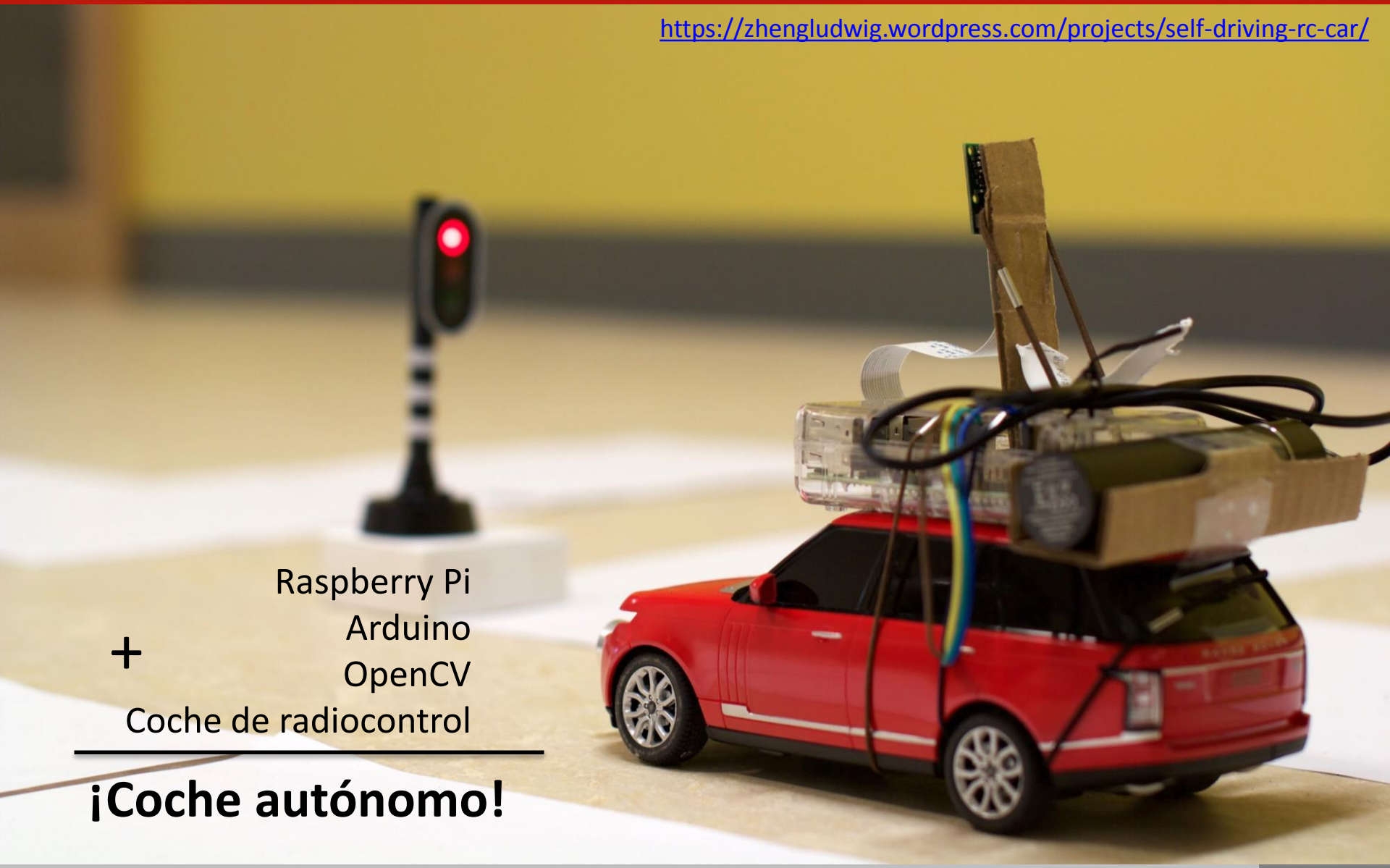
Triangulación láser: aplicaciones (y III)





Todo junto

https://zhengludwig.wordpress.com/projects/self-driving-rc-car/



Raspberry Pi
Arduino
\+ OpenCV
Coche de radiocontrol

¡Coche autónomo!

Recursos: opencv.org

OPENCV (OPEN SOURCE COMPUTER VISION)

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

QUICK LINKS:

- Online documentation
- User Q&A forum
- Report a bug
- Build farm
- Books
- Developer site

LATEST DOWNLOADS

2016-12-23

VERSION 3.2

- OpenCV for Windows
- OpenCV for Linux/Mac
- OpenCV for Android
- OpenCV for iOS

WHAT'S NEW

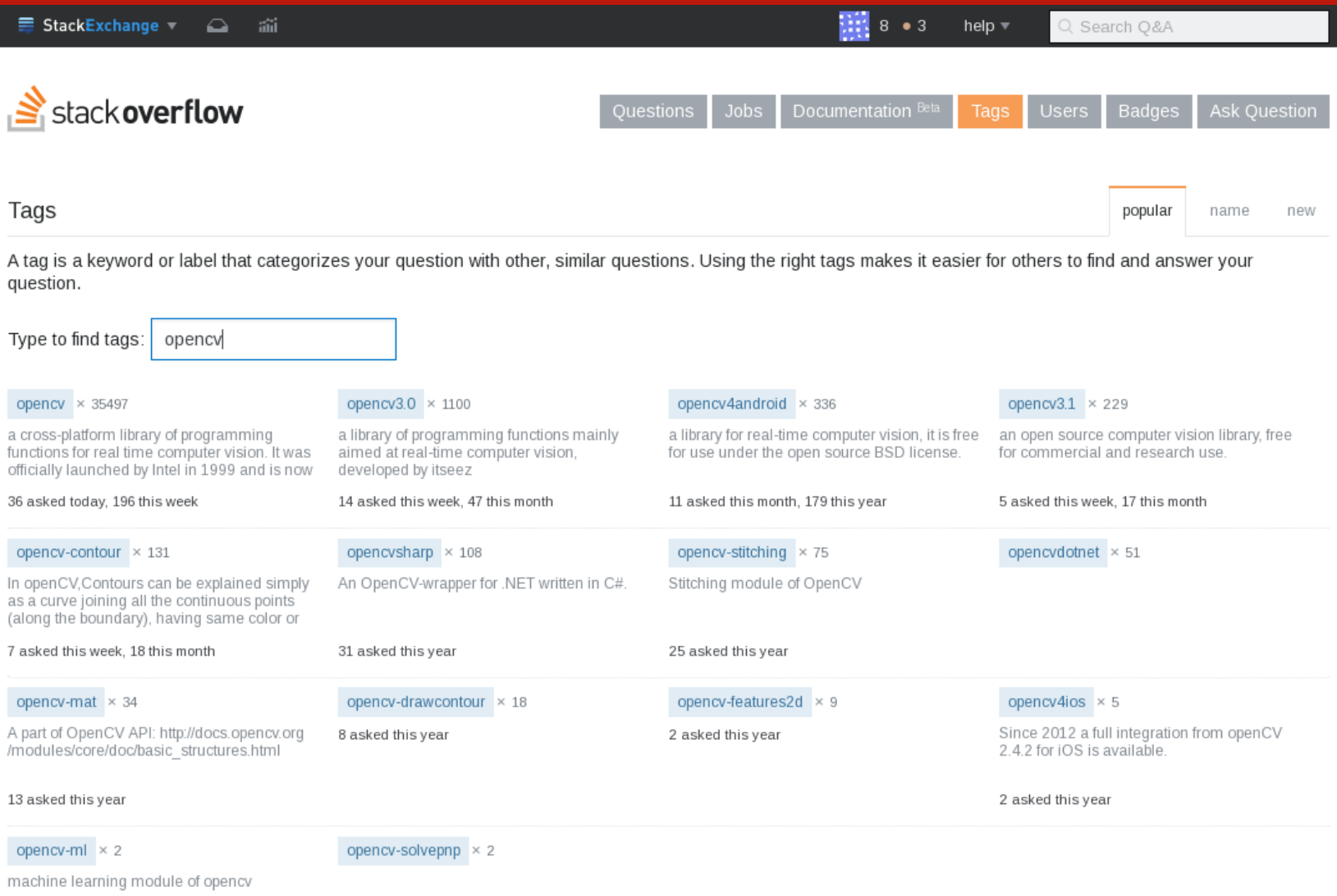
2016-12-23
OpenCV 3.2
OpenCV 3.2 is out. It's a long-awaited update to OpenCV 3.x release series, with tons of improvements and bug fixes.

2016-06-27
People's Choice Award at CVPR 2016
OpenCV will be running the People's Choice Award, sponsored by Intel, at CVPR 2016, to update and extend the OpenCV library.

2016-05-27
Intel acquires Itseez
On May 25, Intel signed a definitive agreement to acquire Itseez, Inc., an expert in Computer Vision (CV) algorithms and implementations for embedded and specialized hardware. Itseez contributes software tuning and integration in many market-leading products shipping today from cars to security systems and more. This

2016-05-11
Service outage
http://answers.opencv.org and http://code.opencv.org are down due to hardware failure. We will restore these services by the end of the week.

Recursos: stackoverflow.com



Recursos: “Learning OpenCV 3”

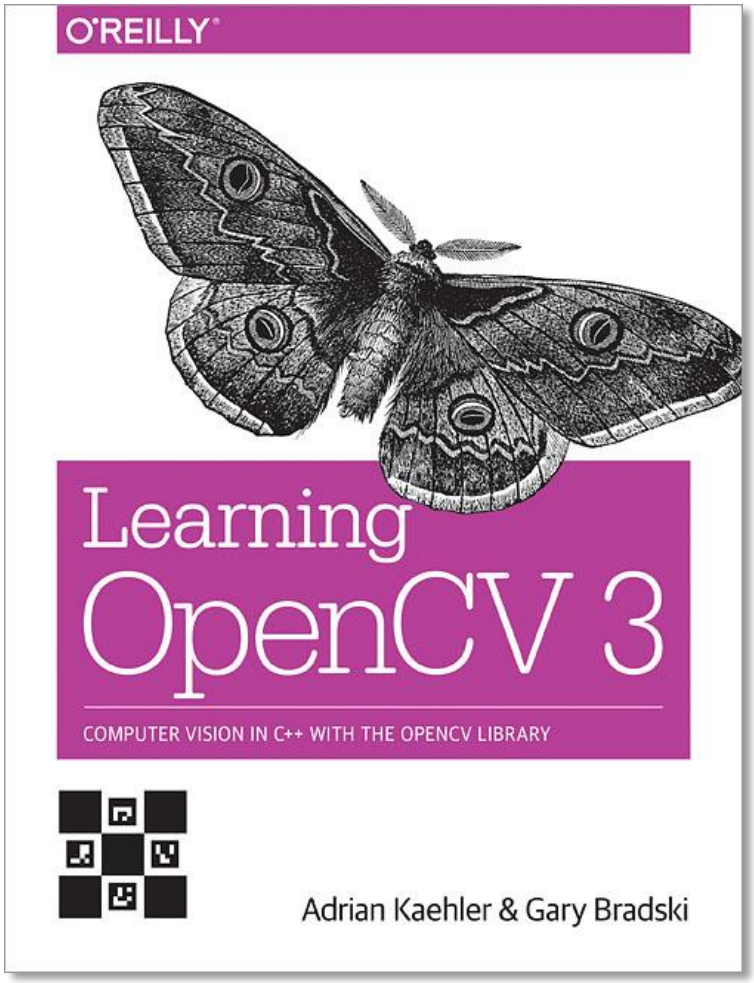
Learning OpenCV 3

Computer Vision in C++ with the OpenCV Library

Adrian Kaehler & Gary Bradsky

O’Reilly Media – 2016

ISBN: 978-1491937990



Recursos: ¡Google it!

¡Gracias por venir!

Visión artificial con

15 de febrero de 2017

Ricardo Samaniego
Director de Proyectos
ricardo.samaniego@imatia.com