HELLO WORLD

Real-Time Detection with Webcam

by Dmitry Batenkov

OpenCV is an open-source, cross-platform library for real-time computer vision.

Originally developed by Intel, the library will use Intel's Integrated Performance Primitives, if it is found on the system. It is very well-documented with a full reference manual, many examples and tutorials, and a book (which is also a good introduction to computer vision). Interfaces for C, C++, and Python are also available in OpenCV.

Example applications of the OpenCV library include human-computer interaction; object identification, segmentation and recognition; face recognition; gesture recognition; motion tracking, ego motion, motion understanding; structure from motion (SFM); stereo and multi-camera calibration and depth computation; and mobile robotics.

In this tutorial, we will learn how to do real-time face detection using a webcam. We will utilize a machine-learning object detection algorithm known as the Viola-Jones detector. It's a fast classification mechanism using Haar-like wavelet features. OpenCV ships with a very good "classifier file" for faces, but one can also train the classifier to recognize any kind of objects.

Instructions

First, download the latest OpenCV release for your platform from http://opencv.willowgarage.com and install it.

Next, copy the attached program to a file named facedetect.py. You can also download it from http://XRDS.acm.org.

In the downloaded source archive, locate the classifier file data/haarcascades/haarcascade_frontalface_alt_tree.xml and replace the placeholder in the code with this original location.

Make sure that the Python interpreter knows the location for the OpenCV Python bindings. In Linux, it should be set automatically. In Windows, set the environment variable

set pythonpath = <opencvdir>\Python2.6\Lib\site-package.

Now, connect your webcam and run the program: python facedetect.py

To exit, press Esc. Have fun!

Improvements

Once an object is detected, we can start tracking it. OpenCV has an implementation for CamShift tracking algorithm. (See the example on http://XRDS.acm.org.)

Add detection of the eyes, mouth, and so on. (OpenCV ships with corresponding classifiers.) You can recognize emotions! See the video: www.youtube.com/watch?v=V7UdYzCMKvw.

If you replace the face classifier with hands classifier, and add tracking, you can now recognize gestures!

—Dmitry Batenkov

"In this tutorial, we will learn how to do real-time face detection using a webcam. We will utilize a machine-learning object detection algorithm known as the Viola-Jones detector."

RESOURCES

Object identification links
Viola-Jones algorithm
http://www.face-rec.org/algorithms/Boosting-Ensemble/16981346.pdf

Haar training tutorial
http://note.sonots.com/SciSoftware/haartraining.html

Haar cascades repository
http://alereimondo.no-ip.org/OpenCV/34

HCI PROJECTS USING OPENCV

HandVu
Gesture recognition
www.movesinstitute.org/~kolsch/HandVu/HandVu.html

EHCI Head tracking
http://code.google.com/p/ehci

PyEyes Eyes tracking
http://eclecti.cc/olpc/pyeyes-xeyes-in-python-with-face-tracking

CCV/touchlib Multi-touch library
http://nuigroup.com

OTHER HCI/CV TOOLKITS

TUIO Common API for tangible multitouch surfaces
www.tuio.org/?software (list of implementations)

Trackmate Do-it-yourself tangible tracking system
http://trackmate.media.mit.edu

Sphinx Speech recognition toolkit
www.speech.cs.cmu.edu/sphinx/tutorial.html

VXL versatile computer vision libraries
http://vxl.sourceforge.net

Integrating Vision Toolkit
http://ivt.sourceforge.net

```
import sys
import cv

storage=cv.CreateMemStorage(0)
image_scale=1.3
haar_scale=1.2
min_neighbors=1
haar_flags=0

def detect_and_draw(img):
    # allocate temporary images
    gray=cv.CreateImage((img.width,img.height),8,1)
    small_img=cv.CreateImage((cv.Round(img.width/
    image_scale),
        cv.Round(img.height/image_scale)), 8, 1 )

    # convert color input image to grayscale
    cv.CvtColor( img, gray, cv.CV_BGR2GRAY )

    # scale input image for faster processing
    cv.Resize( gray, small_img, cv.CV_INTER_NN )
    cv.EqualizeHist( small_img, small_img )

    # start detection
    if( cascade ):
        faces=cv.HaarDetectObjects( small_img,
        cascade, storage,
            haar_scale, min_neighbors, haar_flags )
    if faces:
        for (x,y,w,h),n in faces:
        # the input to cvHaarDetectObjects was resized, so scale the
        # bounding box of each face and convert it to two CvPoints
        pt1=(int(x*image_scale),int(y*image_scale))
        pt2=(int((x+w)*image_scale),
        int((y+h)*image_scale))
        # Draw the rectangle on the image
        cv.Rectangle(img,pt1,pt2,cv.CV_RGB(255,0,0),3,8,0)
        cv.ShowImage( "result", img )

if __name__=='__main__':
    # Load the Haar cascade
    cascade_name="./haarcascade_frontalface
    alt_tree.xml"
    cascade=cv.Load(cascade_name)

    # Start capturing.Can change index if more than one
    camera present
    capture=cv.CaptureFromCAM(0)

    # Create the output window
    cv.NamedWindow("result",1)

    frame_copy=None
    while True:
        frame=cv.QueryFrame( capture )
        # make a copy of the captured frame
        if not frame_copy:
            frame_copy=cv.CreateImage((frame.
            width,frame.height),
            cv.IPL_DEPTH_8U, frame.nChannels )
            cv.Copy( frame, frame_copy )

        detect_and_draw(frame_copy)

        c=cv.WaitKey(7)
        if c==27: # Escape pressed
                break
```