

## pdp\_opencv & pix\_opencv

Open CV bindings for PureData (for PDP and GEM libraries).

Most of the code here is inspired by the Open CV examples code and/or taken from the book : "Learning Open CV : Computer Vision with the Open CV library" by Gary Bradski & Adrian Kaehler, O'Reilly.

You can find an introduction to computer vision and to this package here : pd\_opencv-0.2.pdf [[http://giss.tv/pd-opencv/pd\\_opencv-0.2.pdf](http://giss.tv/pd-opencv/pd_opencv-0.2.pdf)].

You can see a few demonstration videos made in workshops here : Baltan workshop [<http://giss.tv/dmddb/index.php?channel=pdopencv>] hackteria.org workshop [<http://giss.tv/dmddbflv/index.php?channel=giss>] alexandria OLE workshop [<http://giss.tv/dmddb/index.php?channel=pdole>].

We're encouraging people to test it and report us any problems encountered :

```
Lluis Gomez i Bigorda ( lluisgomez_at_hangar_dot_org )
```

and

```
Yves Degoyon ( ydegoyon_at_gmail_dot_com ).
```

## CONTENTS

### Filtering Modules

#### pdp\_opencv\_threshold / pix\_opencv\_threshold

Many of Open CV analysis object ( blob detection, contours detection ) operate better on binary images, this means on an image that is reduced to the pixels above or below a certain level of intensity. The object threshold applies a fixed-level threshold to frames. It produces a binary like image but still in RGB (RGBA or whatever it was before).

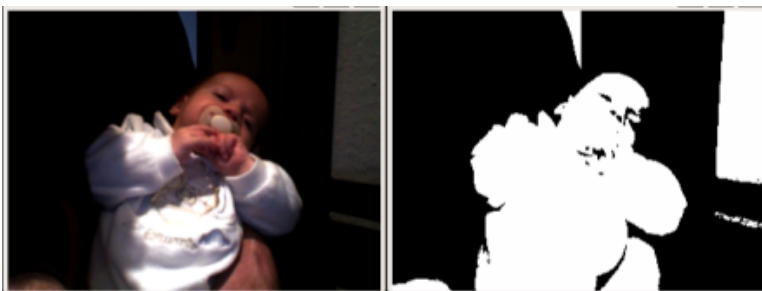
Input Parameters :

```
mode :
CV_THRESH_BINARY : maxvalue if src>threshold
CV_THRESH_BINARY_INV : maxvalue if src<threshold
CV_THRESH_TRUNC : threshold if src>threshold
CV_THRESH_TOZERO : 0 if src<threshold
CV_THRESH_TOZERO_INV : 0 if src>threshold
```

```
threshold : value of the threshold for testing pixels.
```

```
maxvalue : maxvalue used for BINARY mode, default 255.
```

Output :



#### pdp\_opencv\_athreshold / pix\_opencv\_athreshold

For some kind of images, where some zones are really dark and some zones are really bright, a fixed-level threshold might be too destructive, and for such cases, we need to use an adaptative threshold filter, where the

level of the threshold depends on the average luminosity in the block of surrounding pixels.

Input Parameters :

```

method :
CV_ADAPTIVE_THRESH_MEAN_C : use as a threshold the average value of pixels in the size x size surrounding pixels
CV_ADAPTIVE_THRESH_GAUSSIAN_C : applies a gaussian weight to the pixels around the current pixel.
Pixels in the center have a heavier weight than the ones on the outskirts of the block.

mode :
CV_THRESH_BINARY : maxvalue if src>threshold
CV_THRESH_BINARY_INV : maxvalue if src<threshold

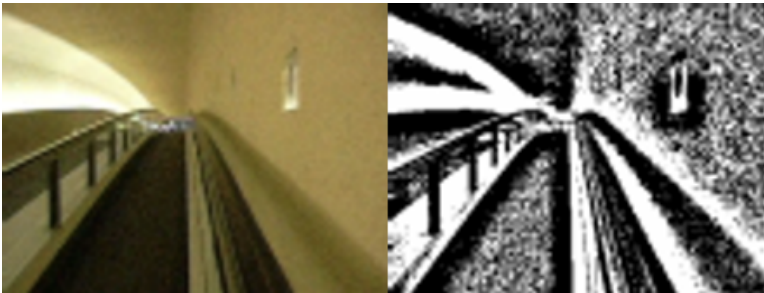
blocksize : size of the surrounding block to calculate the threshold.

dim : attenuation of the threshold, the real threshold used is the calculated threshold minus dim.

maxvalue : maxvalue used for BINARY mode, default 255.

```

Output :



### pdp\_opencv\_bgsubtract / pix\_opencv\_bgsubtract

This filter is used in motion detection when you want to distinguish moving objects from a static background.

It takes an image as a background reference when receiving a 'SET' message and then compare each incoming frame with that, using a threshold value to compare the pixels. It is useful if you want to detect objects that are in front of a static background and you want to isolate the foreground silhouettes of moving objects.

Input Parameters :

```

SET message : This is used to capture the background.

threshold : value of the threshold for testing if pixels have changed.

```

Output :



### pdp\_opencv\_bgstats / pix\_opencv\_bgstats

This object is very similar to bgsubtract but it uses a time parameter and a statistical model that makes objects considered as the foreground disappear after a while and then be considered as background. So it behaves as an adaptive background extractor.

It also performs a supplemental stage of noise filtering through internal morphology ( erosion / dilation ) and eliminates small blobs that can be considered as noise.

The output of the foreground is in binary mode.

Input Parameters :

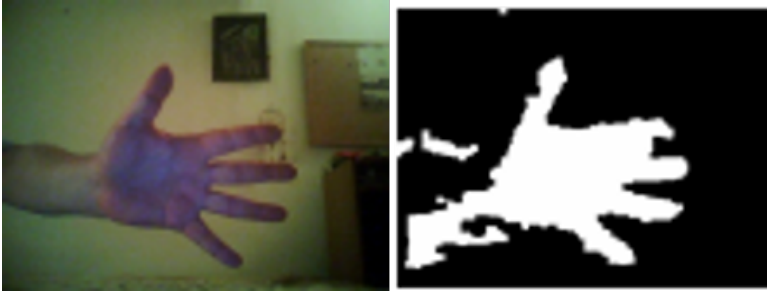
```
RESET message : Reset the background to current frame.
```

```
alpha : period of time after which a foreground object is considered as background ( default : 0.1 seconds ).
```

```
erode : number of iterations of the erode/dilate filter to eliminate noise.
```

```
minarea : minimum size of the objects shown in the foreground.
```

Output :

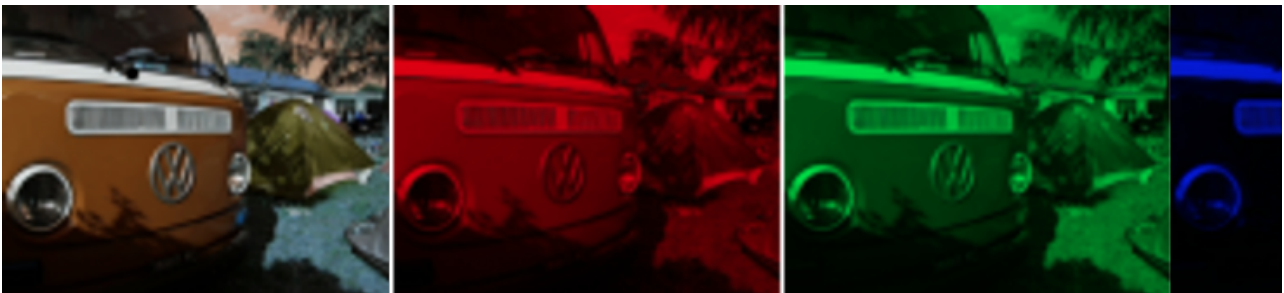


### pdp\_opencv\_channels

This object is a simple RGB channels separator that can be useful when it's necessary to detect some specific hues in a video input. It only exists in the version for PDP, in GEM you can use `pix_bitmask` instead.

Input Parameters :None.

Output :



### pdp\_opencv\_colorfilt

This filter is used to filter a certain color in the video input, isolating and identifying objects which are of this given color, using a tolerance factor for the identification of the color. The selected color can be picked from the video and set through RGB inlets.

Input Parameters :

second inlet : R component of the selected color.

```
third inlet : G component of the selected color.
```

```
fourth inlet : B component of the selected color.
```

```
tolerance : tolerance factor for the identification of the color.
```

```
pick px py : pick the color from the incoming video.
```

Output :



## Pre-processing Modules

---

These modules are useful to pre-process the image before any further analysis.

The difference with analysis modules is that they don't output any analysis data.

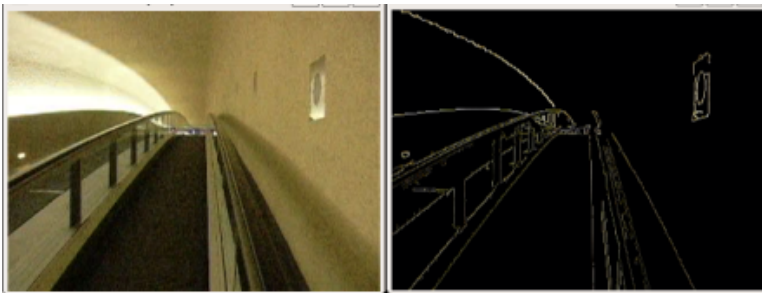
### pdp\_opencv\_edge / pix\_opencv\_edge

This filter is a classical edge detector that detects changes of pixels in the image ( using first degree gradient ), it can be useful to pre-process an image before passing it to a contour processing analysis object.

Input Parameters :

second inlet : threshold to detect edges.

Output :



### pdp\_opencv\_laplace / pix\_opencv\_laplace

This filter is also an edge detector but it uses a calculation of the second derivative known as Laplace in both directions and detects more accurately the important edges in a picture.

Input Parameters :

second inlet : aperture size ( number of points considered to calculate the derivative ).

Output :



### pdp\_opencv\_morphology / pix\_opencv\_morphology

This technique is aimed at distinguishing more precisely the connected pixels blobs in an image using the algorithms of opening/closing and erosion/dilation that increase or reduce the size of zones of bright or dark

pixels.

It works better on binary image and is helpful to join some zones in a picture that define a wider zone to be processed or tracked.

Input Parameters :

```

-----
mode : switch between open/close and dilate/erode algorithms.
-----
shape : form of the kernel that is used in the algorithm.
It can be one of : rectangle (0), elliptic (1) or cross-shaped (2).
-----
second inlet : the second inlet indicates the number of iterations that has the effect of accentuating the transfo
-----

```

Output :



pdp\_opencv\_distrans / pix\_opencv\_distrans

Another filter that is really close to morphology is the 'distance' algorithm filter that skeletizes the different forms in an image and let's you track a simplified form.

It works better on binary images and is particularly appropriate to simplified silhouettes.

Input Parameters :

```

-----
type (0/3/5) : optionally uses a mask in the transformation. 0 is none.
-----
second inlet : threshold to detect edges.
-----
voronoi (0/1) : activates/desactivates the voronoi triangles partitioning.
-----

```

Output :



pdp\_opencv\_dft / pix\_opencv\_dft

This filter is very useful in image analysis as it transform an image from its spacial representation to a frequency domain representation, modeling an image as an infinite combination of sinusoidal waves. This is know as the Discrete Fourier Transform, very well known of people working in sound processing and it can be useful also in image analysis. It first binarize the incoming images and then calculates the Fourier magnitude and phase diagrams. As it is quite greedy of ressources, it only process a frame when it receives a 'bang' message.

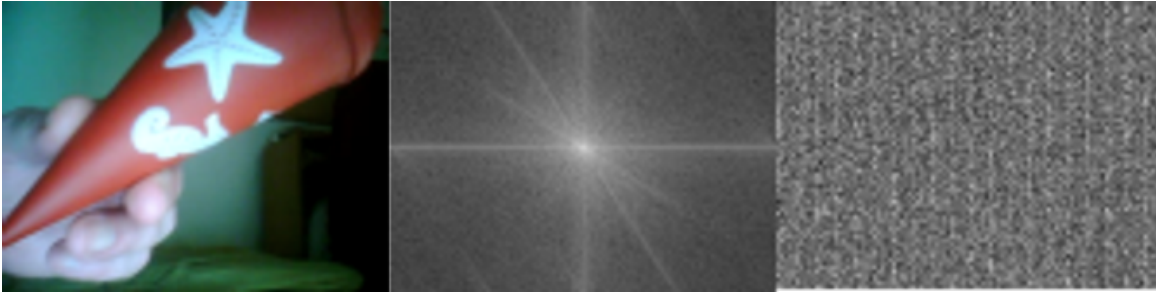
Input Parameters :

```

-----
bang message : triggers the calculation of a DFT.
-----

```

Output :



## Analysis Modules

These modules are the last stage in a processing chain and try to extract significant analysis data from the video stream.

### pdp\_opencv\_floodfill / pix\_opencv\_floodfill

This module marks some pixel zones that identifies some blobs, e.g zones of adjacent pixels of almost the same intensity and color, taking into account a tolerance in the variation of pixels. It also repaints these zones in a specific color.

Input Parameters :

second inlet : lower tolerance for pixels.

third inlet : upper tolerance for pixels.

connectivity : use 4-points or 8-points connectivity mode.

color : activates/desactivates color mode.

mark x y : mark a blob containing this pixel.

delete n : delete a marker.

clear : delete all markers.

fillcolor n r g b : set the fill color for blob n.

Output :

second outlet : position of each marked blobs in the form 'number x y width height'.



### pdp\_opencv\_contours\_boundingrect / pix\_opencv\_contours\_boundingrect

This object calculates the up-right bounding rectangle of all contours of an image. This object works better on binary images distinguishing white pixels zones on a black background.

Input Parameters :

```

mode : contour detection mode ( see cvFindContours [1] ).
method : contour detection method ( see cvFindContours [1] ).
second inlet : minimal size of a contour.
third inlet : maximal size of a contour.
maxmove : maximum movement of a contour between 2 frames
( used for identification/numbering of contours ).
ftolerance : nnumber of frames where a contour can disappear without being considered as lost ( used for identific
nightmode 0/1 : only shows the detected contours.
draw 0/1 : draw the contours.
show 0/1 : show the bounding rectangle or not.

```

**Output :**

```

second outlet : position of each contour in the form 'number x y width height'
third outlet : number of detected contours.

```

**pdp\_opencv\_contours\_convexity / pix\_opencv\_contours\_convexity**

This object looks for the convexity curves of the biggest contour of an image.

Each convexity curve is defined by three points : the starting point, the ending point and the depth point. This object works better on binary images.

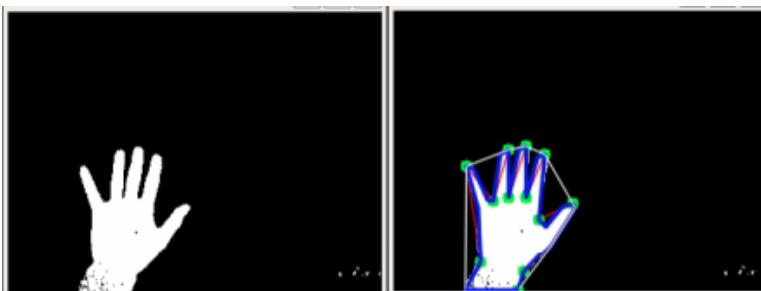
Input Parameters : none.

**Output :**

```

second outlet : the number of convexity curves of the biggest contour.
third outlet : the three points defining a convexity curve in the form : 'number xstart ystart xdepth ydepth xend

```

**pdp\_opencv\_contours\_convexhull / pix\_opencv\_contours\_convexhull**

This object draws the convex hull of the biggest contour found in a video input. A convex hull is a polygon that

approximates the contour.

#### Input Parameters :

```
accuracy : polygon approximation accuracy.
```

```
mode : contours retrieval mode.
```

```
method : contours retrieval method.
```

```
nightmode (0/1) : show/hide original video.
```

#### Output :

```
second outlet : the number of points in the convex hull.
```

```
third outlet : the list of the points in the convex hull as a list of coordinates : x1 y1 x2 y2 ... xn yn.
```



### pdp\_opencv\_lk / pix\_opencv\_lk

This object detects remarkable points in a contour, using Shi and Tomasi point detection algorithm, enabling to mark peculiar points like angles and asperities.

You can then choose the points you want to track by clicking on them or select all of them for tracking, using Lukas-Kanade ( hence lk ) tracking technique.

#### Input Parameters :

```
init : re-calculates contours and most significant points.
```

```
mindistance : minimal distance between points.
```

```
quality : quality factor used in approximation. This behaves like a threshold on the gradient of remarkable points
```

```
mark x y : mark a point to be tracked.
```

```
mark all : mark all points.
```

```
mark none : unmark all points.
```

```
delete n : delete a marker.
```

```
clear : delete all markers.
```

```
maxmove n : maximum displacement of a point between two frames ( used for identification / numbering ).
```

```
ftolerance n : number of frames where a point can disappear ( used for identification / numbering ).
```

```
delaunay on : draw a delaunay triangulation between all points.
```

```
delaunay off : deactivate the delaunay.
```

```
pdeleunay <n> <thresgold> : draw a delaunay containing the point n and all points which difference with point n is
```

```
nightmode (0/1) : show/hide original video frame.
```

```
second inlet : window size for calculating the gradient.
```

#### Output :

```
second outlet : for each marked point, point position in the form : 'number x y'.
```



### pdp\_opencv\_surf / pix\_opencv\_surf

This object is very similar to Lukas-Kanade point tracking technique, but it uses the SURF ( Speed Up Robust Features ) to detect remarkable points. It generally detects more points than the lk detection algorithms.

This object seems to be available only with Open CV 1.1.0, it will not work with Open CV 1.0.0.

#### Input Parameters :

```
hessian : hessian threshold for the detection of points. This influences the number of detected points.
```

```
mark x y : mark a point to be tracked.
```

```
mark all : mark all points.
```

```
mark none : unmark all points.
```

```
delete n : delete a marker.
```

```
clear : delete all markers.
```

```
maxmove n : maximum displacement of a point between two frames ( used for identification / numbering ).
```

```
ftolerance n : number of frames where a point can disappear ( used for identification / numbering ).
```

```
delaunay on : draw a delaunay triangulation between all points.
```

```
delaunay off : disactivate the delaunay.
```

```
pde-launay <n> <threshold> : draw a delaunay containing the point n and all points which difference with point n is less than threshold.
```

```
nightmode (0/1) : show/hide original video frame.
```

#### Output :

```
second outlet : for each marked point, point position in the form : 'number x y'.
```



### pdp\_opencv\_motempl / pix\_opencv\_motempl

This object tracks movement of detected objects using the history of motion on a variable number of frames. It operates internally with binary images, thresholding the incoming frames and build an history of image changes with timestamps.

#### Input Parameters :

|   |
|---|
| second inlet : value of threshold to distinguish blobs.   |
| third inlet : minimal size of a detected blob.  |
| fourth inlet : maximal size of a detected blob.   |
| frame_buffer_size : number of frames in the motion history ( default 4 ).                       |
| max_time_delta : maximum time of a pixel change in the motion history for detecting a movement. |
| min_time_delta : minimum time of a pixel change in the motion history for detecting a movement. |
| aperture ( 1   3   5   7 ) : aperture size for the calculated derivative.                       |
| mhi_duration : duration of displaying the detected motions.                                     |

#### Output :

|  |
|--|
| second outlet : for each blob detected, the position, the size of the blob and the direction of its movement in! |
|--|



### pdp\_opencv\_hist\_compare / pix\_opencv\_hist\_compare

This object memorize and draw the statistical composition of an image in the form of an histogram of the hue and saturation of the pixels. Therefore, it works only on color frames.

You can record up to 80 histograms that defines the composition of the image and by comparing them to the actual frame entering, you can recognize specific configuration of the input image if it has been memorized before.

This can work quite well to recognize special gestures like from a human hand and can be used in a kind of gesture command to a pd patch.

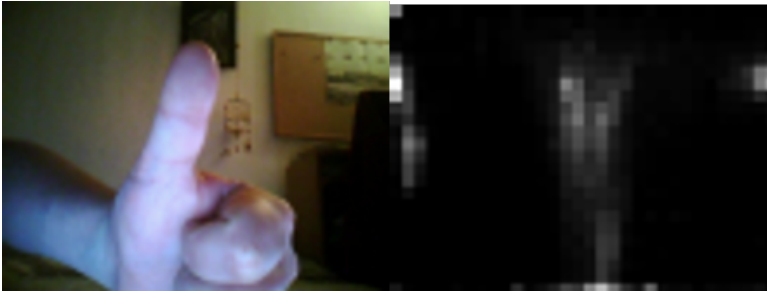
#### Input Parameters :

|  |
|--|
| second inlet : send there a number to memorize a specific histogram. |
|--|

## Output :

```
second outlet : the closest stored histogram to the actual entering frame.
```

```
third outlet : all measured distances with presently stored histograms in the form 'd1 d2 d3...dn'.
```



## pdp\_opencv\_harcascade / pix\_opencv\_harcascade

This object loads a Haar's cascade decision tree in the form of an XML file, that is based on Haar's technique of dividing an image in squares and calculate some pixels sums on these squares, and the differences between them.

Using this technique, we can detect some peculiar form in an image ( face, mouth, eyes, ... ), but this needs to be trained to have the right coefficients stored in the XML with a great number of sample images ( > 10.000 ). Open CV provides a tool to create such a decision tree, but the task is fastidious and not easy.

There are various haar's cascade available on the web like this collection : <http://alereimondo.no-ip.org/OpenCV/34> [<http://alereimondo.no-ip.org/OpenCV/34>].

## Input Parameters :

```
load <xml file> : load an XML Haar's cascade decision tree.
```

```
mode (0/1) : use edge detection or not.
```

```
min_size : minimal size of a detected object.
```

```
min_neighbours : minimum number (minus 1) of neighbour rectangles that makes up an object ( default 2 ).
```

```
scale_factor : scale applied for block size.
```

```
ftolerance : number of frames where an object can temporarily disappear ( used for identification / numbering ).
```

```
clear : clear all objects markers.
```

## Output :

```
second outlet : number of detected objects.
```

```
third outlet : coordinates of each object identified by a circle under the form 'number x y radius'.
```



## pdp\_opencv\_camshift / pix\_opencv\_camshift

This object implements the Continuously Adaptive Mean-shift tracking algorithm (CAM).

You can select an object with the mouse and it starts tracking the object from a rectangular zone around this point, but adapting the zone at each iteration.

It gives also the orientation of the object. It's working in the HSV colorspace, distinguishing the hue from other fields S, V.

Input Parameters :

```

-----
vmin : V pre-filtering minimal value.
-----
vmax : V pre-filtering maximal value.
-----
smin : S pre-filtering minimal value.
-----
backproject (0/1) : show the backproject ( binary image of selected pixels ).
-----
rwidth : initial width of the tracking zone.
-----
rheight : initial height of the tracking zone.
-----
track x y : track an object containing this point.
-----

```

Output :

```

-----
second outlet : coordinates, size and orientation of the track object ( only one ) in the form : 'x y width height
-----

```



## pdp\_opencv\_hough\_lines / pix\_opencv\_hough\_lines

This object detects segments of lines in the incoming frames using the Hough detection algorithm.

Input Parameters :

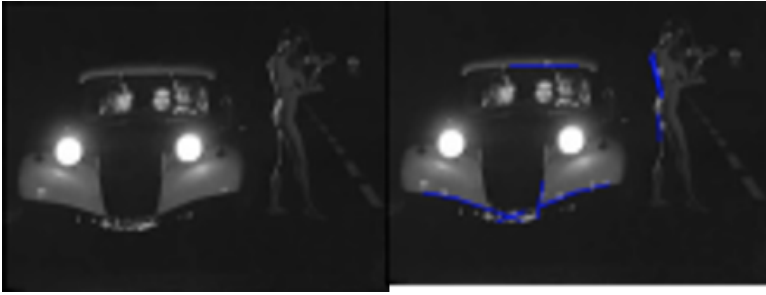
```

-----
mode : detection mode used for the algorithm, one of :
CV_HOUGH_STANDARD : full lines.
CV_HOUGH_PROBABILISTIC : line segments.
CV_HOUGH_MULTI_SCALE : line segments.
-----
maxlines : maximum number of detected lines.
-----
threshold : difference threshold for detection of edges.
-----
minlength : minimal length of a detected segment.
-----
gap : gap between lines ( for mode CV_HOUGH_PROBABILISTIC )
-----
aresolution : angle resolution ( for mode CV_HOUGH_MULTI_SCALE )
-----
dresolution : distance resolution ( for mode CV_HOUGH_MULTI_SCALE )
-----
nightmode (0/1) : show/hide the incoming video frame.
-----

```

Output :

```
second outlet : for each detected line, coordinates of starting and ending points in the form : 'number xstart ystart xend yend'
```



### pdp\_opencv\_hough\_circles / pix\_opencv\_hough\_circles

This object detects segments of circles in the incoming frames using the Hough detection algorithm.

Most of the time, the circles do not really exist but are 'almost there'.

Input Parameters :

```
maxcircles : maximum number of detected circles.
```

```
threshold : threshold used in the edges detection ( cvCanny ).
```

```
threshold2 : threshold used for the detection of the centers.
```

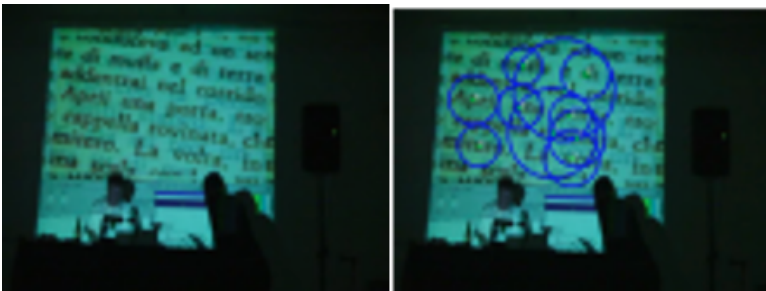
```
mindist : minimum distance between circles centers.
```

```
resolution : resolution of the accumulator.
```

```
nightmode (0/1) : show/hide the incoming video frame.
```

Output :

```
second outlet : for each detected circles, coordinates of the center and radius in the form : 'number xcenter ycenter radius'
```



### pdp\_opencv\_hu\_compare / pix\_opencv\_hu\_compare

This object takes two inputs, generally an incoming video stream and a pattern image, also in form of a video stream and tries to detect the pattern in the incoming video frames.

It operates as follows : it extracts from the pattern image the biggest contour and compare the contours of the incoming video using cvMatchShapes, which compares contours calculating their Hu moments, characteristics of the topology of the contours.

Input Parameters :

```
minsize : minimal size of threated contours, to remove small shapes and noise.
```

```
method : method used in cvMatchShapes which can be one of :
```

```
CV_CONTOURS_MATCH_I1, CV_CONTOURS_MATCH_I2 or CV_CONTOURS_MATCH_I3 : the difference between the Hu moments are compared
```

```
clear : recalculate the contour of the pattern image if ever it changes.
```

```
criteria : minimal distance under which the contours are considered as detected and highlighted.
```

#### Output :

```
first outlet : the contours of the incoming video stream with detected contours highlighted.
```

```
second outlet : the pattern image contour.
```

```
third outlet : the minimum distance between the contours of the incoming stream with the pattern image.
```

```
fourth outlet : the coordinates of detected matching contours.
```



note : as the contours are all normalized before the comparisons, the size and orientation of the contours should not impair the detection, but from the tests we've been running, this is more relevant with the next module that compares contours using a PGH histogram.

#### pdp\_opencv\_pgh\_compare / pix\_opencv\_pgh\_compare

This object is very similar to the previous one, except that for comparing contours, it uses a PGH histogram instead of the Hu moments.

The PGH histogram is an histogram of the convexities of the contours storing the size and angle of each curve. This method is statistical and is better for detecting contours regardless of their size and orientation.

#### Input Parameters :

```
minsize : minimal size of threated contours, to remove small shapes and noise.
```

```
clear : recalculate the contour of the pattern image if ever it changes.
```

```
criteria : minimal distance under which the contours are considered as detected and highlighted.
```

#### Output :

```
first outlet : the contours of the incoming video stream with detected contours highlighted.
```

```
second outlet : the pattern image contour.
```

```
third outlet : the minimum distance between the contours of the incoming stream with the pattern image.
```

```
fourth outlet : the coordinates of detected matching contours.
```



note : this object is less sensitive to the change of size and orientation, but as it is statistical, it can also confuse

a contour for another that would include the same number of curves with the same angle.

### pdp\_opencv\_knear / pix\_opencv\_knear

This object compares the incoming video frames with a collection of samples loaded from a directory.

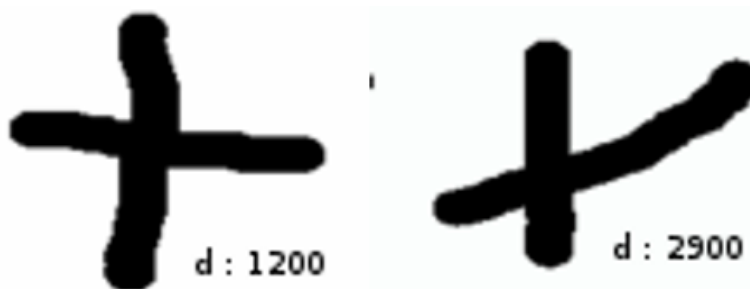
The idea is to implement a kind of form recognition close to some OCR techniques that recognizes incoming characters calculating the distance from the current input with all the samples of a database. The calculated distance just calculates the number of pixels that are different after binarizing them.

Input Parameters :

```
load <directory> <count> : load all the samples from a database. The directory must contain files named : 000.png
bang : calculates the distance of the incoming video frame with the database.
```

Output :

```
second outlet : the distance of the incoming image with the database.
```



Here the database is a database of '+' signs and a distance smaller than 2000 detects a more-or-less well-formed sign but this threshold ( 2000 ) depends on the size of the samples and on the tolerance you want to give to the algorithm, so it must be adapted to each particular case.

### pdp\_opencv\_of\_bm / pix\_opencv\_of\_bm

This object, as the next two following ones, calculates the optical flow, trying to correlate the movement of blocks from a frame to next one, this algorithm is known as Block Matching. It uses a variable size of blocks that can be adjusted.

Input Parameters :

```
blocksize <width> <height> : sets the size of the blocks.
shiftsize <width> <height> : sets the coordinate increment between blocks ( blocks can be overlapping ).
maxrange <width> <height> : sets the size of the pixels scanned around a block ( the actual region of search ).
threshold <value> : sets the threshold of movement for a single block.
minblocks <value> : sets the minimum number of blocks that moved over which a global movement is detected.
useprevious <0/1> : use previous results of movements to calculate the new one ( cumulative movements ).
nightmode <0/1> : show or hide the original video.
```

Output :

```
second outlet : the average angle of the blocks movement.
third outlet : the maximum movement of a block with its amplitude and its angle in the form : <amplitude> <angle>
```



### pdp\_opencv\_of\_hs / pix\_opencv\_of\_hs

This object is very similar to the previous one but does not operate on blocks, rather on each individual pixel. It uses the Horn and Schunck algorithm for the identification of pixels.

#### Input Parameters :

```

-----
lambda <value> : sets the value of the Lagrange multiplier.
-----
threshold <value> : sets the threshold of movement for a single piksel.
-----
minblocks <value> : sets the minimum number of blocks of one pixel that moved over which a global movement is detected.
-----
useprevious <0/1> : use previous results of movements to calculate the new one ( cumulative movements ).
-----
nightmode <0/1> : show or hide the original video.
-----

```

#### Output :

```

-----
second outlet : the average angle of the blocks movement.
-----
third outlet : the maximum movement of a block with its amplitude and its angle in the form : <amplitude> <angle>
-----

```



### pdp\_opencv\_of\_lk / pix\_opencv\_of\_lk

This object is almost the same as the previous one, except that it uses the Lucas/Kanade algorithm for the identification and tracking of points.

#### Input Parameters :

```

-----
winsize <width> <height> : size of the averaging window to group pixels.
-----
threshold <value> : sets the threshold of movement for a single piksel.
-----
minblocks <value> : sets the minimum number of blocks of one pixel that moved over which a global movement is detected.
-----
nightmode <0/1> : show or hide the original video.
-----

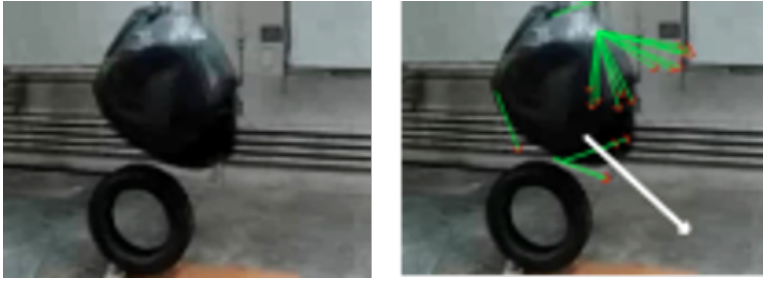
```

#### Output :

```

-----
second outlet : the average angle of the blocks movement.
-----
third outlet : the maximum movement of a block with its amplitude and its angle in the form : <amplitude> <angle>
-----

```



## DOWNLOAD & INSTALL

---

Pdopencv is now version 0.2-rc6 ( july 2010 – codename DONTATPAUL ).

### GNU/Linux

---

You have the option of installing binary packages or to compile it if you have a special need for it, for example if there are no binary package for your system : gentoo, fedora, ...

### Packages for Ubuntu and Debian

- First install Pd-Extended 0.41.4 from the main repository : Pd-extended [<http://puredata.info/downloads>].

\* For most of the current debian-based systems, pdopencv binary packages are available here : <http://giss.tv/pd-opencv/pkg> [<http://giss.tv/pd-opencv/pkg>].

```

* Just click on the package that corresponds to your system, with the right system version and the kind of CPU you
( for example for hardy 64 bits use package : pdopencv-0.2.1-20100211-hardy-amd64.deb ), then open the package with

```

next time you start pd, the library should appear in the help browser.

### Install from sources

(actually only tested in GNU/Linux Ubuntu Gutsy)

- first install opencv development packages, on ubuntu :

```

apt-get install libcv-dev
apt-get install libcvaux-dev
apt-get install libhighgui-dev
* then get the _SOURCES_ of the pd you are using and of the PDP or GEM that you are loading ( it might not work if

```

- download the sources of openCV for PDP [pdp\\_opencv-0.2-rc6.tar.gz](http://giss.tv/pd-opencv/pdp_opencv-0.2-rc6.tar.gz) [[http://giss.tv/pd-opencv/pdp\\_opencv-0.2-rc6.tar.gz](http://giss.tv/pd-opencv/pdp_opencv-0.2-rc6.tar.gz)] or for Gem [pix\\_opencv-0.2-rc6.tar.gz](http://giss.tv/pd-opencv/pix_opencv-0.2-rc6.tar.gz) [[http://giss.tv/pd-opencv/pix\\_opencv-0.2-rc6.tar.gz](http://giss.tv/pd-opencv/pix_opencv-0.2-rc6.tar.gz)],

then compile the externals :

- unpack it ::

```

tar xzvf pix_opencv-0.2-rc6.tar.gz

```

```

or

```

```

tar xzvf pdp_opencv-0.2-rc6.tar.gz

```

- cd into the library folder ::

```

cd pix_opencv

```

```

or

```

```
cd pdp_opencv
```

- configure the package with the appropriate command for your system :

```
./configure --with-pd=<path to pd_sources_> --with-gem=<path to gem_sources_>
```

for example :

```
./configure --with-pd=/usr/local/pd --with-gem=/usr/local/pd/gem
```

or

```
./configure --with-pd=<path to pd_sources_> --with-pdp=<path to pdp_sources_>
```

- then, compile it ::

```
make clean
make
```

- and finally copy the .pd\_linux to your externals folder ::

```
cp *.pd_linux /usr/lib/pd/extra/
cp *.pd /usr/lib/pd/doc/5.reference
```

next time you start pd, the library should appear in the help browser.

## MAC OSX ( macintel or powerpc )

---

### Binary package

- install Pd-extended from the main repository: <http://puredata.info/downloads> [<http://puredata.info/downloads>]
- download: OpenCV-Private-Framework-1.2.dmg [<http://www.ient.rwth-aachen.de/~asbach/OpenCV-Private-Framework-1.2.dmg>] from <http://www.ient.rwth-aachen.de/cms/software/opencv/> [<http://www.ient.rwth-aachen.de/cms/software/opencv/>]
- open it and drop the **OpenCV.framework** into your **/Library/Frameworks/** folder
- pix\_opencv for Mac OSX 10.6 and Pd-extended 0.42.5 : `pix_opencv-0.2rc7_macosx-10.6_bins.tar`
- pix\_opencv for Mac OSX 10.6 and Pd-extended 0.41.4 : `pix_opencv-0.2rc5_macosx-10.6.3_bins.tgz`
- pix\_opencv for Mac OSX 10.5, download `pix_opencv-0.2rc4_macosx-10.5.1_bins.tgz`
- unpack the archive
- open a terminal and cd to the directory :

```
cd /Users/<user>/Desktop/pix_opencv
( where <user> is the name of your user )
```

- copy the .pd\_darwin to your externals folder ::

```
sudo cp *.pd_darwin /Applications/Pd-0.4*.*-extended.app/Contents/Resources/extra/
( it will ask you your password, enter your user password )
```

- and finally copy all the documentation .pd files to your documentation folder ::

```
sudo cp *.pd /Applications/Pd-0.4*.*-extended.app/Contents/Resources/doc/5.reference
```

next time you start pd, the library should appear in the help browser.

### Install from sources

(Although pdp\_opencv could work on OSX, we recommend to use the version for GEM for performance reasons : pix\_opencv )

first install openCV MacOS framework :

- download: OpenCV-Private-Framework-1.2.dmg [http://www.ient.rwth-aachen.de/~asbach/OpenCV-Private-Framework-1.2.dmg] from http://www.ient.rwth-aachen.de/cms/software/opencv/ [http://www.ient.rwth-aachen.de/cms/software/opencv/]
- open it and drop the OpenCV.framework into your /Library/Frameworks/ directory
- get the \_SOURCES\_ of the pd you are using and of the GEM that you are loading ( it might not work if the version is different )
- download the sources of openCV for Gem pix\_opencv-0.2-rc6.tar.gz [http://giss.tv/pd-opencv/pix\_opencv-0.2-rc6.tar.gz],

then compile the externals :

- unpack it ::

```
tar xzvf pix_opencv-0.2-rc6.tar.gz
```

- cd into the package folder ::

```
cd pix_opencv
```

- configure the package with the appropriate command for your system :

```
./configure --with-pd=<path to pd _sources_> --with-gem=<path to gem _sources_>
```

for example :

```
./configure --with-pd=/usr/local/pd --with-gem=/usr/local/pd/gem
```

- then, compile it ::

```
make clean
make
```

- copy the .pd\_darwin to your externals folder ::

```
cp *.pd_darwin /Applications/Pd-*.app/Contents/Resources/extra/
```

- and finally copy all the documentation .pd files to your documentation folder ::

```
cp *.pd_darwin /Applications/Pd-*.app/Contents/Resources/doc/5.reference
```

( of course change the path to the pd version you are using ).

next time you start pd, the library should appear in the help browser.

## IMPORTANT TIPS & KNOWN BUGS

---

In order to make the pix\_opencv objects work You always had to set colorspace RGBA. otherwise, You will not see any effect.

## FOR DEVELOPPERS

---

Here you can find some examples of commented code to help you programming with Open CV in pd.

**pdp\_opencv\_threshold & pix\_opencv\_threshold**

**pdp\_opencv\_edge & pix\_opencv\_edge**

(from pdp\_opencv\_edge\_process\_rgb) ::

```

// We make here a copy of the PDP packet in image->imageData ...
// image is a IplImage struct, the default image type in openCV, take a look on the IplImage data structure here
// http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/opencv-intro.html
memcpy( x->image->imageData, data, x->x_size*3 );

// Convert to grayscale
cvCvtColor(x->image, x->gray, CV_BGR2GRAY);

cvSmooth( x->gray, x->edge, CV_BLUR, 3, 3, 0, 0 );
cvNot( x->gray, x->edge );

// Run the edge detector on grayscale
cvCanny(x->gray, x->edge, (float)x->edge_thresh, (float)x->edge_thresh*3, 3);

cvZero( x->cedge );

// copy edge points
cvCopy( x->image, x->cedge, x->edge );

//memory copy again, now from x->cedge->imageData to the new data pdp packet
memcpy( newdata, x->cedge->imageData, x->x_size*3 );_threshold**

```

(from pix\_opencv\_edge :: processRGBAImage)

```

// Here we make a copy of the pixel data from image to orig->imageData
// orig is a IplImage struct, the default image type in openCV, take a look on the IplImage data structure here,
// http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/opencv-intro.html
memcpy( orig->imageData, image.data, image.xsize*image.ysize*4 );

// Convert to grayscale
cvCvtColor(orig, gray, CV_BGRA2GRAY);

cvSmooth( gray, edge, CV_BLUR, 3, 3, 0, 0 );
cvNot( gray, edge );

// Run the edge detector on grayscale
cvCanny(gray, edge, (float)this->edge_thresh, (float)this->edge_thresh*3, 3);

cvZero( cedge );
// copy edge points
cvCopy( orig, cedge, edge );

//copy back the processed frame to image
memcpy( image.data, cedge->imageData, image.xsize*image.ysize*4 );

```

**pdp\_opencv\_bgsubtract & pdp\_opencv\_bgsubtract**

**pdp\_opencv\_distrans & pdp\_opencv\_distrans**

**pdp\_opencv\_laplace & pdp\_opencv\_laplace**

**pdp\_opencv\_motempl & pdp\_opencv\_motempl**

**pdp\_opencv\_morphology & pdp\_opencv\_morphology**

**pdp\_opencv\_floodfill & pdp\_opencv\_floodfill**

**pdp\_opencv\_harcascade & pix\_opencv\_harcascade**

**pdp\_opencv\_motempl & pix\_opencv\_motempl**

**pdp\_opencv\_contours\_boundingrect & pix\_opencv\_contours\_boundingrect**

**pdp\_opencv\_lk & pdp\_opencv\_lk**

## TRACKS TO GO FURTHER

---

body component extraction : <http://wsnl.stanford.edu/gesture.html> [<http://wsnl.stanford.edu/gesture.html>]

start/puredata\_opencv.txt · Last modified: 2011/10/17 18:14 by sevy

Except where otherwise noted, content on this wiki is licensed under the following license:CC Attribution-Noncommercial-Share Alike 3.0 Unported [<http://creativecommons.org/licenses/by-nc-sa/3.0/>]