OpenCV Tutorial II: Video Processing

iPAL @ PENNSTATE
Information Processing and Algorithms Laboratory

Xuan Mo

iPAL Group Meeting

February 11, 2011

Outline

- Reading video
- Writing video
- Edge Detection
- Demo: Laplacian edge detection

Capturing a frame from a video sequence

- Initializing capture from a camera:
  $CvCapture * capture = cvCaptureFromCAM(0);$
  capture from video device 0
- Initializing capture from a file:
  $CvCapture * capture = cvCaptureFromAVI("infile.avi");$
- Capturing a frame:
  $cvGrabFrame(capture);$
- retrieve the captured frame:
  $img = cvRetrieveFrame(capture);$
- Releasing the capture source:
  $cvReleaseCapture(\&capture);$
  Don't forget to release!

Get capture device properties

- Get capture device properties:
  $cvQueryFrame(capture);$
  this call is necessary to get correct capture properties
  $cvGetCaptureProperty(capture, property\_id, value);$
- property_id:
  $CV\_CAP\_PROP\_FRAME\_HEIGHT$
  $CV\_CAP\_PROP\_FRAME\_WIDTH$
  $CV\_CAP\_PROP\_FPS$
  $CV\_CAP\_PROP\_FRAME\_COUNT$
  Frame count does not seem to be working properly.

Get frame information

- Function is the same:
  $cvGetCaptureProperty(capture, property\_id, value);$
- property_id:
  $CV\_CAP\_PROP\_POS\_MSEC$
  $CV\_CAP\_PROP\_POS\_FRAMES$
  $CV\_CAP\_PROP\_POS\_AVI\_RATIO(0-1)$
  and so on...
- Example:
  $float \quad posRatio =$
  $cvGetCaptureProperty(capture, CV\_CAP\_PROP\_POS\_AVI\_RATIO)$

Set frame information

- Set capture device properties:
  $cvSetCaptureProperty(capture, property\_id, value);$ sets the specified property of camera or AVI.
- property_id:
  The same as getting frame information
- Example:
  $cvSetCaptureProperty$
  $(capture, CV\_CAP\_PROP\_POS\_AVI\_RATIO, 0.9);$
  start capturing from a relative position of 0.9 of a video file

Write/save video

- First initializing a video writer:
  $CvVideoWritercvCreateVideoWriter$
  $(filename, fourcc, fps, frame\_size, is\_color = 1)$
- fourcc: four-Character Codes
  $CV\_FOURCC('P','I','M','1') : MPEG-1$
  $CV\_FOURCC('M','J','P','G') : motion-jpeg$
  $CV\_FOURCC('M','P','4','2') : MPEG-4.2$
  $CV\_FOURCC('D','I','V','3') : MPEG-4.3$
  $CV\_FOURCC('D','I','V','X') : MPEG-4 = MPEG-1$
  $CV\_FOURCC('U','2','6','3') : H263$
  $CV\_FOURCC('I','2','6','3') : H263I$
  $CV\_FOURCC('F','L','V','1') : FLV1$
- Example:
  $CvVideoWriterwriter = 0;$
  $writer = cvCreateVideoWriter(\dots)$

Write/save video

- Then writing the video file: add the frame to the file
  $cvWriteFrame(writer, img);$
- Example:
  $IplImage * img = 0;$
  $intnFrames = 50;$
  $for(i = 0; i < nFrames; i++)\{$
  $cvGrabFrame(capture);$
  $img = cvRetrieveFrame(capture);$
  $cvWriteFrame(writer, img);\}$
- view the captured frames during capture
  $cvShowImage("mainWin", img);$
  $key = cvWaitKey(20);$ wait 20 ms
  wait 20 ms in order to display properly

Edge detection

- search-based
  1. Computing a measure of edge strength, usually a first-order derivative expression such as the gradient magnitude.
  2. Searching for local directional maxima of the gradient magnitude using a computed estimate of the local orientation of the edge, usually the gradient direction.
- zero-crossing based
  search for zero crossings in a second-order derivative expression computed from the image in order to find edges, usually the zero-crossings of the Laplacian or the zero-crossings of a non-linear differential expression.

Laplacian edge detection

Approximation of second derivative (horizontal):

$$\frac{\partial f^2(x,y)}{\partial^2 x} = f''(x,y) = f'(x+1,y) - f'(x,y) =$$
$$= [f(x+1,y) - f(x,y)] - [f(x,y) - f(x-1,y)]$$
$$= f(x+1,y) - 2\,f(x,y) + f(x-1,y)$$

convolution with: $[\,1\;\; -2\;\; 1\,]$

Approximation of second derivative (vertical):

convolution with: $\begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$

Laplacian Operator $\boxed{\nabla^2 = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)}$

convolution with: $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

Demo: Laplacian edge detection

1. Capture the video from WebCam
   - Function: $cvCaptureFromCAM$
2. Smoothing the frames
   - Function: $cvSmooth$
3. Split into different color spaces
   - Function: $cvSplit$
4. Add Laplacian filter in each color space
   - Function: $cvLaplace$
5. Merge the 3 color spaces
   - Function: $cvMerge$
6. Show the frame
   - Function: $cvShowImage$