Working with OpenCV and Intel Image Processing Libraries. Processing image Data Tools.

Faraón Llorens, Francisco José Mora , Mar Pujol, Ramón Rizo and Carlos Villagrá.

Grupo Vgia: Visión, Gráficos e Inteligencia Artificial
Departamento de Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante
E-03690, San Vicente, Spain
Phone +34 96 590 39 00, Fax +34 96590 39 02
e-mail: mora@dccia.ua.es

We will provide an overview of Intel OpenCV and Image Processing Libraries. We present an application of real-time gesture recognition using the libraries (segmenting a foreground object, creating Motion History Image (MHI), updating the intensity gradients, and recovering directional motion information).

Some times most companies spent a lot of time and money implementing those well-known techniques. OpenCV and IPL implement a huge amount of standard and advanced image processing techniques. With OpenCV we now have access to a well implemented version that is fast and reliable.

1. INTRODUCTION

OpenCV is computer-vision library for extracting and processing meaningful data from images. This library needs other Library, the Intel Image Processing Library. **IPL** provides a set of **low-level image manipulation** and **OpenCV** implements **techniques** and functions in the areas of **object/human/face segmentation, detection, recognition,** and **tracking,** as well as **camera calibration, stereovision,** and **2D/3D shape reconstruction**.

OpenCV is an **open-source** release that includes C source code for all of the library's functionality, it gives us confidence in the code and we have know **we can make local modifications** if necessary.

Developed by the Intel research group, OpenCV is **freely** available at www.intel.com/research/mrl/research/opencv/ and has **royalty free redistribution** license. Anyone interested in joining the user group needs to register with Yahoo groups at http://groups.yahoo.com/ and then can subscribe by sending email to OpenCV-subscribe@yahoogroups.com.

Intel has developed a uniquely decentralized research model with over 80 labs situated around the world. The bulk of the OpenCV software team resides Intel's Software Development Center in Nihzniy Novgorod, Russia. Established in 1999, the center currently employees over 100 computer research scientists and engineers working in areas such as computer graphics, vision, media, Bayesian networks, compilers and tools research. The **support** and **new versions** of the library are guaranteed.

OpenCV currently is supported to run on Windows and Linux operating systems. We can use the library with programming languages, for example Visual Basic or Java, and applications like MathLab.

There are other libraries developed for the Intel group. For example the Signal Processing Library is a library for work with audio signal, and Intel Vtune Performance analycer to calculate the time-codes.

IPL and OpenCV reference manuals detailed describe structure, operation ands functions of the libraries.

2. INTEL IMAGE PROCESSING LIBRARY V2.5

The Intel Image Processing Library provides a set of low-level image manipulation functions in standard DLLs and static libraries form. The functions are optimized for Intel Architecture processors, and are particularly effective at taking advantage of MMX™ technology, the Streaming SIMD Extensions (SSE) and SSE-2.. Currently, versions have been developed for the Intel486™ and compatible processors, the Pentium® processor, the Pentium Pro processor, the Pentium processor with MMX technology, the Pentium processor, the

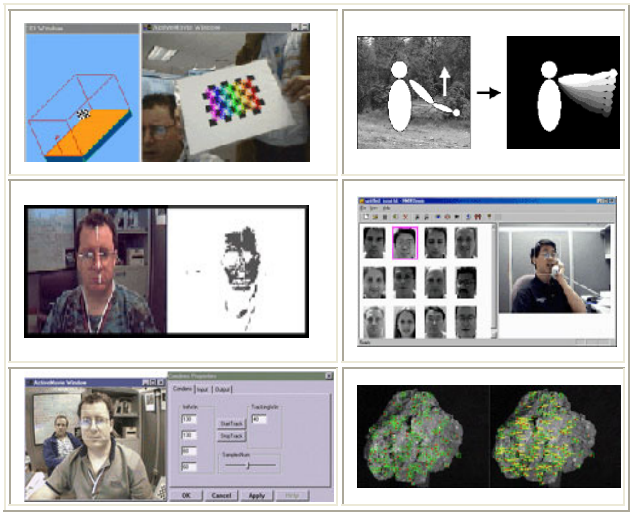

Figure 1. Some example areas would be Human-Computer Interaction (HCI), Motion Tracking , Face Recognition, Gesture Recognition and Segmentation and Recognition.

Pentium III processor, and, most recently, the Pentium 4 processor. A separate DLL is available for each processor.

The library contains functions that perform filtering, thresholding, and transforms (FFT, DCT, geometric), as well as arithmetic and morphological operations. The library uses a flexible image format, supporting channels of 1-, 8-, 16-, and 32-bit integer pixels, as well as 32-bit floating point pixels, with an arbitrary number of channels per image. Conversion to and from the Windows* DIB (device independent bitmap) image format is supported, as are conversions between color and gray-scale.

Major changes from the Image Processing Library 1.0 include many more geometric functions, mask region of interest, floating point support, example code, and color space conversions.

Changes since version 2.2 include further optimization for Pentium III processors and support for the Pentium 4 processor.

Additional information on this software as well as other Intel Performance Libraries is available at http://developer.intel.com/software/products/perflib/.

Intel® Image Processing Library (included in OpenCV WinOS download):

- Image creation and access (same image header used for both libraries).
- Image arithmetic and logic operations.
- Image filtering.
- Linear image transformation.
- Image morphology.
- Color space conversion.
- Image histogram and thresholding.
- Geometric transformation (zoom-decimate, rotate, mirror, shear, warp, perspective transform, affine transform).
- Image moments.

3. THE OPENCV LIBRARY

OpenCv contains more than 500 funtions. While the API of the library is C/C+, the bulk of the libray is made up of C functions and the objects themselves are mostly self contained.

OpenCV is aimed at making computer vision accessible to programmers and users in the area of real-time human-computer interaction and mobile robotics. Thus, the library comes with source code and hand-tuned assembly language binaries optimized fot Intel processors, so that users can both leran from the library and make use of its performance.

When you build and run an application using OpenCV, a built-in DLL switcher is called at run time to automatically detect the processor type and load the appropriate optimized DLL for that processor. If the processor type cannot be determined or if the appropriated DLL is not available, an optimized C code DLL is used. Included in the OpenCv download is the optimized Intel Image Processing Library (IPL) on wich OpenCv partially depends. Although it is included with OpenCV, you can also get IPL and other libraries for signal processing, matrix math, JPEG and pattern recognition at http://developer.intel.com/vtune/perflibst/.

OpenCv ships with HTML overview and a detailed manual in Pdf format. Added to this are many source-code examples, detailed papers, and tutorials on differents topics. The source-code examples currently include camera calibration, face traking, kalman filter, condensation filter, face recognition, optical flow, and morphing and image to produce intermediate views. An interpretative C prototyping environment for OpenCv and IPL is also available for download.

OpenCV addresses the areas of object/human/face segmentation, detection, recognition, and tracking, as well as camera calibration, stereovision, and 2D/3D shape reconstruction. A full matrix algebra package is also included in the library to support algorithms in these areas.

Also included in the library are routines in:

- Image functions: Creation, allocation, destruction of images. Fast pixel access macros.
- Data Structures: Static types and dynamic storage.
- Contour Processing: Finding, displaying, manipulation, and simplification of image contours.
- Geometry: Line and ellipse fitting. Convex hull. Contour analysis.
- Features: 1st & 2nd Image Derivatives. Lines: Canny, Hough. Corners: Finding, tracking.
- Image Statistics: In region of interest: Count, Mean, STD, Min, Max, Norm, Moments, Hu Moments.
- Image Pyramids: Power of 2. Color/texture segmentation.

- Morphology: Erode, dilate, open, close. Gradient, top-hat, black-hat.
- Background Differencing: Accumulate images and squared images. Running averages.
- Distance Transform: Distance Transform
- Thresholding: Binary, inverse binary, truncated, to zero, to zero inverse.
- Flood Fill: 4 and 8 connected
- Camera Calibration: Intrinsic and extrinsic, Rodrigues, un-distortion, Finding checkerboard calibration pattern
- View Morphing: 8 point algorithm, Epipolar alignment of images
- Motion Templates: Overlaying silhouettes: motion history image, gradient and weighted global motion.
- CAMSHIFT: Mean shift algorithm and variant
- Active Contours: Snakes
- Optical Flow: HS, L-K, BM and L-K in pyramid.
- Estimators: Kalman and Condensation.
- POSIT: 6DOF model based estimate from 1 2D view.
- Histogram (recognition) Manipulation, comparison, backprojection. Earth Mover's Distance (EMD).
- Gesture Recognition: Stereo based: Finding hand, hand mask. Image homography, bounding box.
- Matrix: Matrix Math: SVD, inverse, cross-product, Mahalanobis, eigen values and vectors. Perspective projection.
- Eigen Objects: Calc Cov Matrix, Calc Eigen objects, decomp. coeffs. Decomposition and projection.
- embedded HMMs: Create, destroy, observation vectors, DCT, Viterbi Segmentation, training and test.
- Drawing Primatives: Line, rectangle, circle, ellipse, polygon. Text on images.
- System Functions: Load optimized code. Get processor info.
- Utility: Abs difference. Template matching. Pixel order<->Plane order. Convert Scale. Sampling lines. Bi-linear interpolation. ArcTan, sqrt, inv-sqrt, reciprocal. CartToPolar, Exp, Log. Random numbs. Set image. K-Means.

4. WORKING WITH OPENCV AND IMAGE PROCESSING LIBRARIES.

We will present a real-time application that detects motions. This application has built on a standard Pc platform employing optimized OpenCv and IPL routines.

For more details of this algorithm and code just described, go to "http://www.cse.lehigh.edu/FRAME/ Davis/DavisBradski.htm" In listings you can se that all functions and types of the OpenCV start with "cv" and Image Processing Library with "ipl". We construct a more localized motion characterization for the Motion History Image that extract motion orientations in real-time.

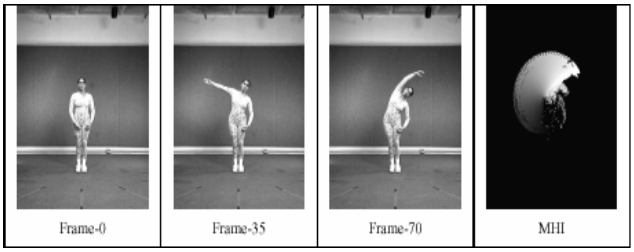

Figure 2. Motion History Image for arm-streaching mvement generated from image differences.

Basically, directional motion information can be recovered directly from the intensity gradients with the MHI. In addition, we provide a few simple motion features using these orientations. Figure 2 is a flowchart of this algorithm.

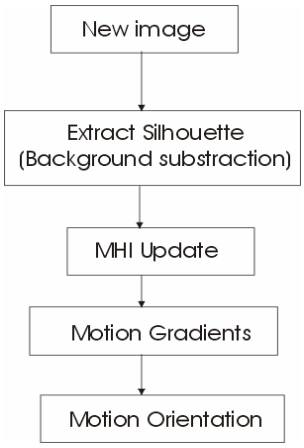

Figure 2: FlowChart algorithm.

For this discussion, assume the camera is stationary. Althought there are many more sophisticated techniques of segmenting a foregroung object from the learned background, in this example we label as potention foreground those pixel that are a set number standard deviations from the mean color background. When a new video image comes in, the foreground is separated from the background using background substraction. In order to do this, we need to first learn the background model. The learned background model will consist of the means and standard deviations of each pixel over many frames when no foreground object is present. Listing 1 shows the routine that does this. The pixel values and squared values of pixels are assumed into floating-point images and these images are divided by the number of video frames (here 45 frames or 1.5 seconds at 30 frames/sec) after collection is finished. This yields the mean and standard deviation of each pixel. Since IstdFP will be used as the threshold difference from Imean[x,y] at wich a future pixel at x,y will be declared to be a foreground pixel.

```
void captureBackground(IplImage
*IvidIm,
         IplImage *IstdFP, IplImage
*Iu,
         IplImage *Istd)
{
int height = IvidIm->height;
int width = IvidIm->width;
int I;

//Create background mask (find mean and
variance of color background):
IplImage *Imean =
iplCreateImageHeader(3. 0,
IPL_DEPTH_32F, "RGB", BGR",
IPL_DATA_ORDER_PIXEL, IPL ORIGIN_TL,
IPL_ALIGN_QWORD,width, height, NULL,
NULL, NULL, NULL);
iplAllocateImageFP(Imean, 1,0.0);
int len = width*height*3;

//take statistics over 45 frames (~1.5
sees)
for(i = 0: i<45: i++)
   {
   grabIm(IvidIm);//Get an image into
IvidIm
   cvAcc(IvidIm, Imean);//Accumulate it
into
                        // Imean
   cvSquareAcc(IvidIm,
IstdFP);//Accumulate
           // squared image into IstdFP
   }
//find mean and vars
//meanI
iplMultiplySFP(Imean, Imean,
(float)1.0/i);

//meanI^2
iplMultiplySFP(IstdFP, IstdFP.
(float)1.0/i); IplImage* ImeanSqr =
iplCloneImage(Imean);
iplSquare(ImeanSqr, ImeanSqr);

//Ivar = meanIA2 - (meanI)^2
iplSubtract(IstdFP, LneanSqr, IstdEP);
iplDeallocate(ImeanSqr, IPL_IMAGE_ALL);

//IstdFP = sqrt(Ivar)
cvbSqrt((const
float*)IstdFP->imageData,
       (float*)IstdFP->imageData, len);
   //since we use Istd as a threshold,
   enforce that no threshold is too small
   float *pIstdFP = (float
   *)IstdFP->imageData;
   for(i=0: i<len: i++)
      {
      if (*pIstdFP < 0.3) *pIstdFP = 0.3;
      pIstdFP++;
      }

   //meanI^2
   iplMultiplySFP(IstdFP,
   IstdFP.backThresh);

//convert to 8u images
convert32Fto8U (Imean, Iu);
convert32Fto8U(IstdFP,Istd);
iplDeallocate(Imean, IPL_IMAGE ALL);
```

Listing 1.
Collect background mean and std of each pixel for background differencing code.

After the mean and the standard deviation of the color background are calculated, the code then creates a binary mask of all possible regions are those pixels that deviate more than a given number of standard deviations from the mean values. Image dilation is used to help cose holes in the foreground regions caused by image noise as in Listing 2.

```
//Ii Video input image BGR
//Im Mean image of background BGR
//Is Standard deviation of background from
mean BGR
//Io Output image -- Grayscale
//Iot Temporary output image -- Grayscale
//Itl,It2 Temporary images of same size,
           depth and number of channels
   BGR
//numIterations Number of dialations to
preform on foreground
void backsubCVL(IplImage *Ii, IplImage *Im.
IplImage *Is, IplImage *Io,
IplImage *Iot, IplImage *Itl, IplImage
*It2, int numIterations)
   {
   //Get 'Ii-Im
   iplSubtract(Ii,Im,Itl);
   iplSubtract(Im,Ii,It2);
   iplAdd(Itl,It2,Itl);

   //Get Raw foreground = ~Ii-Im, > Is ?
   255: 0;
   iplSubtract(Itl,Is,Itl);
   iplThreshold(Itl, Itl, 1);
   iplColorToGray(Itl, Io);
   iplThreshold ( Io , Iot , 1 );

   //Fill up holes in the foreground
   iplDilate(Iot,Io, numIterations);
   }
```

Listing 2.
Extract the foreground and fill in holes code.

The foreground object is layered onto the "time Motion History Image" (tMHI). Image gradients of the tMHI are calculated, the gradients directions indicate movement patters of the foreground object. In the OpenCv library, updatinf tMHI image is wrapeped into one function call, cvUpdate MHIByTime(…). The image gradients are calculated from the MHI via the call cvCalMotionGradient(…) this yields directions of motion encoded implicity by the tMHI. Finally the code computes the global direction of movement via the call cvCalcglobalorientation(…). These calls are show in Listing 3. Figure 4 depicts the process of going from the MHI, to a gradient representation to the global motion in that region.

```
   //UPDATE THE MOTION HISTORY IMAGE
   cvUpdateMHIByTime(IsilIm, IrhiIm,
   timestamp, MHI DURATION):

   //CREATE MOTION GRADIENT ORIENATIONS
   FROM THE tMHI
   cvCalcMotionGradient(IrhiIm, Imask,
   Iorient, 3, MAX_TIME_DELTA,MIN
   TIME_DELTA )

   //CALCULATE THE MOTION ORIENTATION
   globalDir = cvCalcGlobalOrientation (
   Iorient, Imask, ImhiIm,
   timestamp,MHI_DURATION):

   globalDir = 360.0 - globalDir:
   //Circularly rotate angle since TL
   origin
   rather //than BL default
```

Listing 3.
Update tMHi, Calculate the Motion Gradient and global motion orientation code.

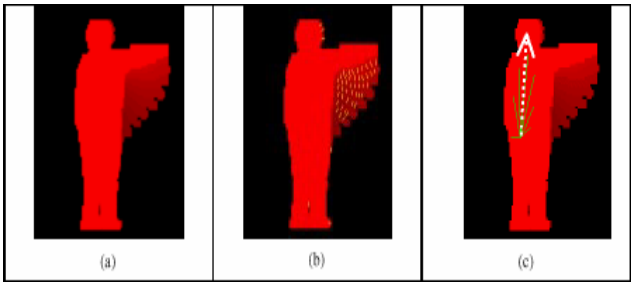

Figure 4. From tMHI to a gradient representation.
(a) tMHI (b) Gradients (c) Global Motion

Table 1 list code timing for some of the IPL-OpenCv functions invoved in this process. The timings are in form of clock cycles per pixel, and speed-up factor is going from optimized C code to optimized assembly is show in this column. Thus, We work with 160x120 video images at 30Hz on 500MHz Pentium III. Intel´s Vtune Performance analycer code was used to record the performance of the functions using both the optimised C (and assembly versions of the libraries). Vtune rapidly samples the application and produces a report aas to the precent of time spent in each module.

Function	Optimized C	Optimized MMX	Speed-up
cvAcc	7.2	2.4	3.02
CvSquareAcc	9.9	2.6	3.77
CvSqrt	24.4	8.6	2.84
CvUpdateMHIByTime	13.0	8.5	1.63
CvCalcMotionGradient	193.2	82.9	2.33
CvCalcglobalOrientation	23.7	21.3	1.11

Table 1. Code Timing for some of the OpenCv functions

5. USING OPENCV AND IPL WITH DIFERENT OPERATING SYSTEMS AND PROGRAMMING LANGUAGES

IPL and OpenCV is supported to run on Windows 98/Millenium/NT/2000/XP. On Linux or other operating systems the source code should build. Actually, There isn't Official Linux Support.

Objects and functions are Externed as C to avoid name mangling, allowing OpenCV to be used with Visual Basic and Java. Object-oriented design is mostly intented to take place at COM or CORBa level for use in applications.

It's easy and quickly work with IPL and OpenCV libraries in MsWindows with Microsoft Visual C++, Borland C++ and other C++ compilers. We can work with other programming languages but it's more difficult, mainly in OPenCV. IPL can be easily used with Borland Delphi to.

In addition, the new 2.1 release includes an optional interface so all OpenCV functions can be imported into Matlab, one of the most widely used software development tools for computer vision research.

Hawk is an application distributed with OpenCV that allows write and run short scripts in C and see graphical results. It is application that integrates C iterpreter (modified version of EiC. Original EiC site is located at http://www.kd-dev.com/~eic/), simple graphic library (highgui) and OpenCV+IPL (but, actually, any shared library can be attached to the application as plugin).

We can work jointly with other libraries, for example Microsoft Vision SDK or ImageMagick . Microsoft Vision SDK is a library for writing programs to perform image manipulation and analysis on computers running Microsoft Windows operating systems. If users need to read and write other file formats, it is possible to using a library named ImageMagick

6. CONCLUSIONS

OpenCV and IPL implement a huge amount of standard and advanced image processing techniques. It's easy and cheap to use these tools. IPL and OpenCV is supported to run on Microsoft Windows and Linux operating Systems. We can use these libraries with differents programming languages and tools such as Microsoft Visual C++, Borland C++, Java, Borland Delphi, Visual Basic, MathLab, etc… We quickly can use fast and reliable implemented functions with IPL and OpenCV.

7. ACKNOWLEDMENTS

This work has been supported by the Spanish Comisión Interministerial de Ciencia y Tecnología (CICYT), project number TIC2001-0245-C02-02

8. REFERENCES

[1] Intel Image Processinf Library Reference Manual. 2001

[2] Intel Open Computer Vision Library Reference Manual. 2001

[3] Bradski, G. and Davis, James. "Real-time Motion Template Gradientes using Intel CVLib". ICCV'99 http://www.cse.lehigh.edu/FRAME/Davis/DavisBradski.htm

[4] Bradski, G.,B-L Yeo, and M.Yeung. "Gesture for Video content navigation" SPIE '99, 356-24 s6, 1999.

[5] Bradski, G. "Comeputer Vision Face Trackingfor use in perceptual user interface". Intel Technology Journal. http://developer.intel.com/technology/itj/q21998/articles/art_2.htm. Q2. 1998

[6] P. Fieguth and D. Terzopoulos, "Color-based tracking of heads and other mobile objects at video frame rates," In Proc. Of IEEE CVPR, pp. 21-27, 1997.

[7] D. Comaniciu and P. Meer, "Robust Analysis of Feature Spaces: Color Image Segmentation," CVPR'97, pp. 750-755.

[8] M. Kass, A. Witkin D.Terzopoulos, "Snakes: Active contour Models," Int. J. o f Computer Vision (1) #4, pp. 321-331, 1988.