Histograms For Object Detection

Histograms is function of counting frequency of events, it can be used to represent
(1) gray scale or color distribution of an object,
(2) an edge gradient template of an object [Freeman95], and
(3) the distribution of probabilities representing current hypothesis about an object's location.

Example from "learning OpenCV" pp. 194, use of histograms for gesture recognition.
(1) Edge gradients were collected from "up", "right", "left ", "stop" and "OK" gestures.
(2) Color ROI were detected from video; then edge gradient directions were computed in the ROI, and
(3) These directions were collected into orientation bins within a histogram.
(4) The histograms were then matched against the gesture models to recognize the gesture.
(5) The vertical bars in Figure show the match levels. The gray horizontal line for the acceptance threshold.

Edge Gradient Definition

https://en.wikipedia.org/wiki/Image_gradient

Edge derivative is defined as follows,

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Which can be computed by using finite difference formula.
The edge gradient is defined as the angle:

$$\theta = \tan^{-1}\left[\frac{g_y}{g_x}\right],$$

The magnitude of the gradient is:

$$\sqrt{g_y^2 + g_x^2}$$

Example: based on edge detection technique, find the edge gradient map (angle), and gradient magnitude map.

Assume central difference kernel

x

H-plane

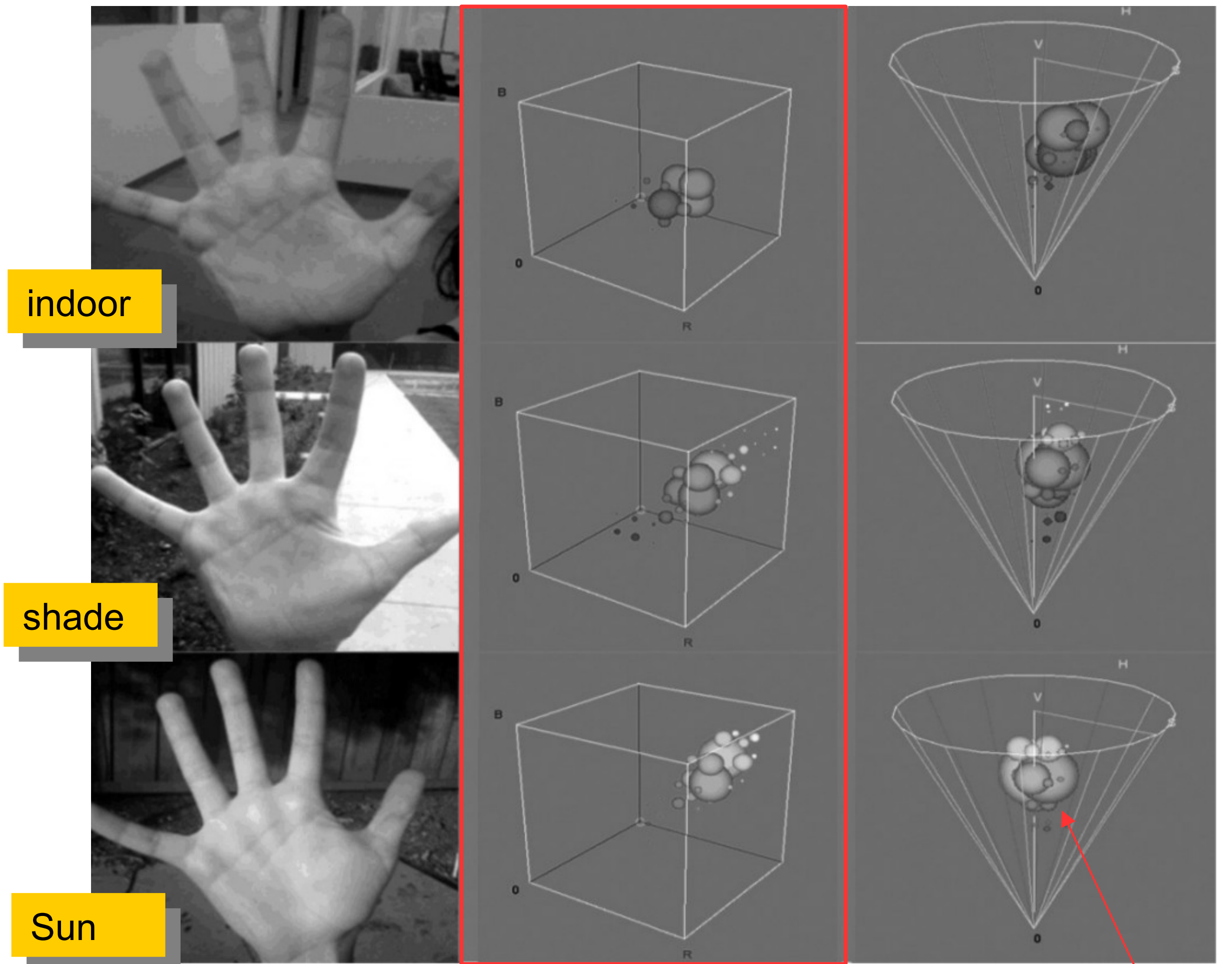
y

1	2	1	2	1
3	4	3	3	2
4	3	2	3	3
5	3	4	5	5
3	3	3	4	5

Homework: write openCV program based on 2D convolution to find edge derivatives, then find the gradient map (image), and display it with dynamic range defined in [0,255].

RGB And HSV Histograms

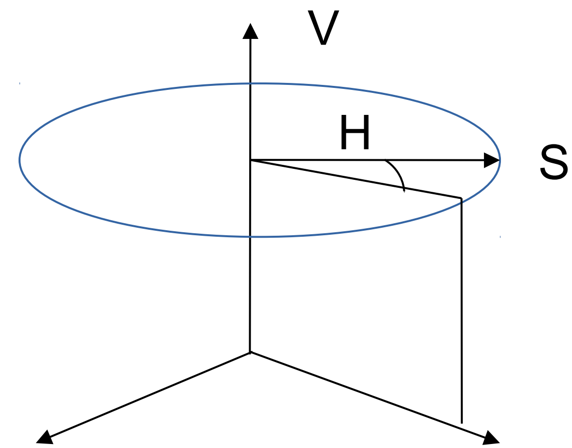
BGR and HSV histograms

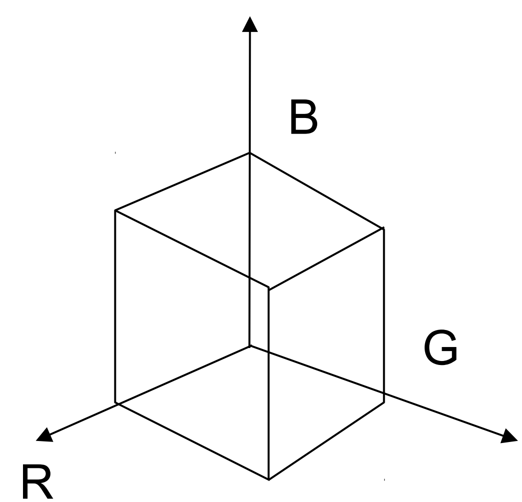


Reference: Learning OpenCV

Each cluster: for each region in the image

HSV histograms:
V (value) vertical axis,
S (saturation) radius, and
H (hue) the angle





Histogram Example

See reference: pp. 195/211

Example: Histogram on grids, from “learning openCV”.

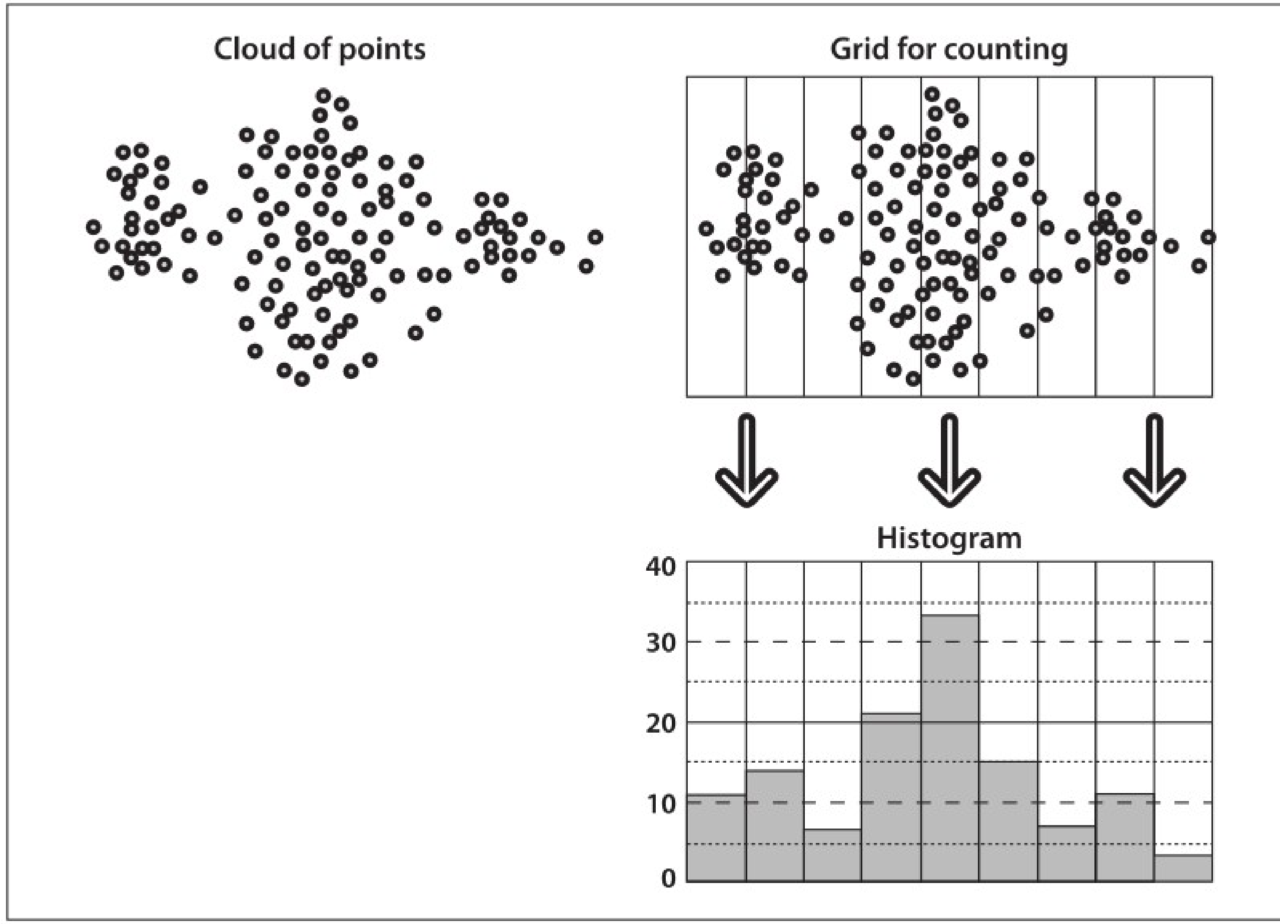
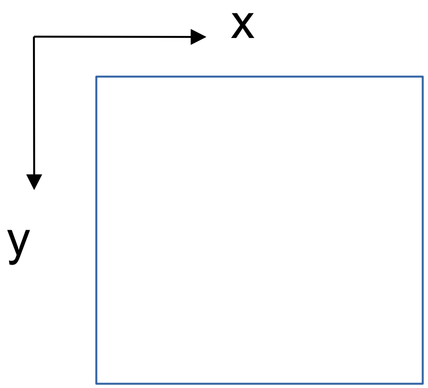


Figure 7-2. Typical histogram example: starting with a cloud of points (upper left), a counting grid is imposed (upper right) that yields a one-dimensional histogram of point counts (lower right)

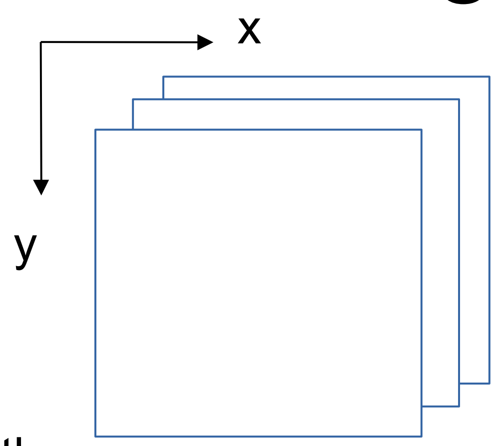
H-S Histograms Example



Given a color image I(x,y), in HSV space, the following example is for BGR planes

Example: writing pixel

```
>>> img[100,100] = [255,255,255]
>>> print img[100,100]
[255 255 255]
```



Split to hsv planes

```
>>> b,g,r = cv2.split(img)
>>> img = cv2.merge((b,g,r))
```

H-plane

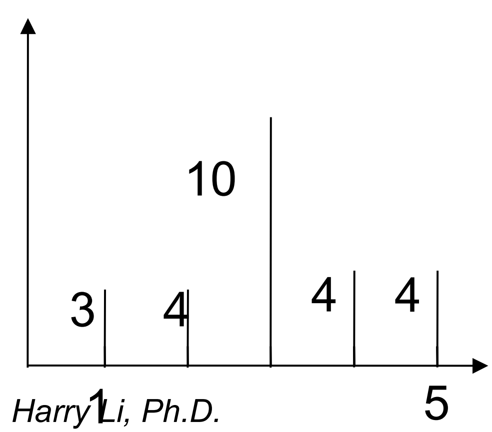
1	2	1	2	1
3	4	3	3	2
4	3	2	3	3
5	3	4	5	5
3	3	3	4	5

S-plane

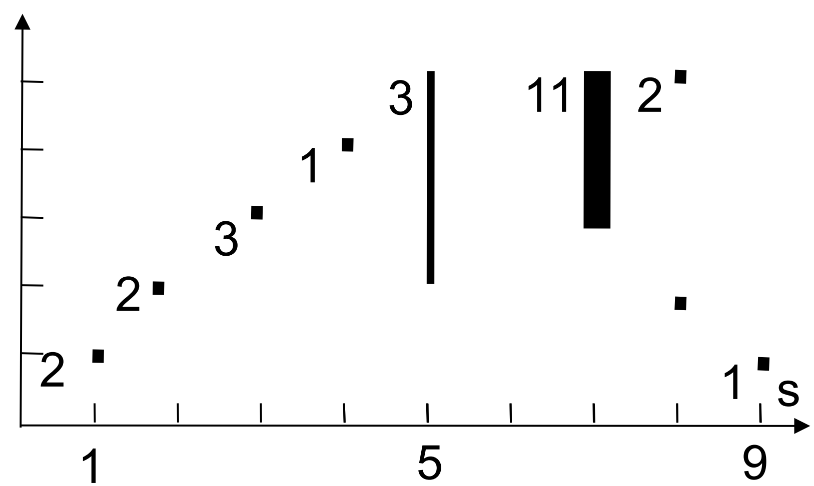
9	2	1	2	1
7	7	7	3	5
7	7	8	3	3
7	7	7	5	8
7	7	7	4	5

Example: (1) generate histogram for H and S; (2) generate histogram with S as independent variable and H as its function (see Learning OpenCV example for hand gesture recognition.

Hist for H



Hist for H



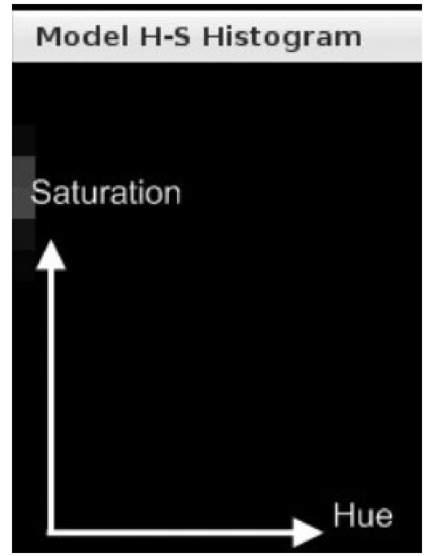
Homework: if H plane has grids as in red frames, then generate histogram.

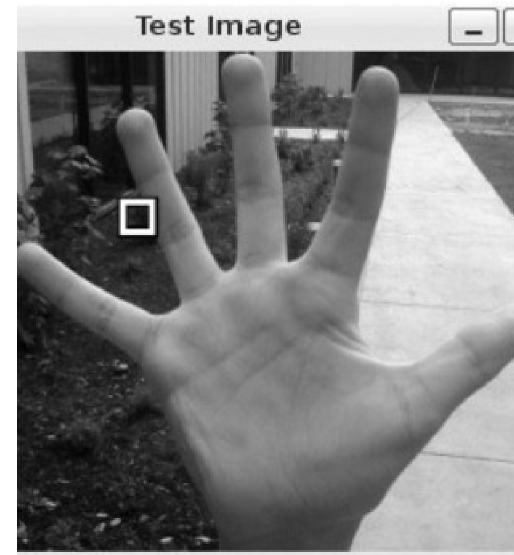
Back Projection Patch (BPP) Technique

Reference: Learning OpenCV

```
void cvCalcBackProjectPatch(
IplImage** images,
CvArr*      dst,
CvSize      patch_size,
CvHistogram* hist,
int method,
float  factor
);
```

For finding pdf distribution and detecting the object





Back projection patch with small white box much smaller than the object; here, the color histogram was of object-color distribution and the peak locations tend to be at the center of the object.

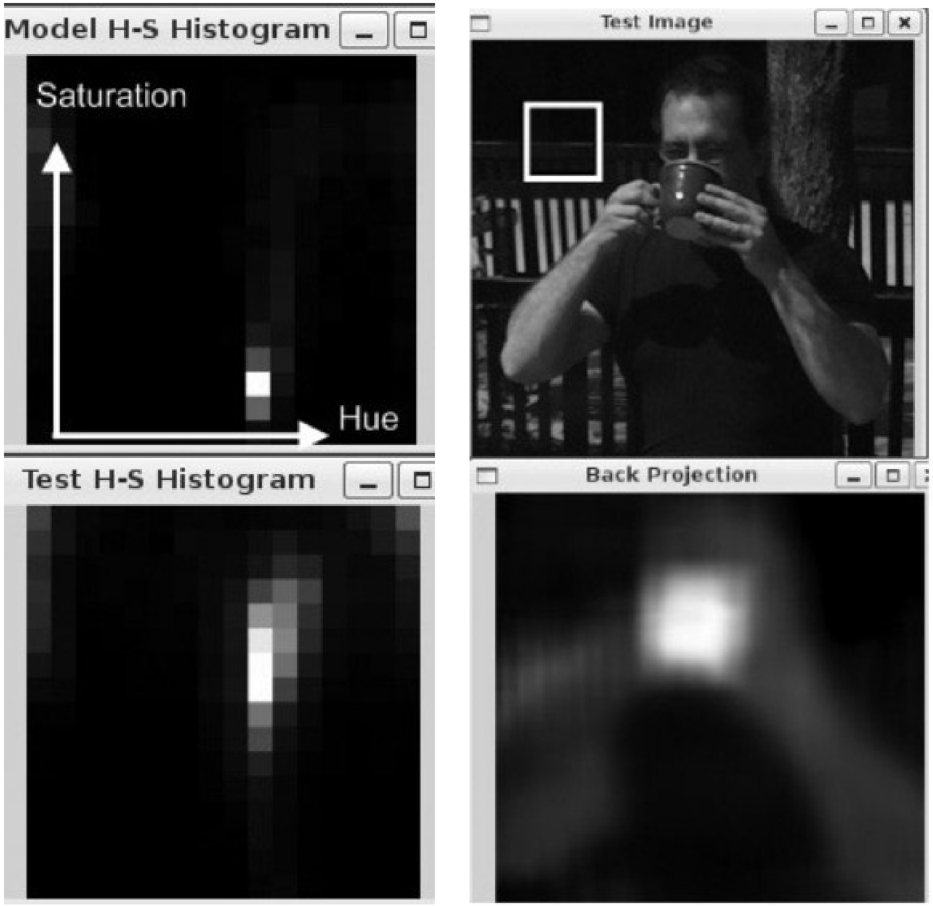
BPP To Find PDF Distribution Map

Reference: Learning OpenCV

Step 1. Experimentally define HS histogram, shown top right, hand-picked histogram (top right), Use a single-channel image to create the histogram using cvCalcHist(), use it to define hist of the patch

Step 2. Then use Back Projection Patch (BPP) to find pdf distribution on the ROI localized on the object;

```
void cvCalcBackProjectPatch(
IplImage** images,
CvArr*      dst,
CvSize      patch_size,
CvHistogram* hist,
int method,
float  factor
);
```

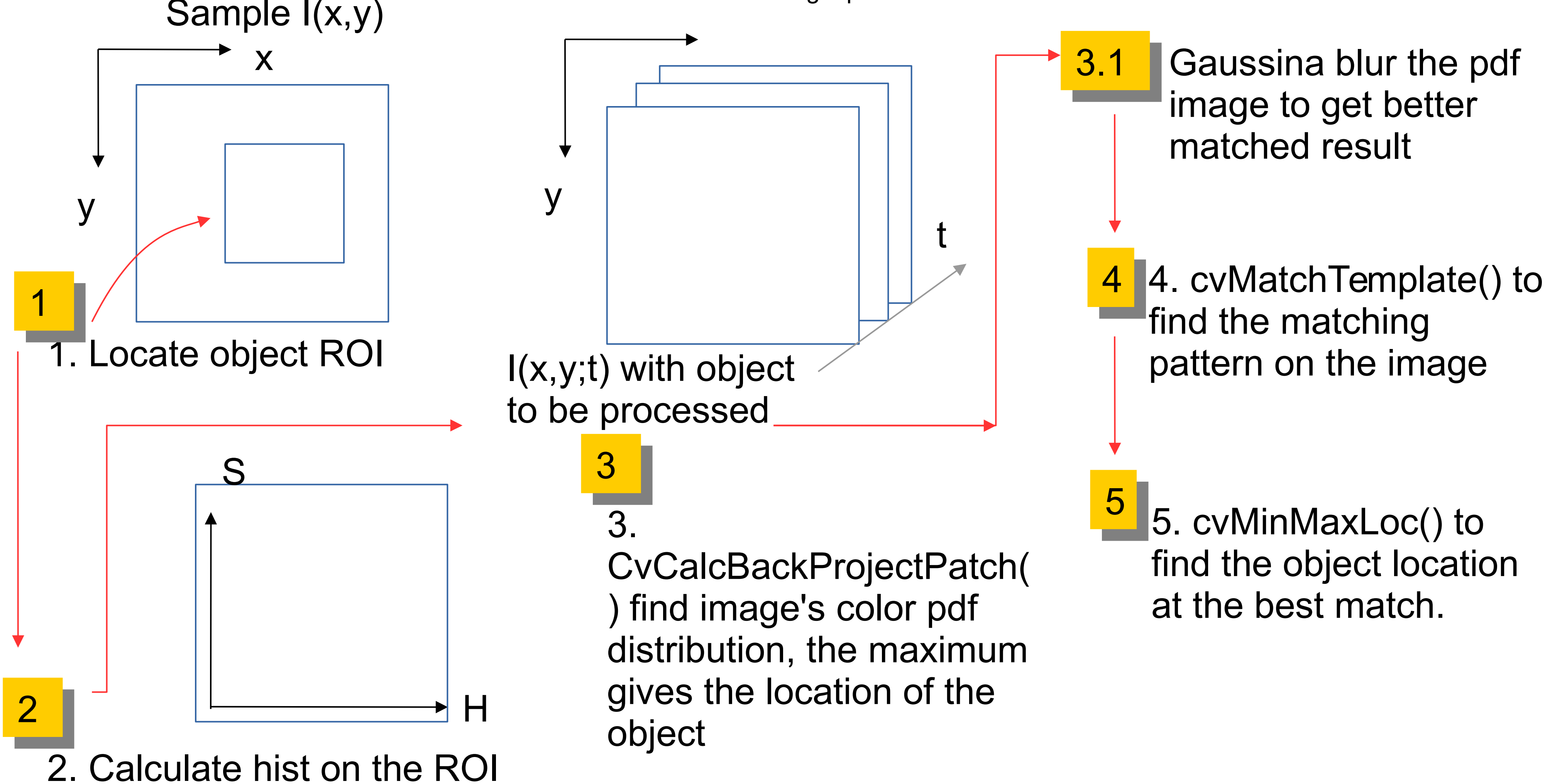


the destination image dst is different: it can only be a single-channel, floating-point image with size (images[0][0].width – patch_size.x + 1, images[0][0].height – patch_size.y + 1)

Harry Li, Ph.D.

BPP To Find Object

Reference: Learning OpenCV

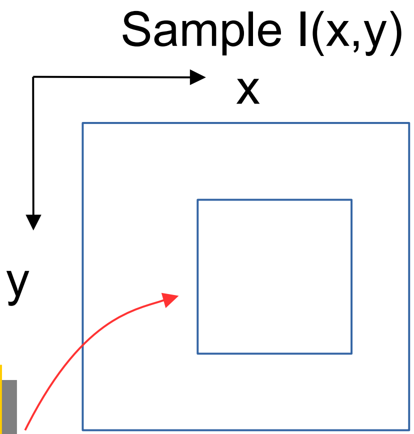


A good match should have good matches nearby, slight misalignments of the template shouldn't vary the results too much for real matches. Looking for the best matching "hill" can be done by slightly smoothing the result image before seeking the maximum. The morphological operators can also be helpful in this context.

Harry Li, Ph.D.

Locate Object ROI Calculate HS Hist

Reference: Learning OpenCV



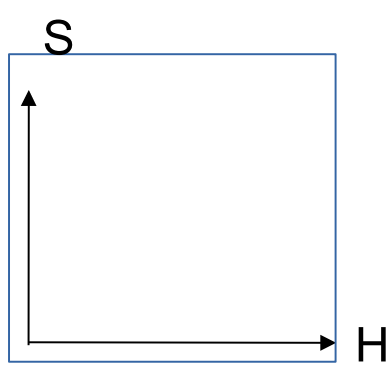
1

1. Locate object ROI

Write a program:
1. Move cursor on a displayed image, by clicking mouse to identify rectangle ROI with (x_min, y_min) and (x_max, y_max);

2.1 then read image ROI;

```
>>> ball = img[280:340, 330:390]
>>> img[273:333, 100:160] = ball
```



2

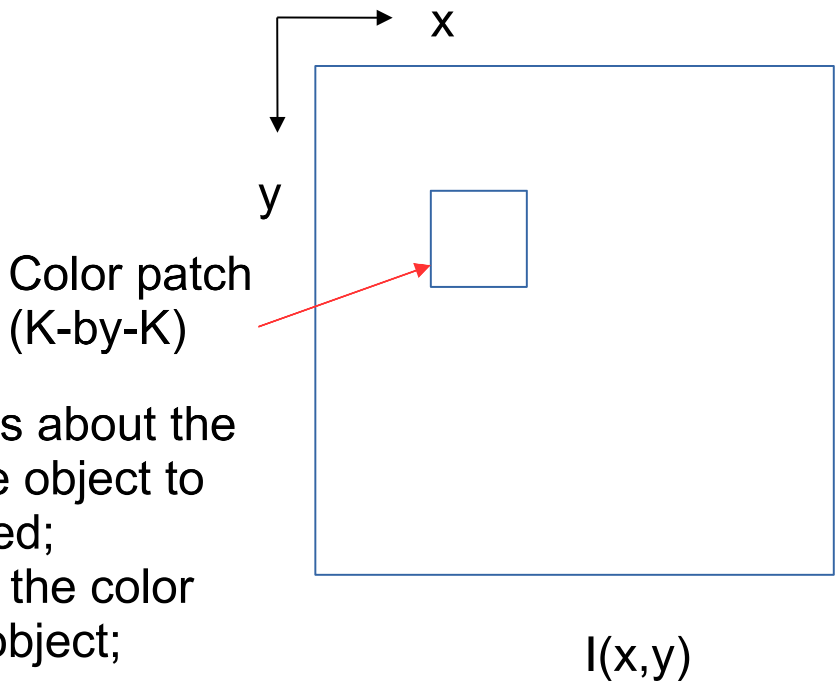
2. Calculate hist on the ROI

2.2 Calculate H-S Hist on the ROI, with H (hue) as independent variable and S (saturation) as a function;

2 Compute histogram

```
int h_bins = 30, s_bins = 32;  //grids
CvHistogram* hist;
{
int hist_size[ ] = { h_bins, s_bins };
float h_ranges[ ] = { 0, 180 }; //hue [0,180]
float s_ranges[ ] = { 0, 255 };
float* ranges[ ] = { h_ranges, s_ranges };
hist = cvCreateHist(
2,
hist_size,
CV_HIST_ARRAY,
ranges,
1
);
}
cvCalcHist( planes, hist, 0, 0 ); //histo
cvNormalizeHist( hist[i], 1.0 );
```

cvCalcBackProjectPatch() Finds Color ROI

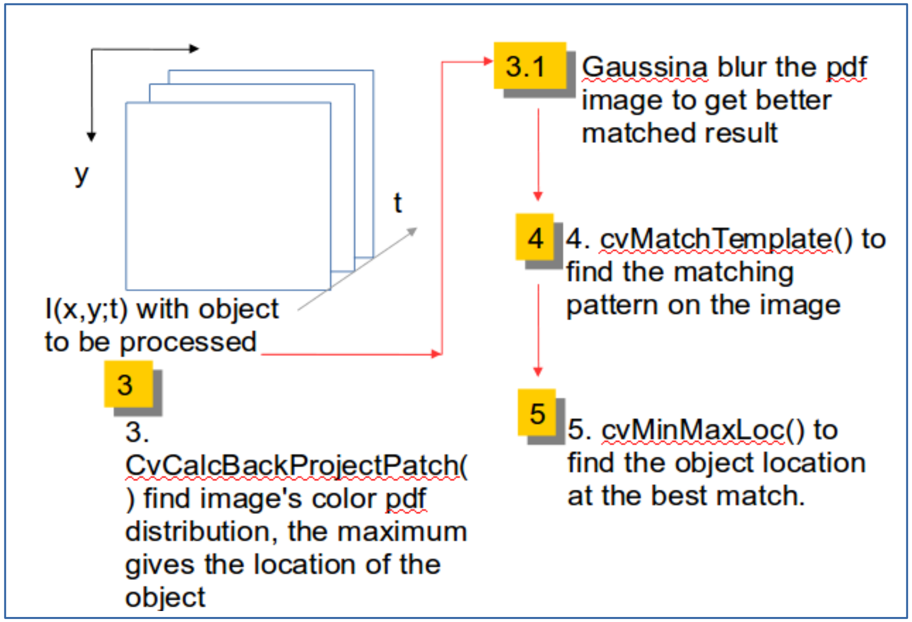


1. size K is about the size of the object to be detected;
2. color is the color from the object;

cvCalcBackProjectPatch() : as a region detector, the sampling window can be much smaller than or equal to the size of the object.

Color distribution, e.g., the probability of the object at that pixel given all the pixels in the surrounding window in the original image. When the window size is roughly the same size as the objects, the whole object “lights up” in the back projection. Finding peaks in the back projection image then corresponds to fi nding the location of objects

CvCalcBackProjectPatch



cvCalcHist() H-S Histogram

Reference: Learning OpenCV

Computes a hue-saturation histogram and draws it an illuminated grid.

Example: pp. 204

1 Decompose hsv image into separate planes

```
src=cvLoadImage(argv[1], 1));
IplImage* hsv = cvCreateImage( cvGetSize(src), 8, 3 );
cvCvtColor( src, hsv, CV_BGR2HSV );

IplImage* h_plane = cvCreateImage( cvGetSize(src), 8, 1 );
IplImage* s_plane = cvCreateImage( cvGetSize(src), 8, 1 );
IplImage* v_plane = cvCreateImage( cvGetSize(src), 8, 1 );
IplImage* planes[] = { h_plane, s_plane };
cvCvtPixToPlane( hsv, h_plane, s_plane, v_plane, 0 );
```

2 Compute histogram

```
int h_bins = 30, s_bins = 32;  //grids
CvHistogram* hist;
{
int hist_size[ ] = { h_bins, s_bins };
float h_ranges[ ] = { 0, 180 }; //hue [0,180]
float s_ranges[ ] = { 0, 255 };
float* ranges[ ] = { h_ranges, s_ranges };
hist = cvCreateHist(
2,
hist_size,
CV_HIST_ARRAY,
ranges,
1
);
}
cvCalcHist( planes, hist, 0, 0 ); //histo
cvNormalizeHist( hist[i], 1.0 );
```

3 display histogram

```
int scale = 10;
IplImage* hist_img = cvCreateImage(cvSize( h_bins * scale,
s_bins * scale ), 8, 3);
cvZero( hist_img );
```

4 Draw hist on image

```
cvGetMinMaxHistValue( hist, 0, &max_value, 0, 0 );
for( int h = 0; h < h_bins; h++ ) {   for( int s = 0; s < s_bins; s++ ) {
float bin_val = cvQueryHistValue_2D( hist, h, s );
int intensity = cvRound( bin_val * 255 / max_value );
cvRectangle(hist_img,cvPoint( h*scale, s*scale ),cvPoint( (h+1)*scale - 1, (s+1)*scale –
1),CV_RGB(intensity,intensity,intensity),CV_FILLED);  }}
```

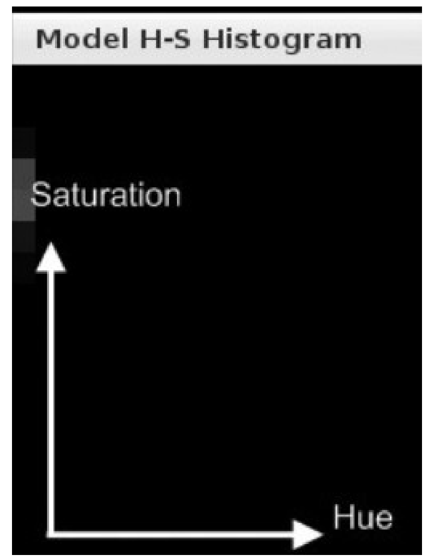
Back Projection Patch Analysis

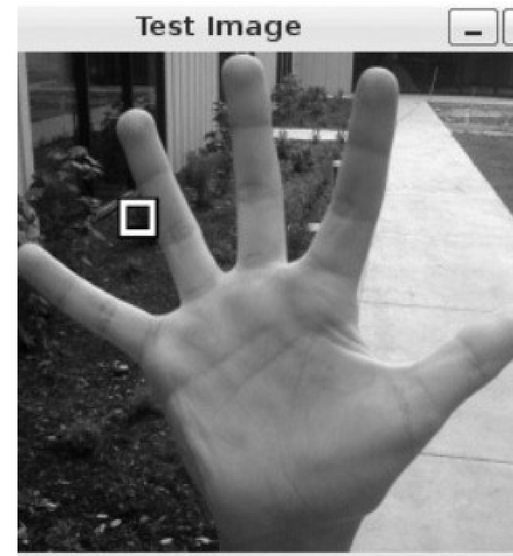
Reference: Learning OpenCV

Back projection is a way of recording how well the pixels (for cvCalcBackProject()) or patches of pixels (for cvCalcBackProjectPatch()) fit the distribution of pixels in a histogram model. If we have a histogram of flesh color then we can use back projection to find flesh color areas in an image.

```
void cvCalcBackProject(
IplImage**
image,
CvArr*
back_project,
const CvHistogram* hist
);
```

For testing histogram






```
void cvCalcBackProjectPatch(
IplImage** images,
CvArr*     dst,
CvSize     patch_size,
CvHistogram* hist,
int method,
float  factor
);
```

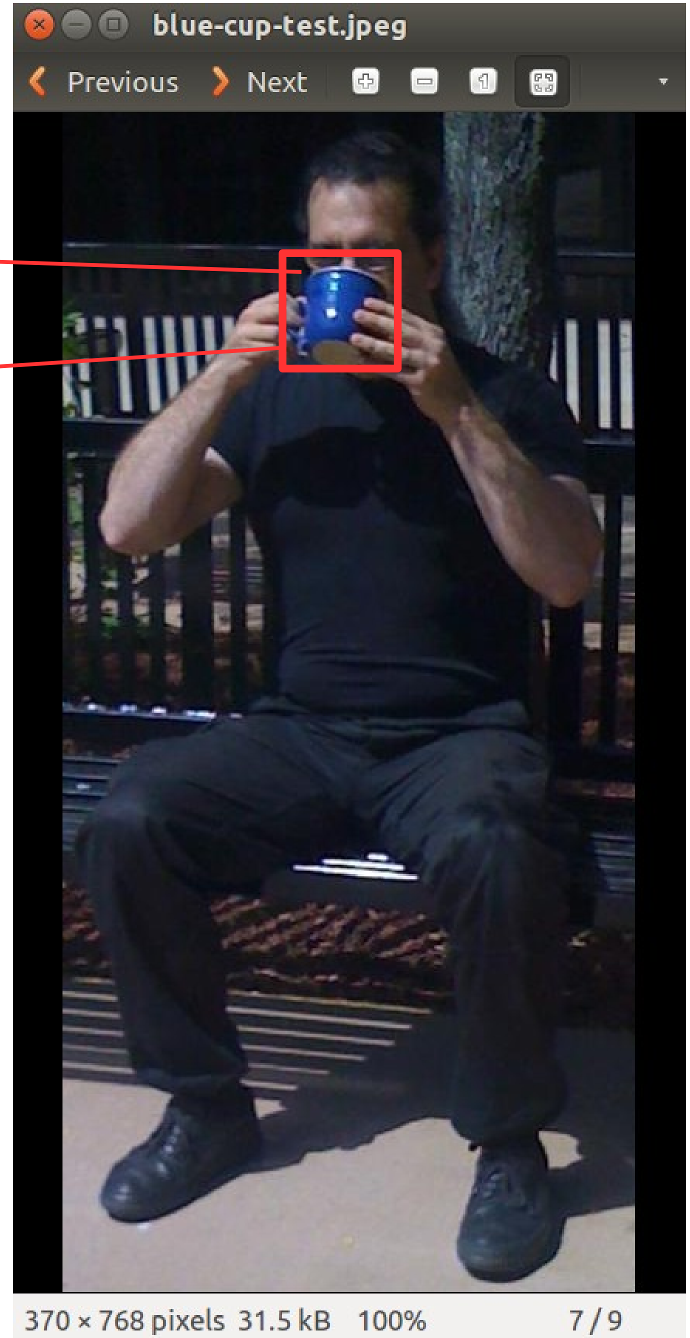
For finding pdf distribution and detecting the object

Back projection patch with small white box much smaller than the object; here, the color histogram was of object-color distribution and the peak locations tend to be at the center of the object.

April 23 Hist On ROI

The first step object image SRC just contains the object to search:

blue-cup-just.jpeg



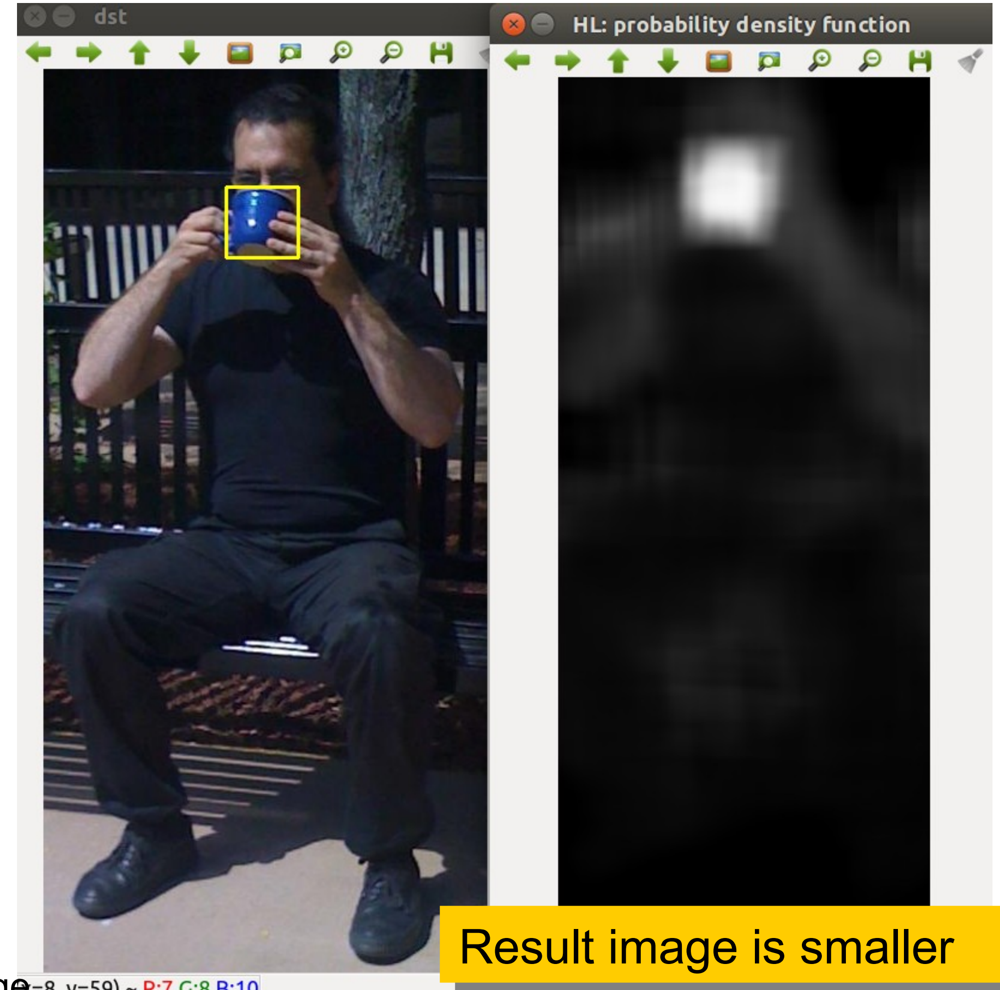
April 23 BPP Implementation

CmakeLists.txt

```
cmake_minimum_required(VERSION 2.8)
project( BackProjectPatch )
find_package( OpenCV REQUIRED )
include_directories( ${OpenCV_INCLUDE_DIRS} )
add_executable( BackProjectPatch calBackProjectPatch.cpp )
target_link_libraries( BackProjectPatch ${OpenCV_LIBS} )
```

```
/*---------------------------------------------------------------------------*
 * Program: calBackProjectPatch.cpp;  coded by:                              *
 * Modified by: HL, CTI Plus Corporation, copyrighted;                       *
 * Date:        April 23, 2018;  Version: 0x1.0;                             *
 * Status:      tested;                                                      *
 * Compile and build: CMakeLists.txt, cmake .  then make                     *
 *---------------------------------------------------------------------------*/
#include <opencv2/opencv.hpp>  //HL
#include <highgui.h>
void GetHSV (const IplImage* image, IplImage** h,
             IplImage** s, IplImage** v);

int main()
{
   IplImage* src = cvLoadImage ("bluecup.jpg", 1); //patch image
   IplImage* h_src = NULL;                  //define h plane
   IplImage* s_src = NULL;                  //define s plane
   GetHSV (src, &h_src, &s_src, NULL);    //just h-s planes
   IplImage *images[] = {h_src,s_src};
   CvHistogram* hist_src = NULL;
```



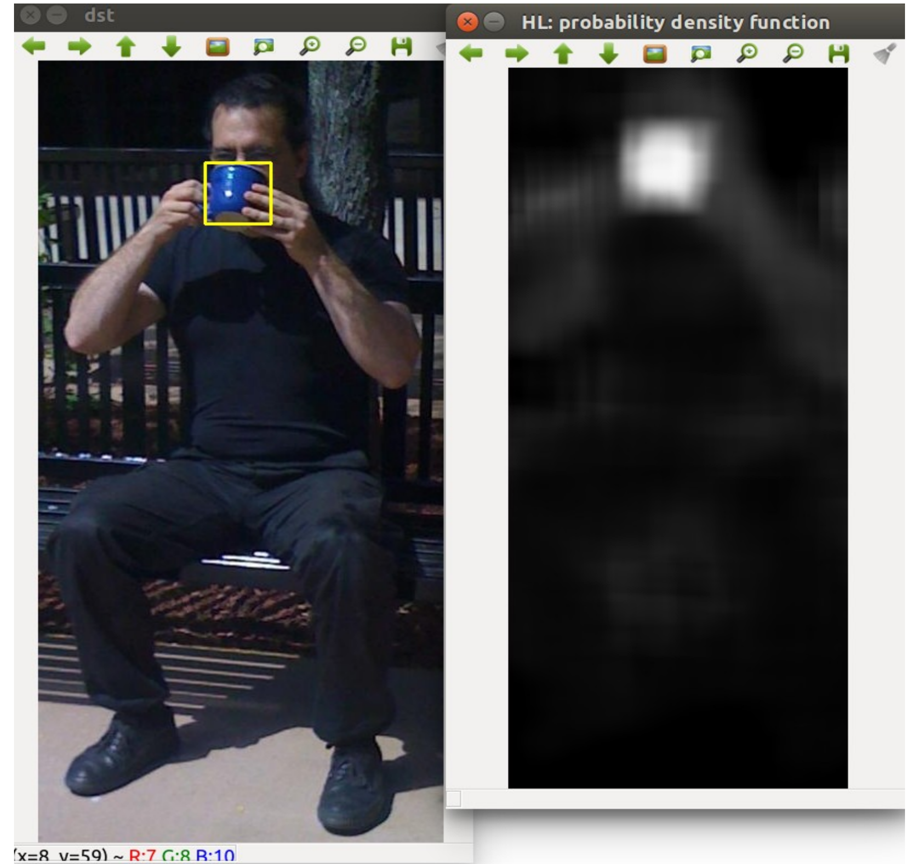
Result image is smaller

April 24 BPP Implementation

```
/*------------------------------------------------------------------------*
 * Program: calBackProjectPatch.cpp;  coded by:                           *
 * Modified by: HL, CTI Plus Corporation, copyrighted;                    *
 * Date:        April 23, 2018;  Version: 0x1.0;                          *
 * Status:      tested;                                                   *
 * Compile and build: CMakeLists.txt,                                     *
 * $cmake .                                                               *
 * $make                                                                  *
 *------------------------------------------------------------------------*/
#include <opencv2/opencv.hpp>  //HL: opencv2/opencv.hpp
#include <highgui.h>

void GetHSV (const IplImage* image, IplImage** h,
                  IplImage** s, IplImage** v);
int main()
{
   //IplImage* src = cvLoadImage ("bluecup.jpg", 1); //patch

   IplImage* src = cvLoadImage ("colorPatch1.jpg", 1); //patch
   IplImage* h_src = NULL;                        //define h plane
   IplImage* s_src = NULL;                        //define s plane
   GetHSV (src, &h_src, &s_src, NULL);    //just h-s planes
   IplImage *images[] = {h_src,s_src};
   CvHistogram* hist_src = NULL;
```



Result image is smaller

Full test code see github posting,
source: openCV

Harry Li, Ph.D.

Appendix Color Hist 4 Matching Method

1. Compare the histogram representation of the colors in images. Flesh tones are often easier to pick out in HSV space.
2. Restricting to the H (hue) and S (saturation) planes is sufficient for the recognition of flesh tones across groups.
3. lighting can cause severe mismatches in color.
4. Sometimes normalized BGR works better than HSV when lighting changes.

Build HS plane histogram

Table 7-1. Histogram comparison of 4 matching methods

Comparison	CORREL	CHISQR	INTERSECT	BHATTACHARYYA
Indoor lower half	0.96	0.14	0.82	0.2
Outdoor shade	0.09	1.57	0.13	0.8
Outdoor sun	–0.0	1.98	0.01	0.99