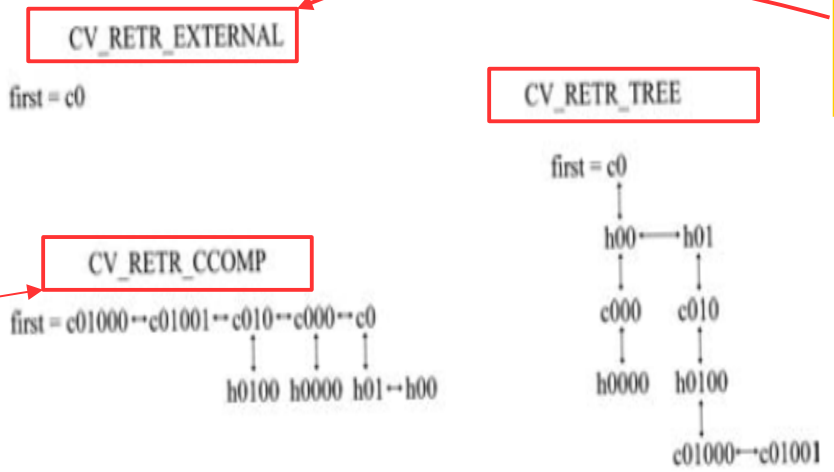
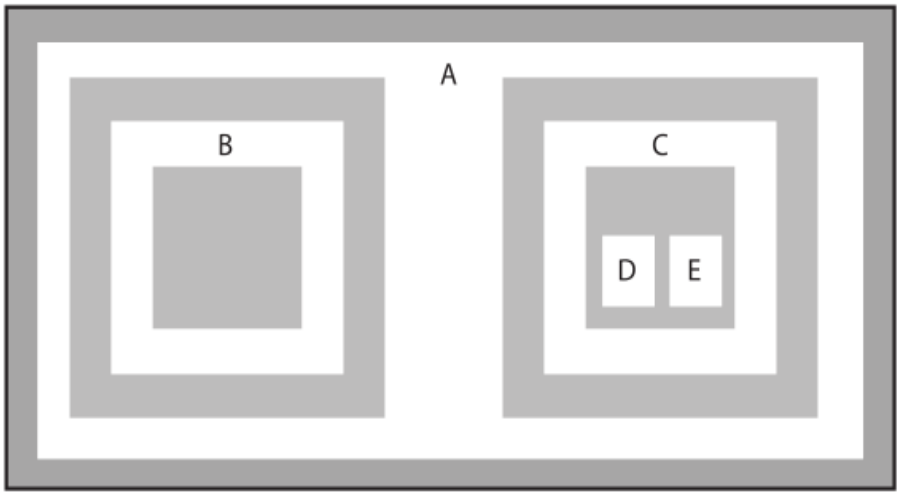
9-9-2018 Revisit Contours Models

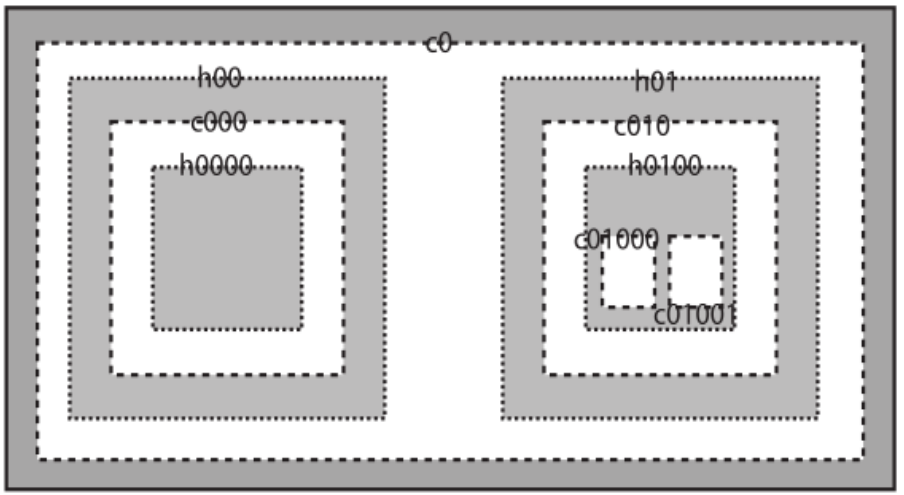


Retrieves only the extreme outer contours. It sets hierarchy[i][2]=hierarchy[i][3]=-1 for all the contours.

CV_RETR_CCOMP Retrieves all the contours into two-level hierarchy, top-level for external boundaries and the 2nd level for the holes.

retrieves all contours without any hierarchical relationships.





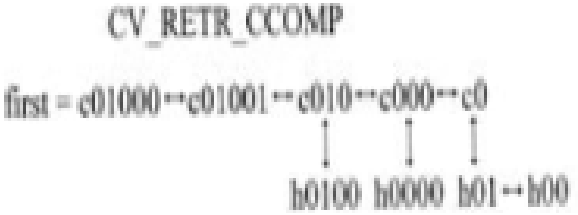
http://opencvexamples.blogspot.com/2013/09/find-contour.html

9-12-2018 Revisit Contours Hierarchy

```
findContours( temp, contours, hierarchy,
                    RETR_CCOMP,
                    CHAIN_APPROX_SIMPLE );

 for( ; idx >= 0; idx = hierarchy[idx][0] )
 {
     const vector<Point>& c = contours[idx];
     double area = fabs(contourArea(Mat(c)));
     if( area > maxArea )
     {
         maxArea = area;
         largestComp = idx;
     }
 }
 Scalar color( 0, 0, 255 );
 drawContours( dst, contours, largestComp, color,
FILLED, LINE_8, hierarchy );
```

Note: 1 hierarchy[idx][0]



From example figure 1, first element, e.g., “0” in hierarchy[idx][0] defines the level for c-type contours (no holes) when use RETR_CCOMP

Note 2: assign all points of a contour to a vector

```
const vector<Point>& c = contours[idx];
```

Note 3: find an area of a contour

```
const vector<Point>& c = contours[idx];

Mat A = Mat(c)
double area = fabs(contourArea(Mat(c)));
```


http://opencvexamples.blogspot.com/2013/09/find-contour.html

9-12-2018 Contours_CCOMP To Individual Patterns

contours_CCOMP[i]

c011
[170, 103;
168, 105;
168, 134;
170, 136;
189, 136;
191, 134;
191, 105;
189, 103]

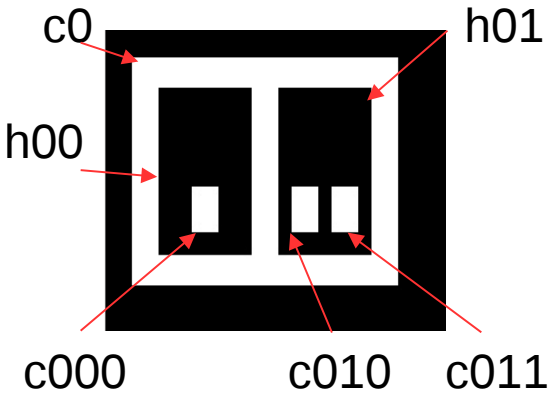
c0
[20, 18;
18, 20;
18, 169;
20, 171;
219, 171;
221, 169;
221, 20;
219, 18]

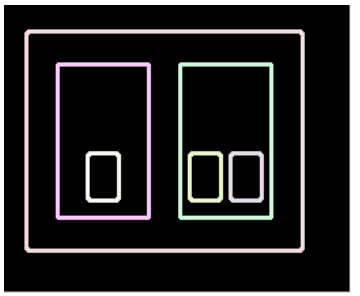
c010
[140, 103;
138, 105;
138, 134;
140, 136;
159, 136;
161, 134;
161, 105;
159, 103]

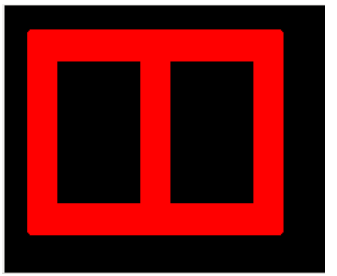
h01
[131, 42;
132, 41;
197, 41;
198, 42;
198, 147;
197, 148;
132, 148;
131, 147]

c000
[65, 103;
63, 105;
63, 134;
65, 136;
84, 136;
86, 134;
86, 105;
84, 103]

h00
[41, 42;
42, 41;
107, 41;
108, 42;
108, 147;
107, 148;
42, 148;
41, 147]







```
Mat frame(200, 256, CV_8UC3, Scalar(0, 0, 0));  //x width, y height
Rect ct1_rect(20, 30, 200, 150); // 1st level
 rectangle(frame, ct1_rect, Scalar(255, 255, 255), FILLED);
 Rect ct2_rect(40, 50, 70, 110);  // 2nd level
 rectangle(frame, ct2_rect, Scalar(0, 0, 0), FILLED);
 Rect ct3_rect(130, 40, 70, 110);
 rectangle(frame, ct3_rect, Scalar(0, 0, 0), FILLED);
 Rect ct4_rect(65, 105, 20, 30);  // 3rd level
 rectangle(frame, ct4_rect, Scalar(255, 255, 255), FILLED);
 Rect ct5_rect(140, 105, 20, 30);
 rectangle(frame, ct5_rect, Scalar(255, 255, 255), FILLED);
 Rect ct6_rect(170, 105, 20, 30);
 rectangle(frame, ct6_rect, Scalar(255, 255, 255), FILLED);
```

c0
h00
h01
c000
c010
c011

c011 → c010 → c000 → c0 → h01 → h00

```
for(int i=0;i<contours_CCOMP.size();i++) {
  cout<<contours_CCOMP[i]<<endl;
 }
```

9-13-2018 Contours_TREE To Individual Patterns

contours_TREE[i]

c011 → [170, 103;
168, 105;
168, 134;
170, 136;
189, 136;
191, 134;
191, 105;
189, 103]

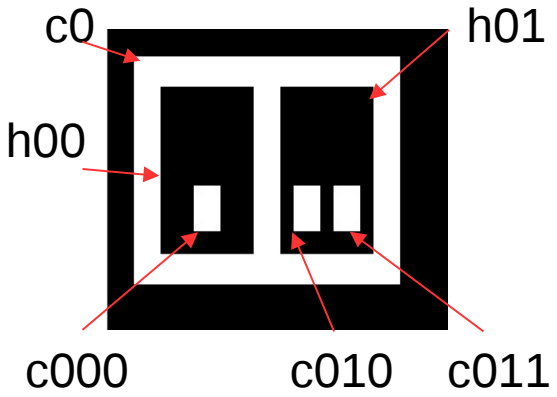
c0 → [20, 18;
18, 20;
18, 169;
20, 171;
219, 171;
221, 169;
221, 20;
219, 18]

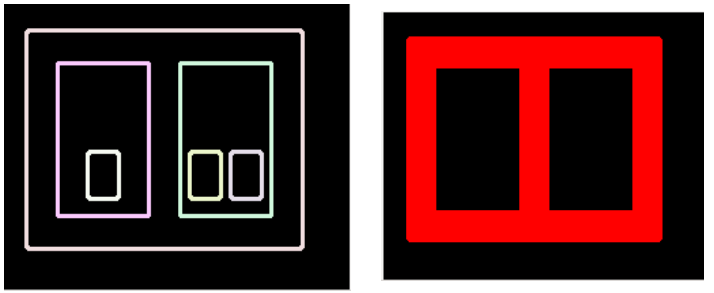
c010 → [140, 103;
138, 105;
138, 134;
140, 136;
159, 136;
161, 134;
161, 105;
159, 103]

h01 → [131, 42;
132, 41;
197, 41;
198, 42;
198, 147;
197, 148;
132, 148;
131, 147]

c000 → [65, 103;
63, 105;
63, 134;
65, 136;
84, 136;
86, 134;
86, 105;
84, 103]

h00 → [41, 42;
42, 41;
107, 41;
108, 42;
108, 147;
107, 148;
42, 148;
41, 147]

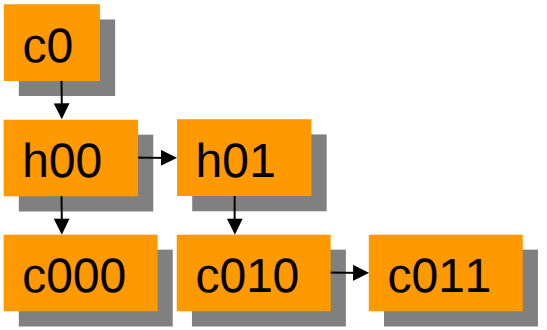




```
Mat frame(200, 256, CV_8UC3, Scalar(0, 0, 0));  //x width, y height
Rect ct1_rect(20, 30, 200, 150); // 1st level
 rectangle(frame, ct1_rect, Scalar(255, 255, 255), FILLED);
 Rect ct2_rect(40, 50, 70, 110);  // 2nd level
 rectangle(frame, ct2_rect, Scalar(0, 0, 0), FILLED);
 Rect ct3_rect(130, 40, 70, 110);
 rectangle(frame, ct3_rect, Scalar(0, 0, 0), FILLED);
 Rect ct4_rect(65, 105, 20, 30);  // 3rd level
 rectangle(frame, ct4_rect, Scalar(255, 255, 255), FILLED);
 Rect ct5_rect(140, 105, 20, 30);
 rectangle(frame, ct5_rect, Scalar(255, 255, 255), FILLED);
 Rect ct6_rect(170, 105, 20, 30);
 rectangle(frame, ct6_rect, Scalar(255, 255, 255), FILLED);
```

c0
h00
h01
c000
c010
c011

```
for(int i=0;i<contours_TREE.size();i++){
    cout<<contours_TREE[i]<<endl;
}
```



Harry Li, Ph.D.

9-13-2018 Hierarchy Results on the Terminal

```
hierarchy_TREE:
countour 0 [ne, pr, ch, pa]: [-1, -1, 1, -1]
countour 1 [ne, pr, ch, pa]: [4, -1, 2, 0]
countour 2 [ne, pr, ch, pa]: [3, -1, -1, 1]
countour 3 [ne, pr, ch, pa]: [-1, 2, -1, 1]
countour 4 [ne, pr, ch, pa]: [-1, 1, 5, 0]
countour 5 [ne, pr, ch, pa]: [-1, -1, -1, 4]

hierarchy_CCOMP:
countour 0 [ne, pr, ch, pa]: [1, -1, -1, -1]
countour 1 [ne, pr, ch, pa]: [2, 0, -1, -1]
countour 2 [ne, pr, ch, pa]: [3, 1, -1, -1]
countour 3 [ne, pr, ch, pa]: [-1, 2, 4, -1]
countour 4 [ne, pr, ch, pa]: [5, -1, -1, 3]
countour 5 [ne, pr, ch, pa]: [-1, 4, -1, 3]
```

The contour hierarchies follow this format:

contour [index]: [next, previous, 1st child, parent]

'-1' means N/A.

For example:

```
countour 1 [ne, pr, ch, pa]: [4, -1, 2, 0]
```

The contour 1 has:

- Contour 4 is the next contour in the same level.
- No previous contour in the same level.
- First child is contour 2.
- Parent is contour 0

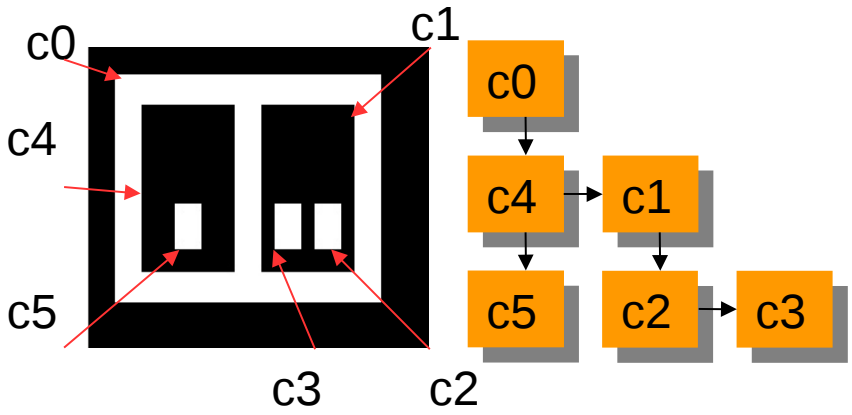


Figure 1. hierarchy_TREE

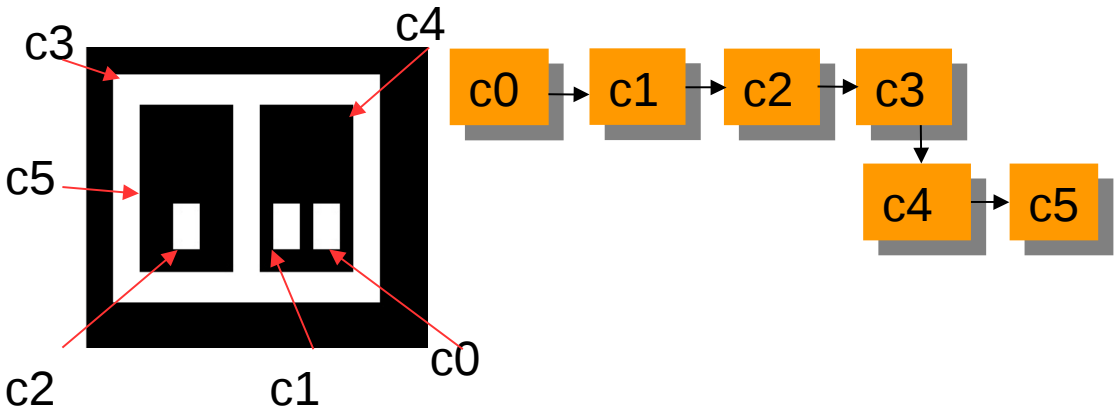


Figure 2. hierarchy_CCOMP

9-12-2018 Contours To Paint Individual Patterns

It is (a) dog

Noun + verb + Noun

Subject + verb + ???

Contour + ??? + holes/contours

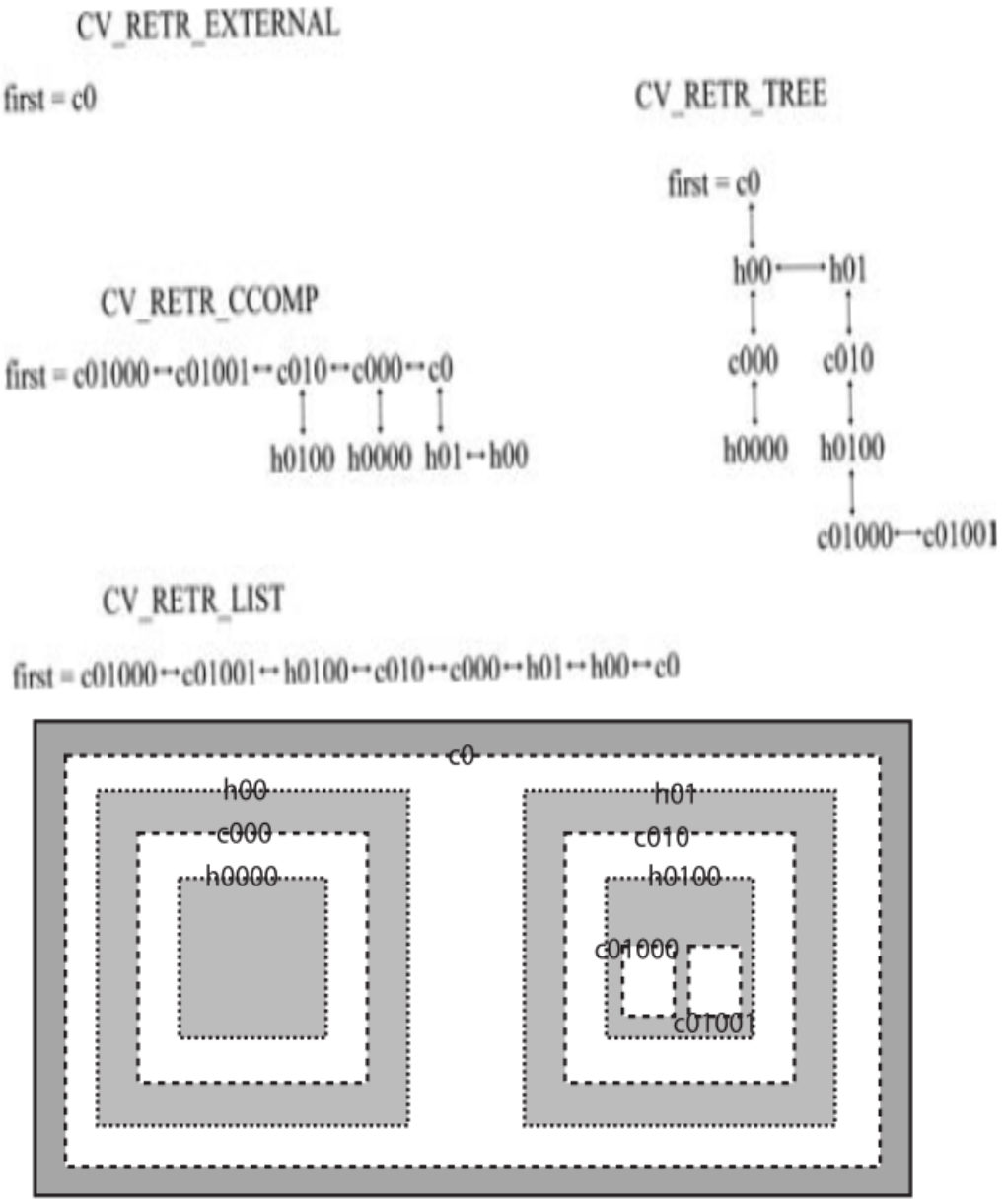
9-12-2018 Contours To Grammar

Harry Li, Ph.D.

9-12-2018 Contour Hierarchy

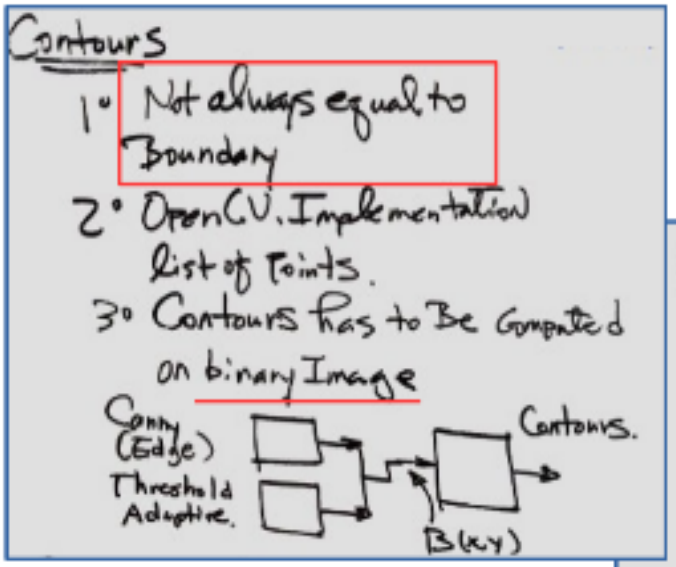
The hierarchy form:
hierarchy[idx][{0,1,2,3}]={next contour (same level), previous contour (same level), child contour, parent contour}

CV_RETR_CCOMP, returns a hierarchy of outer contours and holes. This means elements 2 and 3 of hierarchy[idx] have at most one of these not equal to -1: that is, each element has either no parent or child, or a parent but no child, or a child but no parent.
An element with a parent but no child would be a boundary of a hole.
That means you basically go through hierarchy[idx] and draw anything with hierarchy[idx][3]>-1.
Something like (works in Python, but haven't tested the C++. Idea is fine though.):



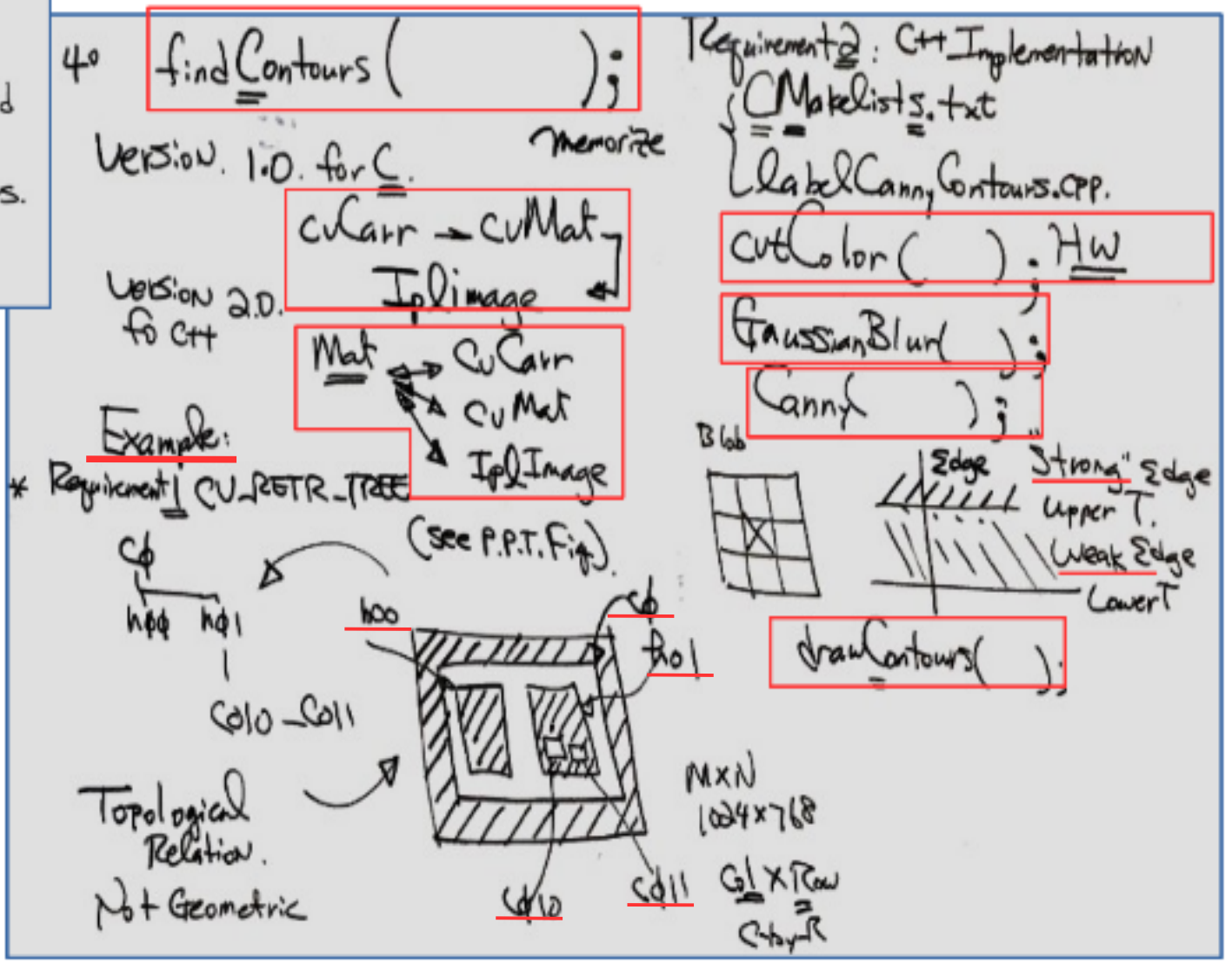
http://opencvexamples.blogspot.com/2013/09/find-contour.html

9-8-2018 Contours Trees



Topological relationship of each contour and mapping from a given contour image to tree structure

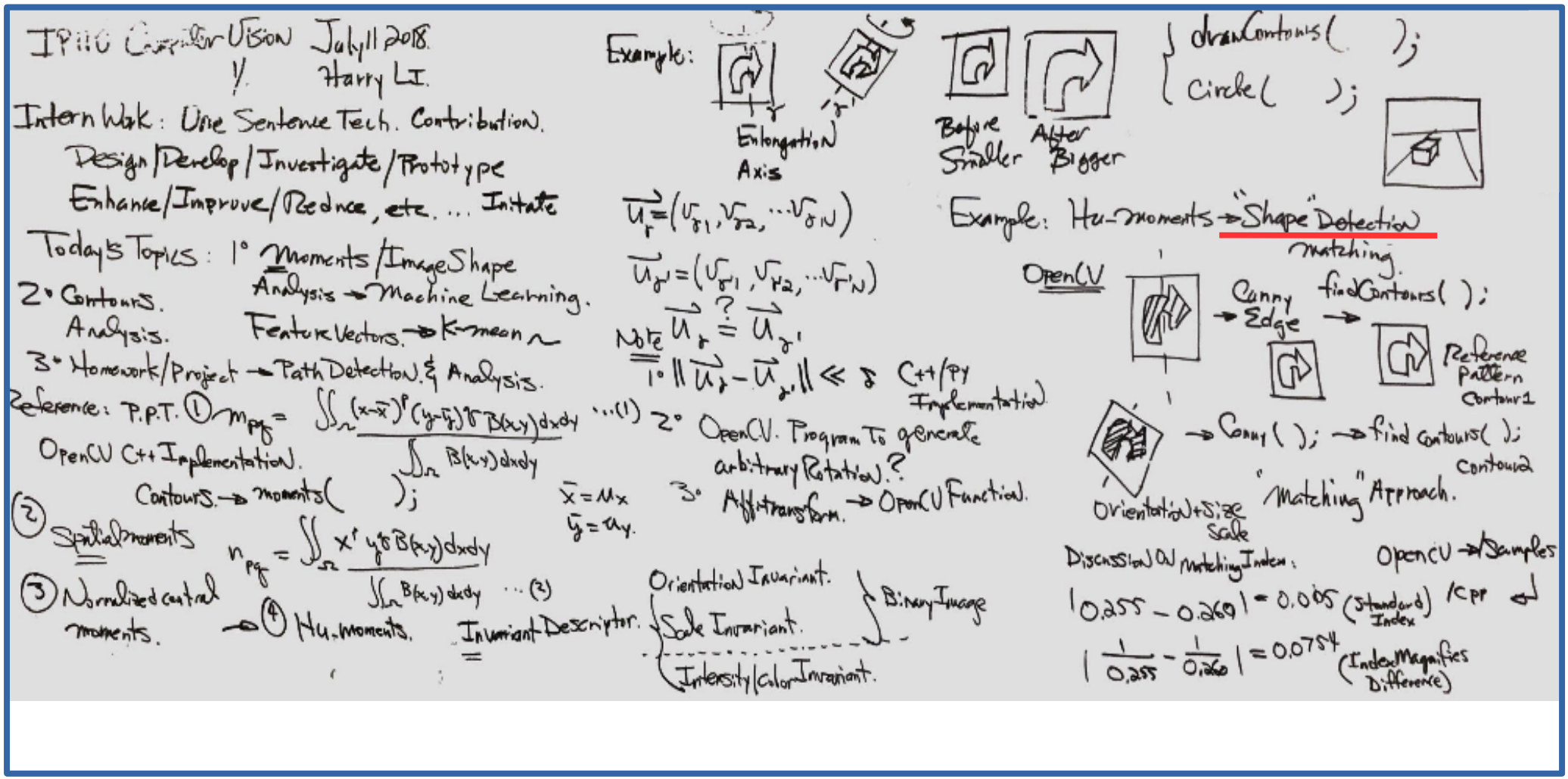
findContour( ); only works on binary image, one of the 3 images, Canny, Threshold and adaptiveThreshold



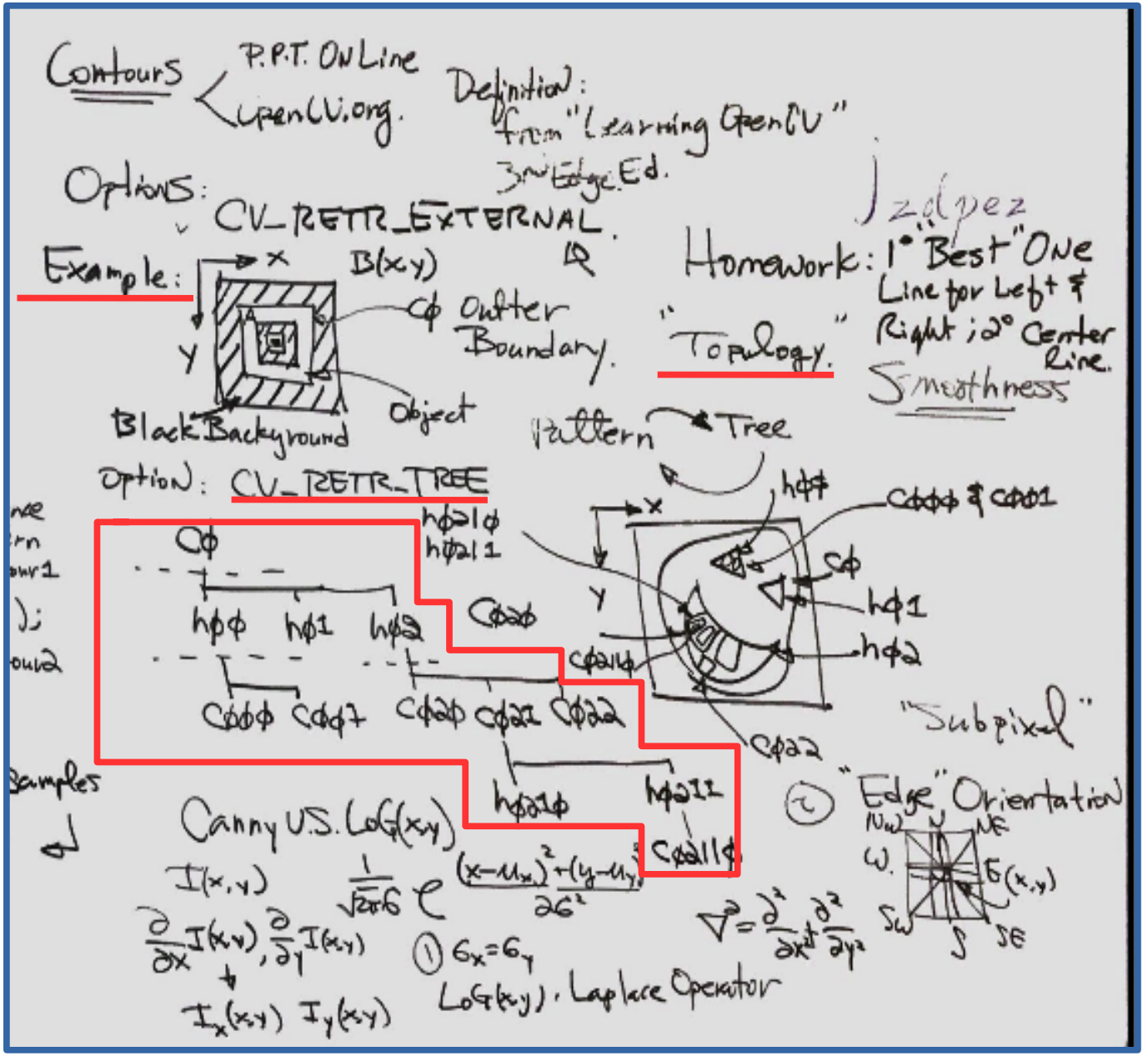
9-8-2018 Remove Segmentation

Define asymmetric structure element as follow to remove the vertical feature of the binary image

7-11-2018 Invariant Concept



7-11-2018 Invariant Concept



9-11-2018 C++ Code Testing

1. C++ code: generate image MAT frame as 256 by 200 and initialize it to all 0s.

Mat frame(200, 256, CV_8UC3, Scalar(0, 0, 0));

2. Then use RECT and rectangle() function to generate a white rectangle (200 by 150) as in the example on the right:

Rect ct1_rect(20, 20, 200, 150);
rectangle(frame, ct1_rect, Scalar(255, 255, 255), FILLED);

3. Then use RECT and rectangle() function to generate 2 black rectangles (70 by 110) inside the white rectangle:

Rect ct2_rect(40, 40, 70, 110);
rectangle(frame, ct2_rect, Scalar(0, 0, 0), FILLED);
Rect ct3_rect(130, 40, 70, 110);
rectangle(frame, ct3_rect, Scalar(0, 0, 0), FILLED);

4. Then use RECT and rectangle() function to generate 3 white rectangles (20 by 30) inside the 2 black rectangles as in the example on the right, then imshow()

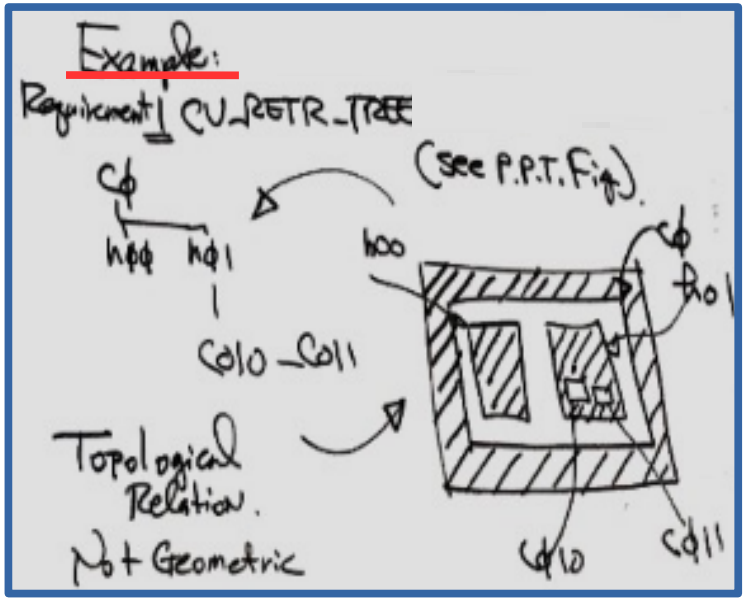
Rect ct4_rect(65, 105, 20, 30);
rectangle(frame, ct4_rect, Scalar(255, 255, 255), FILLED);
Rect ct5_rect(140, 105, 20, 30);
rectangle(frame, ct5_rect, Scalar(255, 255, 255), FILLED);
Rect ct6_rect(175, 105, 20, 30);
rectangle(frame, ct6_rect, Scalar(255, 255, 255), FILLED);

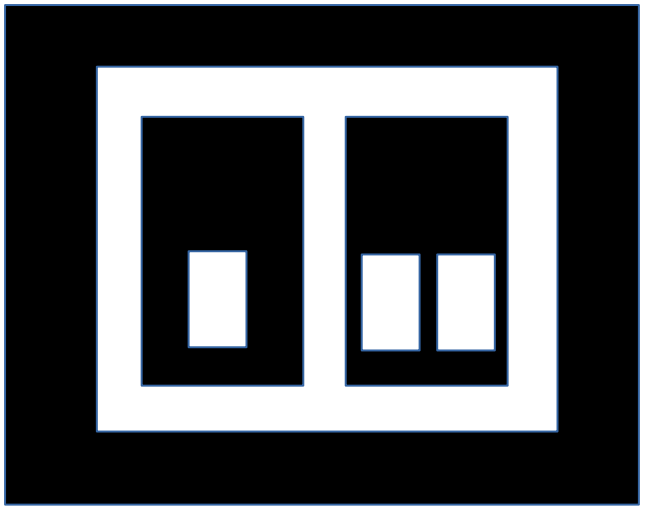
5. findContours( ) with option as CV_RETR_CCOMP

6. print each contour starting (x,y) points;

7. print hierarchical structure to match hand calculation and to verify the contours;

8. perform semantical segmentation based on the result in 7, to remove any one of the

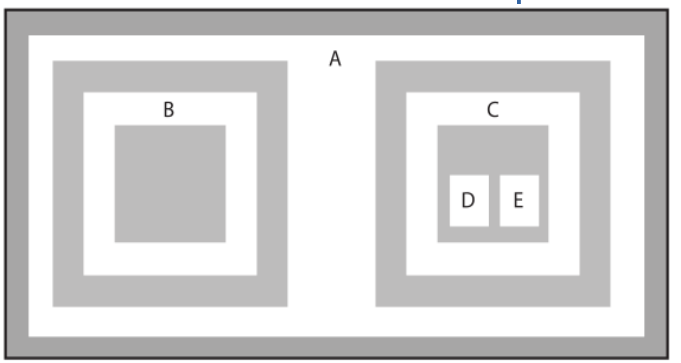




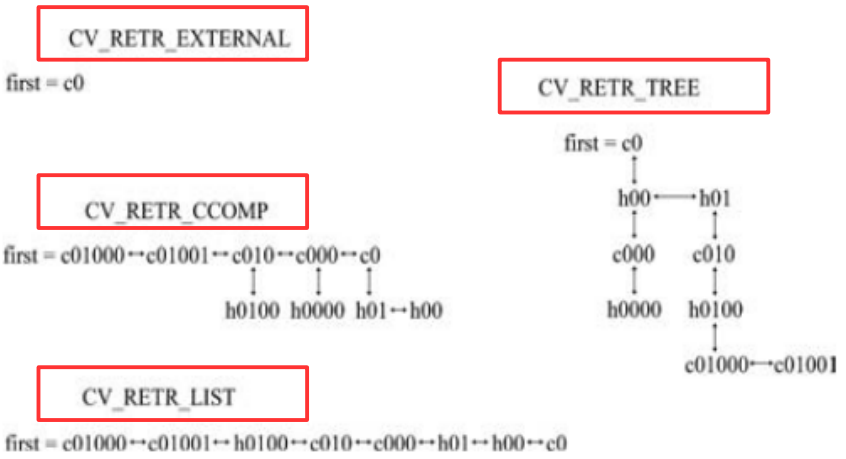
Contours Mode Variable

findContours(image_canny, contours, hierarchy,
RETR_CCOMP, CHAIN_APPROX_SIMPLE);

The mode variable can be set to any of four options: CV_RETR_EXTERNAL, CV_RETR_LIST, CV_RETR_CCOMP, or CV_RETR_TREE. The value of mode indicates to cvFindContours() exactly what contours we would like found and how we would like the result presented to us. In particular, the manner in which the tree node variables (h_prev, h_next, v_prev, and v_next) are used to "hook up" the found contours is determined by the value of mode. In Figure 8-3, the resulting topologies are shown for all four possible values of mode. In every case, the structures can be thought of as "levels" which are related by the "horizontal" links (h_next and h_prev), and those levels are separated from one another by the "vertical" links (v_next and v_prev).

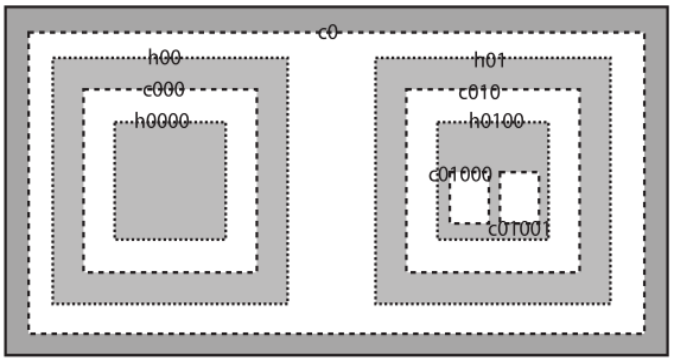


Retrieves only the extreme outer contours. It sets hierarchy[i][2]=hierarchy[i][3]=-1 for all the contours.



CV_RETR_CCOMP
Retrieves all the contours into two-level hierarchy, top-level for external boundaries and the 2nd level for the holes.

retrieves all contours without any hierarchical relationships.



http://opencvexamples.blogspot.com/2013/09/find-contour.html

Contours

actly what a contour is. A contour is a list of points that represent, in one way or another, a curve in an image. This representation can be different depending on the circumstance at hand. There are many ways to represent a curve. Contours are represented in OpenCV by sequences in which every entry in the sequence encodes information about the location of the next point on the curve. We will dig into the details of such

Reference: Learning OpenCV, pp. 250

The function cvFindContours() computes contours from binary images. It can take images created by cvCanny(), which have edge pixels in them, or images created by functions like cvThreshold() or cvAdaptiveThreshold(), in which the edges are implicit as boundaries between positive and negative regions.*

Contours Mode Variable

http://opencvexamples.blogspot.com/2013/09/find-contour.html

```
void drawContours(
        InputOutputArray image,
        InputArrayOfArrays contours,
        int contourIdx,
        const Scalar& color,
        int thickness=1,
        int lineType=8,
        InputArray hierarchy=noArray(),
        int maxLevel=INT_MAX,
        Point offset=Point() )
```

hierarchy – Output vector, containing contour topology. It has as many elements as the number of contours. For each i-th contour contours[i] , the elements hierarchy[i][0] , hiearchy[i][1] , hiearchy[i][2] , and hiearchy[i][3] are set to 0-based indices in contours of the next and previous contours at the same hierarchical level, the first child contour and the parent contour, respectively. If for the contour i there are no next, previous, parent, or nested contours, the corresponding elements of hierarchy[i] will be negative.

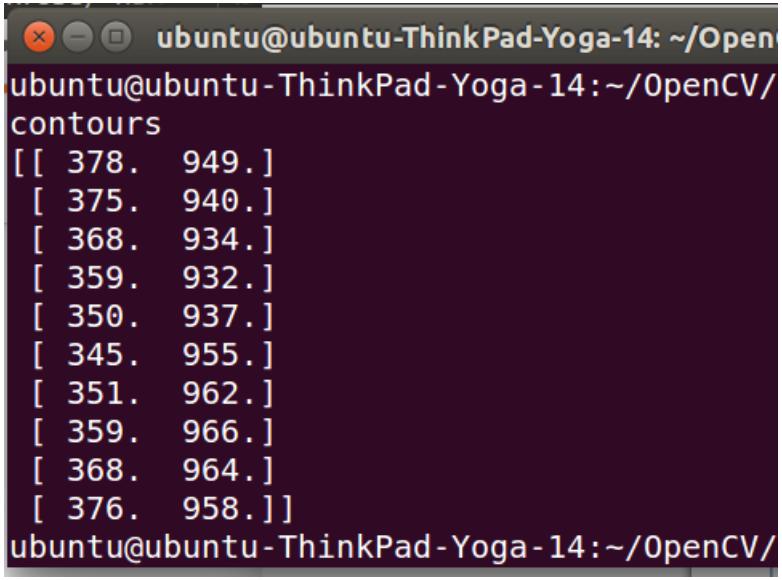
contourIdx – Parameter indicating a contour to draw. If it is negative, all the contours are drawn.

maxLevel – Maximal level for drawn contours. If it is 0, only the specified contour is drawn. If it is 1, the function draws the contour(s) and all the nested contours. If it is 2, the function draws the contours, all the nested contours, all the nested-to-nested contours, and so on. This parameter is only taken into account when there is hierarchy available.

Contours Data Type In Python

https://stackoverflow.com/questions/20928944/create-contour-from-scratch-in-python-opencv-cv2

```
#-------------------------------------#
# program: contour-test.py     #
# tested by: HL                #
import cv2, numpy
contour = numpy.array( [
(378, 949), (375, 940), (368, 934),
(359, 932), (350, 937), (345, 955),
(351, 962), (359, 966), (368, 964),
(376, 958) ], numpy.float32 )
cv2.isContourConvex(contour)
print ('contours')
print (contour)
```



Compute Contours Features

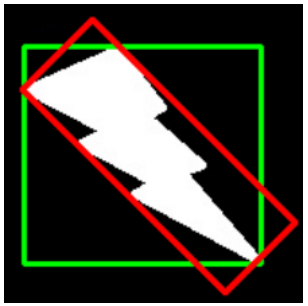
https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html

1. Moments

```
1 import cv2
2 import numpy as np
3
4 img = cv2.imread('star.jpg',0)
5 ret,thresh = cv2.threshold(img,127,255,0)
6 contours,hierarchy = cv2.findContours(thresh, 1, 2)
7
8 cnt = contours[0]
9 M = cv2.moments(cnt)
10 print M
```

2. Contour Area

```
area = cv2.contourArea(cnt)
```



3. Contour Perimeter

```
perimeter = cv2.arcLength(cnt,True)
```

4. Contour Approximation

```
1 epsilon = 0.1*cv2.arcLength(cnt,True)
2 approx = cv2.approxPolyDP(cnt,epsilon,True)
```

5. Convex Hull

Convexity defects

checks a curve for convexity defects and corrects it

```
hull = cv2.convexHull(cnt)
```

6. Checking Convexity

```
k = cv2.isContourConvex(cnt)
```

7.a. Straight Bounding Rectangle

```
1 x,y,w,h = cv2.boundingRect(cnt)
2 cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
```

7.b. Rotated Rectangle

```
1 rect = cv2.minAreaRect(cnt)
2 box = cv2.boxPoints(rect)
3 box = np.int0(box)
4 cv2.drawContours(img,[box],0,(0,0,255),2)
```

Compute Contours Features

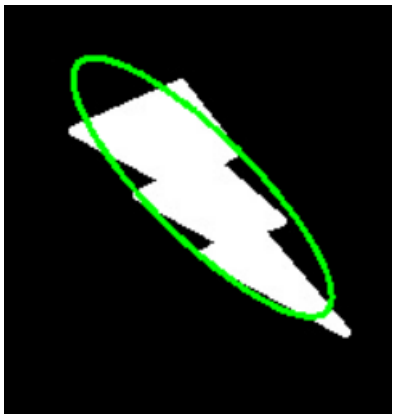
https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html

8. Minimum Enclosing Circle

```
1 (x,y),radius = cv2.minEnclosingCircle(cnt)
2 center = (int(x),int(y))
3 radius = int(radius)
4 cv2.circle(img,center,radius,(0,255,0),2)
```

9. Fitting an Ellipse

```
1 ellipse = cv2.fitEllipse(cnt)
2 cv2.ellipse(img,ellipse,(0,255,0),2)
```



http://nicky.vanforeest.com/misc/fitEllipse/fitEllipse.html

10. Fitting a Line

```
1 rows,cols = img.shape[:2]
2 [vx,vy,x,y] = cv2.fitLine(cnt, cv2.DIST_L2,0,0.01,0.01)
3 lefty = int((-x*vy/vx) + y)
4 righty = int(((cols-x)*vy/vx)+y)
5 cv2.line(img,(cols-1,righty),(0,lefty),(0,255,0),2)
```

From Contour Find Shapes

https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html

4. Contour Approximation

```
1 epsilon = 0.1*cv2.arcLength(cnt,True)
2 approx = cv2.approxPolyDP(cnt,epsilon,True)
```

5. Convex Hull Convexity defects

checks a curve for convexity defects and corrects it

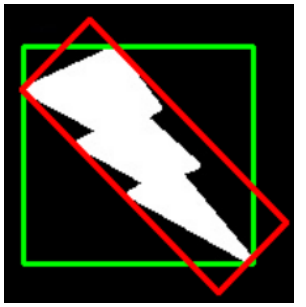
```
hull = cv2.convexHull(cnt)
```

7.a. Straight Bounding Rectangle

```
1 x,y,w,h = cv2.boundingRect(cnt)
2 cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
```

7.b. Rotated Rectangle

```
1 rect = cv2.minAreaRect(cnt)
2 box = cv2.boxPoints(rect)
3 box = np.int0(box)
4 cv2.drawContours(img,[box],0,(0,0,255),2)
```

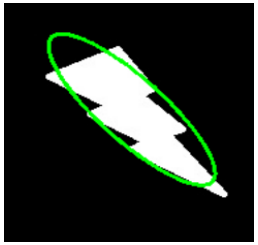


8. Minimum Enclosing Circle

```
1 (x,y),radius = cv2.minEnclosingCircle(cnt)
2 center = (int(x),int(y))
3 radius = int(radius)
4 cv2.circle(img,center,radius,(0,255,0),2)
```

9. Fitting an Ellipse

```
1 ellipse = cv2.fitEllipse(cnt)
2 cv2.ellipse(img,ellipse,(0,255,0),2)
```



Harry Li, Ph.D

Contour-Shapes Properties

http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_contours/py_contour_properties/py_contour_properties.html

11. Aspect Ratio

$$Aspect\ Ratio = \frac{Width}{Height}$$

```
x,y,w,h = cv2.boundingRect(cnt)
aspect_ratio = float(w)/h
```

12. Extent

$$Extent = \frac{Object\ Area}{Bounding\ Rectangle\ Area}$$

```
area = cv2.contourArea(cnt)
x,y,w,h = cv2.boundingRect(cnt)
rect_area = w*h
extent = float(area)/rect_area
```

13. Solidity

$$Solidity = \frac{Contour\ Area}{Convex\ Hull\ Area}$$

```
area = cv2.contourArea(cnt)
hull = cv2.convexHull(cnt)
hull_area = cv2.contourArea(hull)
solidity = float(area)/hull_area
```

14. Equivalent Diameter

$$Equivalent\ Diameter = \sqrt{\frac{4 \times Contour\ Area}{\pi}}$$

```
area = cv2.contourArea(cnt)
equi_diameter = np.sqrt(4*area/np.pi)
```

15. Orientation

Following method also gives the Major Axis and Minor Axis lengths.

```
(x,y),(MA,ma),angle = cv2.fitEllipse(cnt)
```

Contour Mask And Pixel Points

http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_contours/py_contour_properties/py_contour_properties.html

```
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(imgray,mask = mask)
```

16. Mask and Pixel Points

All the points comprises that object (contour)

```
mask = np.zeros(imgray.shape,np.uint8)
cv2.drawContours(mask,[cnt],0,255,-1)
pixelpoints = np.transpose(np.nonzero(mask))
#pixelpoints = cv2.findNonZero(mask)
```

Above, “two methods, one using Numpy functions, next one using OpenCV function (last commented line) are given to do the same. Results are also same, but with a slight difference. Numpy gives coordinates in (row, column) format, while OpenCV gives coordinates in (x,y) format. So basically the answers will be interchanged. Note that, row = x and column = y.”

17. Maximum Value, Minimum Value and their locations

18. Mean Color or Mean Intensity

```
mean_val = cv2.mean(im,mask = mask)
```

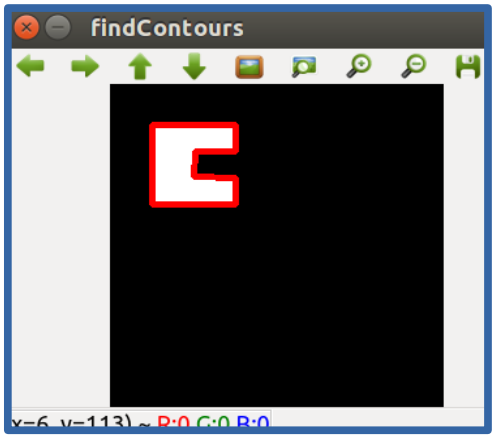
19. Extreme Points

```
leftmost = tuple(cnt[cnt[:,:,0].argmin()][0])
rightmost = tuple(cnt[cnt[:,:,0].argmax()][0])
topmost = tuple(cnt[cnt[:,:,1].argmin()][0])
bottommost = tuple(cnt[cnt[:,:,1].argmax()][0])
```

Contour Attributes

```
_,contours,hierarchy = cv2.findContours(thresh, /
                cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
contours = [cv2.approxPolyDP(cnt, 3, True) for cnt in contours]

cv2.drawContours(img, contours, -1, (0,0,255),3)
```



Inference Engine

1

Table 1. Attribute Table

	Ceiling Lights (class w1)	Window Lights (class w2)
Shape	Rectangles x1	Rectangles x1
	Ellipses x2	Ellipses x2
	Circles x3	Circles x3
Location	Anywhere x4 smaller part image x5	Anywhere x4 smaller part image x5
Color	white x6	white x6
Repeated Pattern	maybe x7	maybe x7

3

So decision function

$$f(X) = x1\ x5\ x6 + x2\ x6 + x3\ x6 + x5\ x6 \quad \ldots (1)$$

4

C/c++ implementation of the inference engine (switching function)

2

Table 2. Identification Table

	x1 rect	x2 elli	x3 cir	x4 loc	x5 sml	x6 wht	x7 rep	f(X)
x1 x5 x6	1	D	D	D	1	1	D	1
x2 x6	D	1	D	D	D	1	D	1
x3 x6	D	D	1	D	D	1	D	1
x5 x6	D	D	D	D	1	1	D	1

Define primary implicant, removal of any of its column will result in the mis-identification of f(X)

No: C/C++ Inference Engine

```
#include<stdio.h>
int And(int a, int b);
int Or(int a, int b);
int Not(int a);
void main()
{
 ///where main body of code will go
}
int And(int a, int b)
{
 int output;
 if(a==0 && b==0)
  output=0;
  if(a==1 && b==0)
  output=0;
 if(a==0 && b==1)
  output=0;
 if(a==1 && b==1)
  output=1;
 return (output);
}
```

Simplify it 1. as boolean; 2. logically as &&

```
int Or(int a, int b)
{
 int output;
 if(a==0 && b==0)
  output=0;
  if(a==1 && b==0)
  output=1;
 if(a==0 && b==1)
  output=1;
 if(a==1 && b==1)
  output=1;
 return (output);
}
int Not(int a)
{
int output;
 if(a==0 )
  output=1;
 if(a==1 )
  output=0;
  return (output);
}
```

In fact C/C++ support all the boolean logic operators, so build inference engine should be straight forward

Simplify it 1. as boolean;

```
int And(int a, int b)
{
    return a && b;
}
```

```
return Not(And(a, b));
```

Build NAND, NOR, XOR etc

C/C++ Bitwise Operators

Operators	Meaning of operators
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
~	Bitwise complement
<<	Shift left
>>	Shift right

```
// C Program to demonstrate the working of logical operators
#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10, result;
    result = (a == b) && (c > b);
    printf("(a == b) && (c > b) equals to %d \n", result);

    result = (a == b) && (c < b);
    printf("(a == b) && (c < b) equals to %d \n", result);

    result = (a == b) || (c < b);
    printf("(a == b) || (c < b) equals to %d \n", result);

    result = (a != b) || (c < b);
    printf("(a != b) || (c < b) equals to %d \n", result);

    result = !(a != b);
    printf("!(a == b) equals to %d \n", result);

    result = !(a == b);
    printf("!(a == b) equals to %d \n", result);
    return 0;
}
```

C/C++ Inference Engine

```
//-----Inference Engine to find reflection spots---//
//-----April 7, 2018, by HL, version 0x0.1; --------//
 #include <stdio.h>
#include <stdbool.h>
#define dimension 100
bool    x[dimension], f_identification;
int     item;

int main()
{
    printf("Inference Engine to identify reflections \n");
    printf("x1 rectangle? 1 for Y or 0 for N \n");
    scanf("%i",&item);
    if (item == 1) x[1] = true;
    if (item == 0) x[1] = false;

    printf("x2 ellips? 1 for Y or 0 for N \n");
    scanf("%i",&item);
    if (item == 1) x[2] = true;
    if (item == 0) x[2] = false;

    printf("x3 circle? 1 for Y or 0 for N \n");
    scanf("%i",&item);
    if (item == 1) x[3] = true;
    if (item == 0) x[3] = false;

    printf("x4 location? 1 for Y or 0 for N \n");
    scanf("%i",&item);
    if (item == 1) x[4] = true;
    if (item == 0) x[4] = false;
```

```
 printf("x5 small size? 1 for Y or 0 for N \n");
    scanf("%i",&item);
    if (item == 1) x[5] = true;
    if (item == 0) x[5] = false;

    printf("x6 white color? 1 for Y or 0 for N \n");
    scanf("%i",&item);
    if (item == 1) x[6] = true;
    if (item == 0) x[6] = false;

    printf("x7 repetative? 1 for Y or 0 for N \n");
    scanf("%i",&item);
    if (item == 1) x[7] = true;
    if (item == 0) x[7] = false;

    f_identification = (x[1] && x[5] && x[6])
                    || (x[2] && x[6])
                    || (x[3] && x[6])
                    || (x[5] && x[6]);

    if (f_identification){
    printf("The object is reflection\n");}
    else {
    printf("The object is not reflection\n");}

    return 0;
}
```

OpenCV Contours For Shapes

Table 3 (based on Table 2) openCV functions

	x1 rect	x2 elli	x3 cir	x4 loc	x5 sml	x6 wht	x7 rep
x1 x5 x6	1	D	D	D	1	1	D
x2 x6	D	1	D	D	D	1	D
x3 x6	D	D	1	D	D	1	D
x5 x6	D	D	D	D	1	1	D

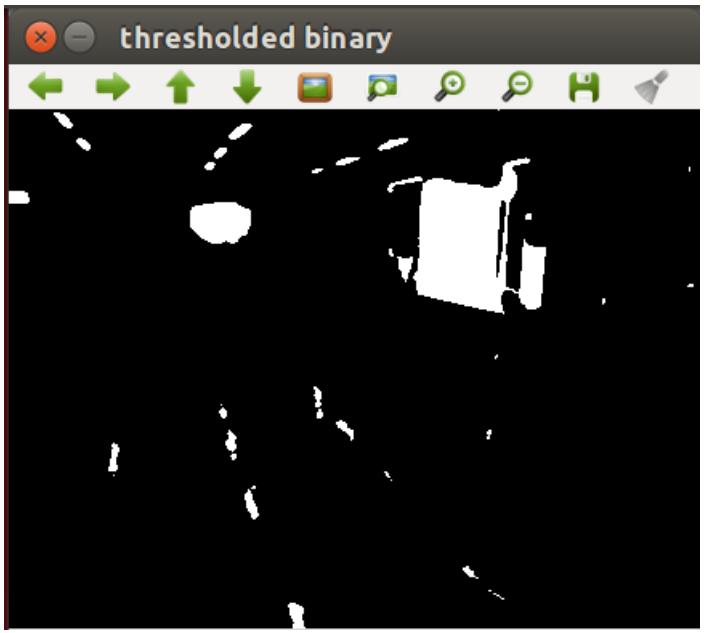
Rectangle detection (size, location and color, as well as total number);
ellips detection (size, location and color, as well as total number);
Circle detection (size, location and color, as well as total number);

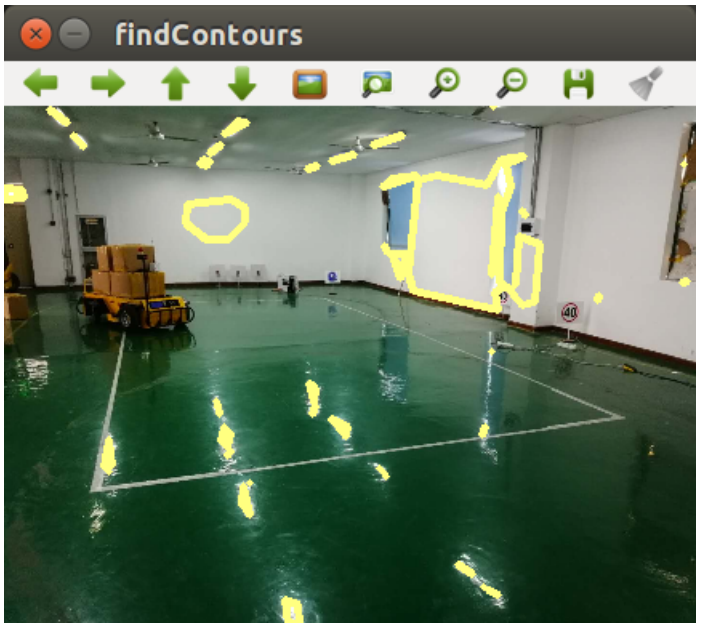
Surgical Removal

http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_core/py_basic_ops/py_basic_ops.html

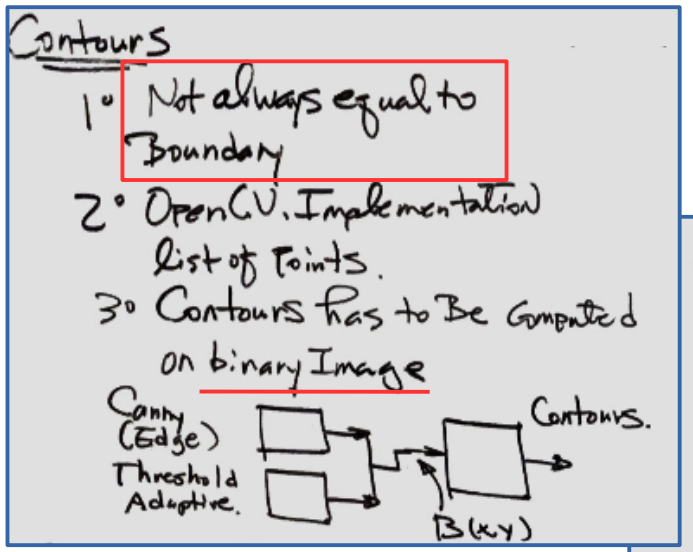
```
>>> px = img[100,100]
>>> print px
[157 166 200]

# accessing only blue pixel
>>> blue = img[100,100,0]
>>> print blue
157
```



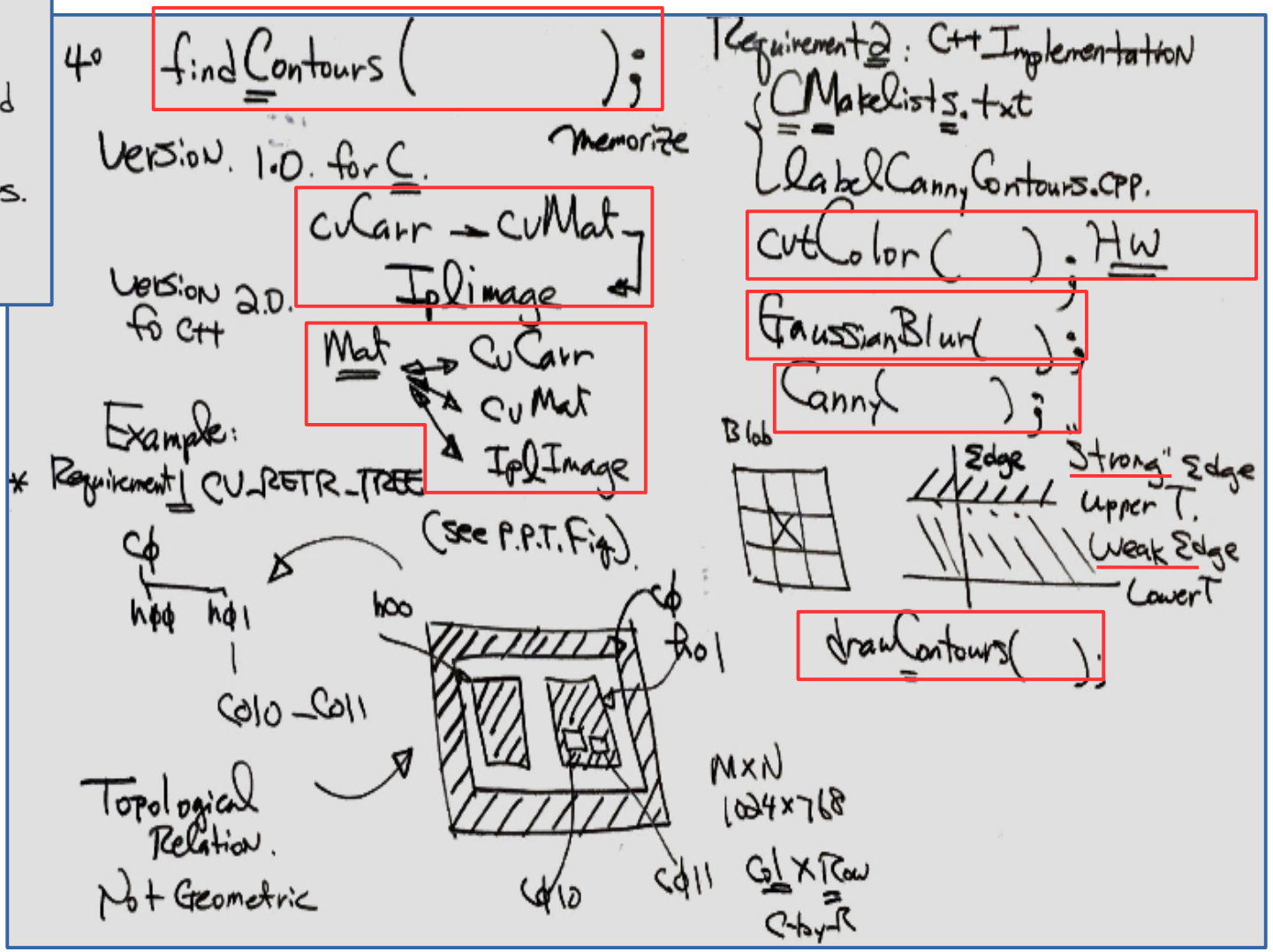


Contours Trees

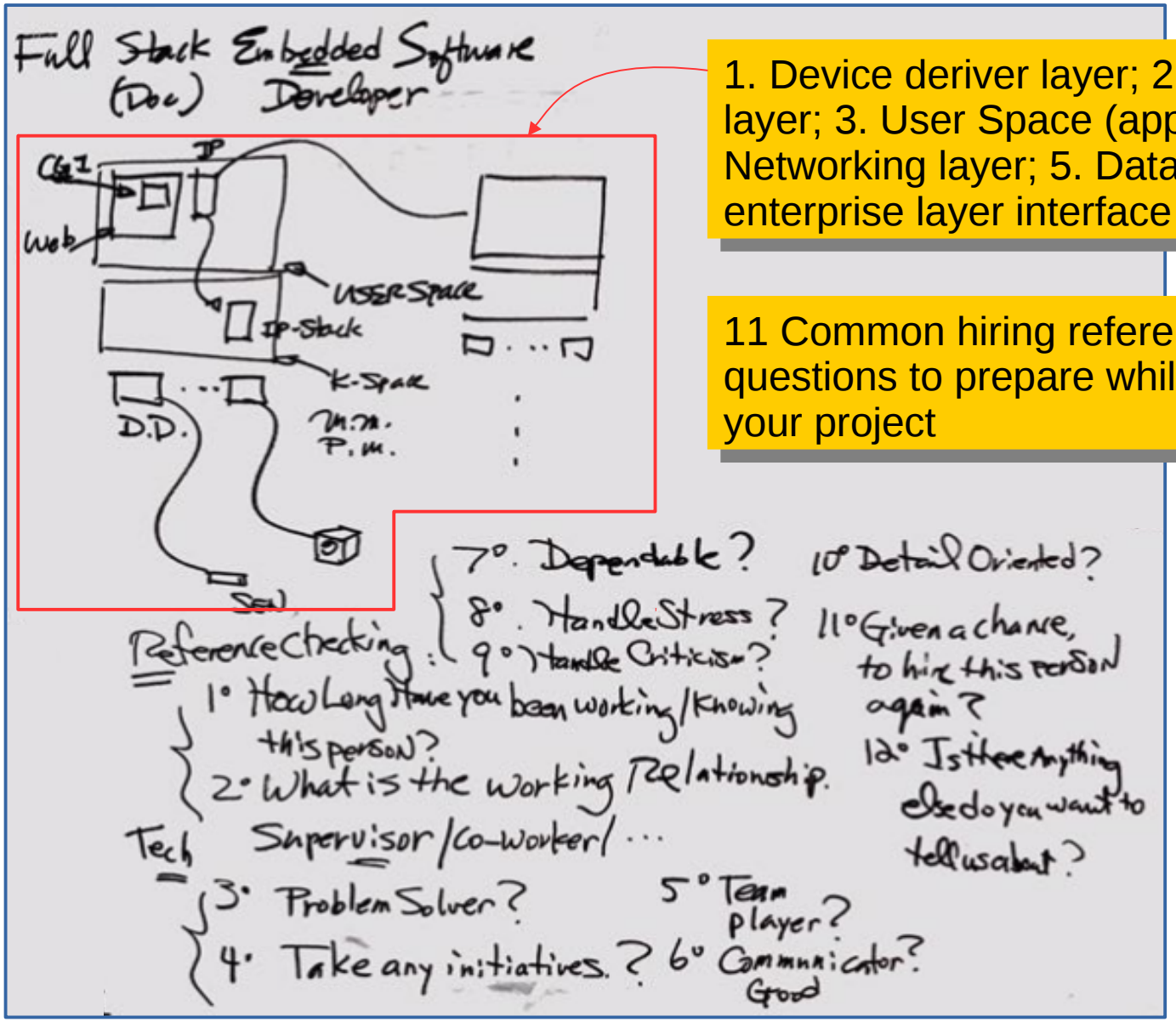


Topological relationship of each contour and mapping from a given contour image to tree structure

findContour( ); only works on binary image, one of the 3 images, Canny, Threshold and adaptiveThreshold



Full Stack Embedded Software Developer



1. Device deriver layer; 2. Kernel Space layer; 3. User Space (app) layer; 4. Networking layer; 5. Database and enterprise layer interface

11 Common hiring reference checking questions to prepare while working on your project