Moments

https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=moments

```
for( size_t i = 0; i < contours.size(); i++ )
 {
  mu[i] = moments( contours[i], false );
  }
```

1

$\mathtt{mu}_{ji}$ are computed as:

$$\mathtt{mu}_{ji} = \sum_{x,y} \left(\mathtt{array}(x, y) \cdot (x - \bar{x})^j \cdot (y - \bar{y})^i \right)$$

```
for( size_t i = 0; i < contours.size(); i++ )
 {
 mc[i] = Point2f( static_cast<float>(mu[i].m10/mu[i].m00) ,
                  static_cast<float>(mu[i].m01/mu[i].m00) );
 }
```

Mass center

$$\bar{x} = \frac{\mathtt{m}_{10}}{\mathtt{m}_{00}}, \; \bar{y} = \frac{\mathtt{m}_{01}}{\mathtt{m}_{00}}$$

spatial moments `Moments::m`$_{ji}$ are computed as:

2

$$\mathtt{m}_{ji} = \sum_{x,y} \left(\mathtt{array}(x, y) \cdot x^j \cdot y^i \right)$$

3

$$\mathtt{nu}_{ji} = \frac{\mathtt{mu}_{ji}}{\mathtt{m}_{00}^{(i+j)/2+1}}.$$

Normalized central moment

Hu Moments

https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=moments

C++: void HuMoments(const Moments& m, OutputArray hu)

C++: void HuMoments(const Moments& moments, double hu[7])

where η_{ji} stands for `Moments::nu`$_{ji}$

$$hu[0] = \eta_{20} + \eta_{02}$$
$$hu[1] = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$
$$hu[2] = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$
$$hu[3] = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$
$$hu[4] = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$
$$hu[5] = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$
$$hu[6] = (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

Match Shapes

https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=moments

```
C++:
double matchShapes(InputArray contour1,
                   InputArray contour2,
                   int method,
                   double parameter)
```

method=CV_CONTOURS_MATCH_I1

$$I_1(A,B) = \sum_{i=1...7} \left| \frac{1}{m_i^A} - \frac{1}{m_i^B} \right|$$

method=CV_CONTOURS_MATCH_I2

$$I_2(A,B) = \sum_{i=1...7} \left| m_i^A - m_i^B \right|$$

method=CV_CONTOURS_MATCH_I3

$$I_3(A,B) = \max_{i=1...7} \frac{\left| m_i^A - m_i^B \right|}{\left| m_i^A \right|}$$

where

$$m_i^A = \mathrm{sign}(h_i^A) \cdot \log h_i^A$$
$$m_i^B = \mathrm{sign}(h_i^B) \cdot \log h_i^B$$

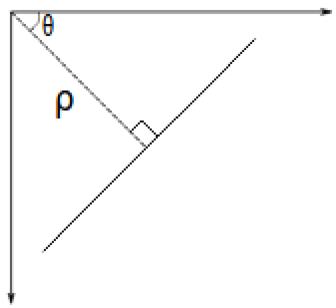
and h_i^A, h_i^B are the Hu moments of A and B, respectively.

Jul 11 Path Detection With Hough Angle

https://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html#houghlines

C++:

```
void HoughLines(InputArray image,
                OutputArray lines,
                double rho,
                double theta,
                int threshold,
                double srn=0,
                double stn=0 )
```



srn – For the multi-scale Hough transform, it is a divisor for the distance resolution `rho` . The coarse accumulator distance resolution is `rho` and the accurate accumulator resolution is `rho/srn` . If both `srn=0` and `stn=0` , the classical Hough transform is used. Otherwise, both these parameters should be positive.

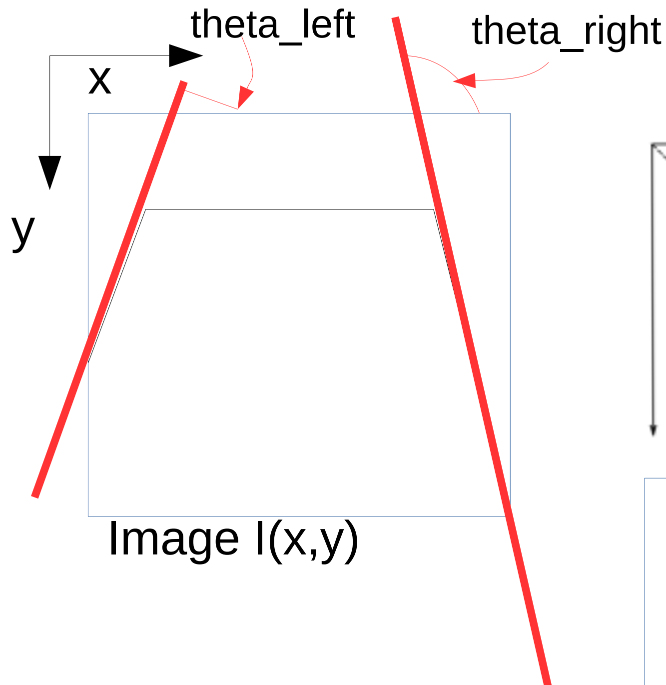
stn – For the multi-scale Hough transform, it is a divisor for the distance resolution `theta`.

CV_HOUGH_STANDARD standard. Every line is represented by two floating-point numbers (\rho, \theta) , where \rho is a distance between (0,0) point and the line, and \theta is the angle between x-axis and the normal to the line. Thus, the matrix must be of CV_32FC2 type

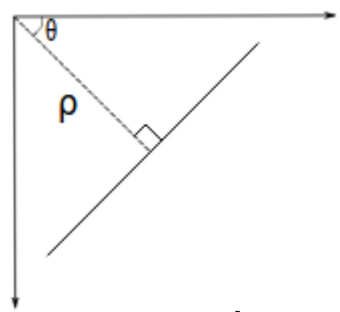
CV_HOUGH_PROBABILISTIC more efficient in case if the picture contains a few long linear segments. It returns line segments rather than the whole line. Each segment is represented by starting and ending points, and the matrix must be CV_32SC4 type.

CV_HOUGH_MULTI_SCALE multi-scale. The lines are encoded the same way as CV_HOUGH_STANDARD.

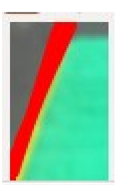
Jul 11 Theta Thresholding Hough Angle



Definition



θ is the angle formed by this perpendicular line and the horizontal axis measured in counter-clockwise (That direction varies on how you represent the coordinate system. This representation is used in OpenCV).

left   right

Example:
Typical path angles from the program
theta_left:0.418879 dgr:24.0122
theta_left:0.349066 dgr:20.0101
theta_right2.37365 dgr:136.069
theta_left:0.383972 dgr:22.0112

Example: sample code

```
vector<Vec2f> lines_l, lines_r; // will hold the results of the detection
HoughLines(roi_blur_gray_binary_l_canny,
        lines_l, 1, CV_PI/180, min_intersection_hough, 0, 0 );

float rho = lines_l[i][0], theta = lines_l[i][1]; // for thresholding

if ((theta >= angle_l_thre_low)
   &&  (theta <= angle_l_thre_up))
{
 cout << "theta_left:" << theta << " dgr:" <<(theta * 180.0/3.14) << endl;
```