JavaScript is disabled on your browser.

* [Overview](http://docs.google.com/overview-summary.html)
* [Package](http://docs.google.com/package-summary.html)
* Class
* [Tree](http://docs.google.com/package-tree.html)
* [Index](http://docs.google.com/index-all.html)
* [Help](http://docs.google.com/help-doc.html)
* [Prev Class](http://docs.google.com/org/opencv/imgproc/Imgproc.html)
* [Next Class](http://docs.google.com/org/opencv/imgproc/Moments.html)
* [Frames](http://docs.google.com/index.html?org/opencv/imgproc/LineSegmentDetector.html)
* [No Frames](http://docs.google.com/LineSegmentDetector.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* Field |
* Constr |
* [Method](#3znysh7)
* Detail:
* Field |
* Constr |
* [Method](#3dy6vkm)

org.opencv.imgproc

## Class LineSegmentDetector

* java.lang.Object
  + [org.opencv.core.Algorithm](http://docs.google.com/org/opencv/core/Algorithm.html)
    - org.opencv.imgproc.LineSegmentDetector
* public class LineSegmentDetector  
  extends [Algorithm](http://docs.google.com/org/opencv/core/Algorithm.html)  
  Line segment detector class following the algorithm described at CITE: Rafael12 . **Note:** Implementation has been removed due original code license conflict

### Method SummaryMethods

| Modifier and Type | Method and Description |
| --- | --- |
| static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) | [**\_\_fromPtr\_\_**](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html#__fromPtr__(long))(long addr) |
| int | [**compareSegments**](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html#compareSegments(org.opencv.core.Size,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Size](http://docs.google.com/org/opencv/core/Size.html) size, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines2) Draws two groups of lines in blue and red, counting the non overlapping (mismatching) pixels. |
| int | [**compareSegments**](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html#compareSegments(org.opencv.core.Size,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Size](http://docs.google.com/org/opencv/core/Size.html) size, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_image) Draws two groups of lines in blue and red, counting the non overlapping (mismatching) pixels. |
| void | [**detect**](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html#detect(org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) \_image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_lines) Finds lines in the input image. |
| void | [**detect**](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html#detect(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) \_image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_lines, [Mat](http://docs.google.com/org/opencv/core/Mat.html) width) Finds lines in the input image. |
| void | [**detect**](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html#detect(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) \_image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_lines, [Mat](http://docs.google.com/org/opencv/core/Mat.html) width, [Mat](http://docs.google.com/org/opencv/core/Mat.html) prec) Finds lines in the input image. |
| void | [**detect**](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html#detect(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) \_image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_lines, [Mat](http://docs.google.com/org/opencv/core/Mat.html) width, [Mat](http://docs.google.com/org/opencv/core/Mat.html) prec, [Mat](http://docs.google.com/org/opencv/core/Mat.html) nfa) Finds lines in the input image. |
| void | [**drawSegments**](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html#drawSegments(org.opencv.core.Mat,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) \_image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines) Draws the line segments on a given image. |

### Methods inherited from class org.opencv.core.[**Algorithm**](http://docs.google.com/org/opencv/core/Algorithm.html)[clear](http://docs.google.com/org/opencv/core/Algorithm.html#clear()), [empty](http://docs.google.com/org/opencv/core/Algorithm.html#empty()), [getDefaultName](http://docs.google.com/org/opencv/core/Algorithm.html#getDefaultName()), [getNativeObjAddr](http://docs.google.com/org/opencv/core/Algorithm.html#getNativeObjAddr()), [save](http://docs.google.com/org/opencv/core/Algorithm.html#save(java.lang.String))

### Methods inherited from class java.lang.Objectequals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Method Detail

#### \_\_fromPtr\_\_ public static [LineSegmentDetector](http://docs.google.com/org/opencv/imgproc/LineSegmentDetector.html) \_\_fromPtr\_\_(long addr)

#### compareSegments public int compareSegments([Size](http://docs.google.com/org/opencv/core/Size.html) size, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines2) Draws two groups of lines in blue and red, counting the non overlapping (mismatching) pixels.Parameters:size - The size of the image, where lines1 and lines2 were found.lines1 - The first group of lines that needs to be drawn. It is visualized in blue color.lines2 - The second group of lines. They visualized in red color. in order for lines1 and lines2 to be drawn in the above mentioned colors. Returns:automatically generated

#### compareSegments public int compareSegments([Size](http://docs.google.com/org/opencv/core/Size.html) size, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines1, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines2, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_image) Draws two groups of lines in blue and red, counting the non overlapping (mismatching) pixels.Parameters:size - The size of the image, where lines1 and lines2 were found.lines1 - The first group of lines that needs to be drawn. It is visualized in blue color.lines2 - The second group of lines. They visualized in red color.\_image - Optional image, where the lines will be drawn. The image should be color(3-channel) in order for lines1 and lines2 to be drawn in the above mentioned colors. Returns:automatically generated

#### detect public void detect([Mat](http://docs.google.com/org/opencv/core/Mat.html) \_image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_lines) Finds lines in the input image. This is the output of the default parameters of the algorithm on the above shown image. ![image](pics/building\_lsd.png)Parameters:\_image - A grayscale (CV\_8UC1) input image. If only a roi needs to be selected, use: lsd\_ptr->detect(image(roi), lines, ...); lines += Scalar(roi.x, roi.y, roi.x, roi.y);\_lines - A vector of Vec4i or Vec4f elements specifying the beginning and ending point of a line. Where Vec4i/Vec4f is (x1, y1, x2, y2), point 1 is the start, point 2 - end. Returned lines are strictly oriented depending on the gradient. bigger the value, logarithmically better the detection.

* + - * -1 corresponds to 10 mean false alarms
      * 0 corresponds to 1 mean false alarm
      * 1 corresponds to 0.1 mean false alarms This vector will be calculated only when the objects type is #LSD\_REFINE\_ADV.

#### detect public void detect([Mat](http://docs.google.com/org/opencv/core/Mat.html) \_image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_lines, [Mat](http://docs.google.com/org/opencv/core/Mat.html) width) Finds lines in the input image. This is the output of the default parameters of the algorithm on the above shown image. ![image](pics/building\_lsd.png)Parameters:\_image - A grayscale (CV\_8UC1) input image. If only a roi needs to be selected, use: lsd\_ptr->detect(image(roi), lines, ...); lines += Scalar(roi.x, roi.y, roi.x, roi.y);\_lines - A vector of Vec4i or Vec4f elements specifying the beginning and ending point of a line. Where Vec4i/Vec4f is (x1, y1, x2, y2), point 1 is the start, point 2 - end. Returned lines are strictly oriented depending on the gradient.width - Vector of widths of the regions, where the lines are found. E.g. Width of line. bigger the value, logarithmically better the detection.

* + - * -1 corresponds to 10 mean false alarms
      * 0 corresponds to 1 mean false alarm
      * 1 corresponds to 0.1 mean false alarms This vector will be calculated only when the objects type is #LSD\_REFINE\_ADV.

#### detect public void detect([Mat](http://docs.google.com/org/opencv/core/Mat.html) \_image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_lines, [Mat](http://docs.google.com/org/opencv/core/Mat.html) width, [Mat](http://docs.google.com/org/opencv/core/Mat.html) prec) Finds lines in the input image. This is the output of the default parameters of the algorithm on the above shown image. ![image](pics/building\_lsd.png)Parameters:\_image - A grayscale (CV\_8UC1) input image. If only a roi needs to be selected, use: lsd\_ptr->detect(image(roi), lines, ...); lines += Scalar(roi.x, roi.y, roi.x, roi.y);\_lines - A vector of Vec4i or Vec4f elements specifying the beginning and ending point of a line. Where Vec4i/Vec4f is (x1, y1, x2, y2), point 1 is the start, point 2 - end. Returned lines are strictly oriented depending on the gradient.width - Vector of widths of the regions, where the lines are found. E.g. Width of line.prec - Vector of precisions with which the lines are found. bigger the value, logarithmically better the detection.

* + - * -1 corresponds to 10 mean false alarms
      * 0 corresponds to 1 mean false alarm
      * 1 corresponds to 0.1 mean false alarms This vector will be calculated only when the objects type is #LSD\_REFINE\_ADV.

#### detect public void detect([Mat](http://docs.google.com/org/opencv/core/Mat.html) \_image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) \_lines, [Mat](http://docs.google.com/org/opencv/core/Mat.html) width, [Mat](http://docs.google.com/org/opencv/core/Mat.html) prec, [Mat](http://docs.google.com/org/opencv/core/Mat.html) nfa) Finds lines in the input image. This is the output of the default parameters of the algorithm on the above shown image. ![image](pics/building\_lsd.png)Parameters:\_image - A grayscale (CV\_8UC1) input image. If only a roi needs to be selected, use: lsd\_ptr->detect(image(roi), lines, ...); lines += Scalar(roi.x, roi.y, roi.x, roi.y);\_lines - A vector of Vec4i or Vec4f elements specifying the beginning and ending point of a line. Where Vec4i/Vec4f is (x1, y1, x2, y2), point 1 is the start, point 2 - end. Returned lines are strictly oriented depending on the gradient.width - Vector of widths of the regions, where the lines are found. E.g. Width of line.prec - Vector of precisions with which the lines are found.nfa - Vector containing number of false alarms in the line region, with precision of 10%. The bigger the value, logarithmically better the detection.

* + - * -1 corresponds to 10 mean false alarms
      * 0 corresponds to 1 mean false alarm
      * 1 corresponds to 0.1 mean false alarms This vector will be calculated only when the objects type is #LSD\_REFINE\_ADV.

#### drawSegments public void drawSegments([Mat](http://docs.google.com/org/opencv/core/Mat.html) \_image, [Mat](http://docs.google.com/org/opencv/core/Mat.html) lines) Draws the line segments on a given image.Parameters:\_image - The image, where the lines will be drawn. Should be bigger or equal to the image, where the lines were found.lines - A vector of the lines that needed to be drawn.

* [Overview](http://docs.google.com/overview-summary.html)
* [Package](http://docs.google.com/package-summary.html)
* Class
* [Tree](http://docs.google.com/package-tree.html)
* [Index](http://docs.google.com/index-all.html)
* [Help](http://docs.google.com/help-doc.html)
* [Prev Class](http://docs.google.com/org/opencv/imgproc/Imgproc.html)
* [Next Class](http://docs.google.com/org/opencv/imgproc/Moments.html)
* [Frames](http://docs.google.com/index.html?org/opencv/imgproc/LineSegmentDetector.html)
* [No Frames](http://docs.google.com/LineSegmentDetector.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* Field |
* Constr |
* [Method](#3znysh7)
* Detail:
* Field |
* Constr |
* [Method](#3dy6vkm)

Generated on 2021-04-02 03:15:03 / OpenCV 3.4.14