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/features2d/MSER.html)
* [Next Class](http://docs.google.com/org/opencv/features2d/Params.html)
* [Frames](http://docs.google.com/index.html?org/opencv/features2d/ORB.html)
* [No Frames](http://docs.google.com/ORB.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* [Field](#3znysh7) |
* Constr |
* [Method](#2et92p0)
* Detail:
* [Field](#4d34og8) |
* Constr |
* [Method](#26in1rg)

org.opencv.features2d

## Class ORB

* java.lang.Object
  + [org.opencv.core.Algorithm](http://docs.google.com/org/opencv/core/Algorithm.html)
    - [org.opencv.features2d.Feature2D](http://docs.google.com/org/opencv/features2d/Feature2D.html)
      * org.opencv.features2d.ORB
* public class ORB  
  extends [Feature2D](http://docs.google.com/org/opencv/features2d/Feature2D.html)  
  Class implementing the ORB (\*oriented BRIEF\*) keypoint detector and descriptor extractor described in CITE: RRKB11 . The algorithm uses FAST in pyramids to detect stable keypoints, selects the strongest features using FAST or Harris response, finds their orientation using first-order moments and computes the descriptors using BRIEF (where the coordinates of random point pairs (or k-tuples) are rotated according to the measured orientation).

### Field SummaryFields

| Modifier and Type | Field and Description |
| --- | --- |
| static int | [**FAST\_SCORE**](http://docs.google.com/org/opencv/features2d/ORB.html#FAST_SCORE) |
| static int | [**HARRIS\_SCORE**](http://docs.google.com/org/opencv/features2d/ORB.html#HARRIS_SCORE) |
| static int | [**kBytes**](http://docs.google.com/org/opencv/features2d/ORB.html#kBytes) |

### Method SummaryMethods

| Modifier and Type | Method and Description |
| --- | --- |
| static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) | [**\_\_fromPtr\_\_**](http://docs.google.com/org/opencv/features2d/ORB.html#__fromPtr__(long))(long addr) |
| static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) | [**create**](http://docs.google.com/org/opencv/features2d/ORB.html#create())() The ORB constructor pyramid, where each next level has 4x less pixels than the previous, but such a big scale factor will degrade feature matching scores dramatically. |
| static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) | [**create**](http://docs.google.com/org/opencv/features2d/ORB.html#create(int))(int nfeatures) The ORB constructor |
| static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) | [**create**](http://docs.google.com/org/opencv/features2d/ORB.html#create(int,%20float))(int nfeatures, float scaleFactor) The ORB constructor |
| static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) | [**create**](http://docs.google.com/org/opencv/features2d/ORB.html#create(int,%20float,%20int))(int nfeatures, float scaleFactor, int nlevels) The ORB constructor |
| static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) | [**create**](http://docs.google.com/org/opencv/features2d/ORB.html#create(int,%20float,%20int,%20int))(int nfeatures, float scaleFactor, int nlevels, int edgeThreshold) The ORB constructor |
| static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) | [**create**](http://docs.google.com/org/opencv/features2d/ORB.html#create(int,%20float,%20int,%20int,%20int))(int nfeatures, float scaleFactor, int nlevels, int edgeThreshold, int firstLevel) The ORB constructor |
| static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) | [**create**](http://docs.google.com/org/opencv/features2d/ORB.html#create(int,%20float,%20int,%20int,%20int,%20int))(int nfeatures, float scaleFactor, int nlevels, int edgeThreshold, int firstLevel, int WTA\_K) The ORB constructor |
| static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) | [**create**](http://docs.google.com/org/opencv/features2d/ORB.html#create(int,%20float,%20int,%20int,%20int,%20int,%20int))(int nfeatures, float scaleFactor, int nlevels, int edgeThreshold, int firstLevel, int WTA\_K, int scoreType) The ORB constructor |
| static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) | [**create**](http://docs.google.com/org/opencv/features2d/ORB.html#create(int,%20float,%20int,%20int,%20int,%20int,%20int,%20int))(int nfeatures, float scaleFactor, int nlevels, int edgeThreshold, int firstLevel, int WTA\_K, int scoreType, int patchSize) The ORB constructor |
| static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) | [**create**](http://docs.google.com/org/opencv/features2d/ORB.html#create(int,%20float,%20int,%20int,%20int,%20int,%20int,%20int,%20int))(int nfeatures, float scaleFactor, int nlevels, int edgeThreshold, int firstLevel, int WTA\_K, int scoreType, int patchSize, int fastThreshold) The ORB constructor |
| java.lang.String | [**getDefaultName**](http://docs.google.com/org/opencv/features2d/ORB.html#getDefaultName())() Returns the algorithm string identifier. |
| int | [**getEdgeThreshold**](http://docs.google.com/org/opencv/features2d/ORB.html#getEdgeThreshold())() |
| int | [**getFastThreshold**](http://docs.google.com/org/opencv/features2d/ORB.html#getFastThreshold())() |
| int | [**getFirstLevel**](http://docs.google.com/org/opencv/features2d/ORB.html#getFirstLevel())() |
| int | [**getMaxFeatures**](http://docs.google.com/org/opencv/features2d/ORB.html#getMaxFeatures())() |
| int | [**getNLevels**](http://docs.google.com/org/opencv/features2d/ORB.html#getNLevels())() |
| int | [**getPatchSize**](http://docs.google.com/org/opencv/features2d/ORB.html#getPatchSize())() |
| double | [**getScaleFactor**](http://docs.google.com/org/opencv/features2d/ORB.html#getScaleFactor())() |
| int | [**getScoreType**](http://docs.google.com/org/opencv/features2d/ORB.html#getScoreType())() |
| int | [**getWTA\_K**](http://docs.google.com/org/opencv/features2d/ORB.html#getWTA_K())() |
| void | [**setEdgeThreshold**](http://docs.google.com/org/opencv/features2d/ORB.html#setEdgeThreshold(int))(int edgeThreshold) |
| void | [**setFastThreshold**](http://docs.google.com/org/opencv/features2d/ORB.html#setFastThreshold(int))(int fastThreshold) |
| void | [**setFirstLevel**](http://docs.google.com/org/opencv/features2d/ORB.html#setFirstLevel(int))(int firstLevel) |
| void | [**setMaxFeatures**](http://docs.google.com/org/opencv/features2d/ORB.html#setMaxFeatures(int))(int maxFeatures) |
| void | [**setNLevels**](http://docs.google.com/org/opencv/features2d/ORB.html#setNLevels(int))(int nlevels) |
| void | [**setPatchSize**](http://docs.google.com/org/opencv/features2d/ORB.html#setPatchSize(int))(int patchSize) |
| void | [**setScaleFactor**](http://docs.google.com/org/opencv/features2d/ORB.html#setScaleFactor(double))(double scaleFactor) |
| void | [**setScoreType**](http://docs.google.com/org/opencv/features2d/ORB.html#setScoreType(int))(int scoreType) |
| void | [**setWTA\_K**](http://docs.google.com/org/opencv/features2d/ORB.html#setWTA_K(int))(int wta\_k) |

### Methods inherited from class org.opencv.features2d.[**Feature2D**](http://docs.google.com/org/opencv/features2d/Feature2D.html)[compute](http://docs.google.com/org/opencv/features2d/Feature2D.html#compute(java.util.List,%20java.util.List,%20java.util.List)), [compute](http://docs.google.com/org/opencv/features2d/Feature2D.html#compute(org.opencv.core.Mat,%20org.opencv.core.MatOfKeyPoint,%20org.opencv.core.Mat)), [defaultNorm](http://docs.google.com/org/opencv/features2d/Feature2D.html#defaultNorm()), [descriptorSize](http://docs.google.com/org/opencv/features2d/Feature2D.html#descriptorSize()), [descriptorType](http://docs.google.com/org/opencv/features2d/Feature2D.html#descriptorType()), [detect](http://docs.google.com/org/opencv/features2d/Feature2D.html#detect(java.util.List,%20java.util.List)), [detect](http://docs.google.com/org/opencv/features2d/Feature2D.html#detect(java.util.List,%20java.util.List,%20java.util.List)), [detect](http://docs.google.com/org/opencv/features2d/Feature2D.html#detect(org.opencv.core.Mat,%20org.opencv.core.MatOfKeyPoint)), [detect](http://docs.google.com/org/opencv/features2d/Feature2D.html#detect(org.opencv.core.Mat,%20org.opencv.core.MatOfKeyPoint,%20org.opencv.core.Mat)), [detectAndCompute](http://docs.google.com/org/opencv/features2d/Feature2D.html#detectAndCompute(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.MatOfKeyPoint,%20org.opencv.core.Mat)), [detectAndCompute](http://docs.google.com/org/opencv/features2d/Feature2D.html#detectAndCompute(org.opencv.core.Mat,%20org.opencv.core.Mat,%20org.opencv.core.MatOfKeyPoint,%20org.opencv.core.Mat,%20boolean)), [empty](http://docs.google.com/org/opencv/features2d/Feature2D.html#empty()), [read](http://docs.google.com/org/opencv/features2d/Feature2D.html#read(java.lang.String)), [write](http://docs.google.com/org/opencv/features2d/Feature2D.html#write(java.lang.String))

### 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()), [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

### Field Detail

#### FAST\_SCORE public static final int FAST\_SCORESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.features2d.ORB.FAST_SCORE)

#### HARRIS\_SCORE public static final int HARRIS\_SCORESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.features2d.ORB.HARRIS_SCORE)

#### kBytes public static final int kBytesSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.features2d.ORB.kBytes)

### Method Detail

#### \_\_fromPtr\_\_ public static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) \_\_fromPtr\_\_(long addr)

#### create public static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) create() The ORB constructor pyramid, where each next level has 4x less pixels than the previous, but such a big scale factor will degrade feature matching scores dramatically. On the other hand, too close to 1 scale factor will mean that to cover certain scale range you will need more pyramid levels and so the speed will suffer. input\_image\_linear\_size/pow(scaleFactor, nlevels - firstLevel). roughly match the patchSize parameter. with upscaled source image. default value 2 means the BRIEF where we take a random point pair and compare their brightnesses, so we get 0/1 response. Other possible values are 3 and 4. For example, 3 means that we take 3 random points (of course, those point coordinates are random, but they are generated from the pre-defined seed, so each element of BRIEF descriptor is computed deterministically from the pixel rectangle), find point of maximum brightness and output index of the winner (0, 1 or 2). Such output will occupy 2 bits, and therefore it will need a special variant of Hamming distance, denoted as NORM\_HAMMING2 (2 bits per bin). When WTA\_K=4, we take 4 random points to compute each bin (that will also occupy 2 bits with possible values 0, 1, 2 or 3). (the score is written to KeyPoint::score and is used to retain best nfeatures features); FAST\_SCORE is alternative value of the parameter that produces slightly less stable keypoints, but it is a little faster to compute. pyramid layers the perceived image area covered by a feature will be larger.Returns:automatically generated

#### create public static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) create(int nfeatures) The ORB constructorParameters:nfeatures - The maximum number of features to retain. pyramid, where each next level has 4x less pixels than the previous, but such a big scale factor will degrade feature matching scores dramatically. On the other hand, too close to 1 scale factor will mean that to cover certain scale range you will need more pyramid levels and so the speed will suffer. input\_image\_linear\_size/pow(scaleFactor, nlevels - firstLevel). roughly match the patchSize parameter. with upscaled source image. default value 2 means the BRIEF where we take a random point pair and compare their brightnesses, so we get 0/1 response. Other possible values are 3 and 4. For example, 3 means that we take 3 random points (of course, those point coordinates are random, but they are generated from the pre-defined seed, so each element of BRIEF descriptor is computed deterministically from the pixel rectangle), find point of maximum brightness and output index of the winner (0, 1 or 2). Such output will occupy 2 bits, and therefore it will need a special variant of Hamming distance, denoted as NORM\_HAMMING2 (2 bits per bin). When WTA\_K=4, we take 4 random points to compute each bin (that will also occupy 2 bits with possible values 0, 1, 2 or 3). (the score is written to KeyPoint::score and is used to retain best nfeatures features); FAST\_SCORE is alternative value of the parameter that produces slightly less stable keypoints, but it is a little faster to compute. pyramid layers the perceived image area covered by a feature will be larger. Returns:automatically generated

#### create public static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) create(int nfeatures, float scaleFactor) The ORB constructorParameters:nfeatures - The maximum number of features to retain.scaleFactor - Pyramid decimation ratio, greater than 1. scaleFactor==2 means the classical pyramid, where each next level has 4x less pixels than the previous, but such a big scale factor will degrade feature matching scores dramatically. On the other hand, too close to 1 scale factor will mean that to cover certain scale range you will need more pyramid levels and so the speed will suffer. input\_image\_linear\_size/pow(scaleFactor, nlevels - firstLevel). roughly match the patchSize parameter. with upscaled source image. default value 2 means the BRIEF where we take a random point pair and compare their brightnesses, so we get 0/1 response. Other possible values are 3 and 4. For example, 3 means that we take 3 random points (of course, those point coordinates are random, but they are generated from the pre-defined seed, so each element of BRIEF descriptor is computed deterministically from the pixel rectangle), find point of maximum brightness and output index of the winner (0, 1 or 2). Such output will occupy 2 bits, and therefore it will need a special variant of Hamming distance, denoted as NORM\_HAMMING2 (2 bits per bin). When WTA\_K=4, we take 4 random points to compute each bin (that will also occupy 2 bits with possible values 0, 1, 2 or 3). (the score is written to KeyPoint::score and is used to retain best nfeatures features); FAST\_SCORE is alternative value of the parameter that produces slightly less stable keypoints, but it is a little faster to compute. pyramid layers the perceived image area covered by a feature will be larger. Returns:automatically generated

#### create public static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) create(int nfeatures, float scaleFactor, int nlevels) The ORB constructorParameters:nfeatures - The maximum number of features to retain.scaleFactor - Pyramid decimation ratio, greater than 1. scaleFactor==2 means the classical pyramid, where each next level has 4x less pixels than the previous, but such a big scale factor will degrade feature matching scores dramatically. On the other hand, too close to 1 scale factor will mean that to cover certain scale range you will need more pyramid levels and so the speed will suffer.nlevels - The number of pyramid levels. The smallest level will have linear size equal to input\_image\_linear\_size/pow(scaleFactor, nlevels - firstLevel). roughly match the patchSize parameter. with upscaled source image. default value 2 means the BRIEF where we take a random point pair and compare their brightnesses, so we get 0/1 response. Other possible values are 3 and 4. For example, 3 means that we take 3 random points (of course, those point coordinates are random, but they are generated from the pre-defined seed, so each element of BRIEF descriptor is computed deterministically from the pixel rectangle), find point of maximum brightness and output index of the winner (0, 1 or 2). Such output will occupy 2 bits, and therefore it will need a special variant of Hamming distance, denoted as NORM\_HAMMING2 (2 bits per bin). When WTA\_K=4, we take 4 random points to compute each bin (that will also occupy 2 bits with possible values 0, 1, 2 or 3). (the score is written to KeyPoint::score and is used to retain best nfeatures features); FAST\_SCORE is alternative value of the parameter that produces slightly less stable keypoints, but it is a little faster to compute. pyramid layers the perceived image area covered by a feature will be larger. Returns:automatically generated

#### create public static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) create(int nfeatures, float scaleFactor, int nlevels, int edgeThreshold) The ORB constructorParameters:nfeatures - The maximum number of features to retain.scaleFactor - Pyramid decimation ratio, greater than 1. scaleFactor==2 means the classical pyramid, where each next level has 4x less pixels than the previous, but such a big scale factor will degrade feature matching scores dramatically. On the other hand, too close to 1 scale factor will mean that to cover certain scale range you will need more pyramid levels and so the speed will suffer.nlevels - The number of pyramid levels. The smallest level will have linear size equal to input\_image\_linear\_size/pow(scaleFactor, nlevels - firstLevel).edgeThreshold - This is size of the border where the features are not detected. It should roughly match the patchSize parameter. with upscaled source image. default value 2 means the BRIEF where we take a random point pair and compare their brightnesses, so we get 0/1 response. Other possible values are 3 and 4. For example, 3 means that we take 3 random points (of course, those point coordinates are random, but they are generated from the pre-defined seed, so each element of BRIEF descriptor is computed deterministically from the pixel rectangle), find point of maximum brightness and output index of the winner (0, 1 or 2). Such output will occupy 2 bits, and therefore it will need a special variant of Hamming distance, denoted as NORM\_HAMMING2 (2 bits per bin). When WTA\_K=4, we take 4 random points to compute each bin (that will also occupy 2 bits with possible values 0, 1, 2 or 3). (the score is written to KeyPoint::score and is used to retain best nfeatures features); FAST\_SCORE is alternative value of the parameter that produces slightly less stable keypoints, but it is a little faster to compute. pyramid layers the perceived image area covered by a feature will be larger. Returns:automatically generated

#### create public static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) create(int nfeatures, float scaleFactor, int nlevels, int edgeThreshold, int firstLevel) The ORB constructorParameters:nfeatures - The maximum number of features to retain.scaleFactor - Pyramid decimation ratio, greater than 1. scaleFactor==2 means the classical pyramid, where each next level has 4x less pixels than the previous, but such a big scale factor will degrade feature matching scores dramatically. On the other hand, too close to 1 scale factor will mean that to cover certain scale range you will need more pyramid levels and so the speed will suffer.nlevels - The number of pyramid levels. The smallest level will have linear size equal to input\_image\_linear\_size/pow(scaleFactor, nlevels - firstLevel).edgeThreshold - This is size of the border where the features are not detected. It should roughly match the patchSize parameter.firstLevel - The level of pyramid to put source image to. Previous layers are filled with upscaled source image. default value 2 means the BRIEF where we take a random point pair and compare their brightnesses, so we get 0/1 response. Other possible values are 3 and 4. For example, 3 means that we take 3 random points (of course, those point coordinates are random, but they are generated from the pre-defined seed, so each element of BRIEF descriptor is computed deterministically from the pixel rectangle), find point of maximum brightness and output index of the winner (0, 1 or 2). Such output will occupy 2 bits, and therefore it will need a special variant of Hamming distance, denoted as NORM\_HAMMING2 (2 bits per bin). When WTA\_K=4, we take 4 random points to compute each bin (that will also occupy 2 bits with possible values 0, 1, 2 or 3). (the score is written to KeyPoint::score and is used to retain best nfeatures features); FAST\_SCORE is alternative value of the parameter that produces slightly less stable keypoints, but it is a little faster to compute. pyramid layers the perceived image area covered by a feature will be larger. Returns:automatically generated

#### create public static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) create(int nfeatures, float scaleFactor, int nlevels, int edgeThreshold, int firstLevel, int WTA\_K) The ORB constructorParameters:nfeatures - The maximum number of features to retain.scaleFactor - Pyramid decimation ratio, greater than 1. scaleFactor==2 means the classical pyramid, where each next level has 4x less pixels than the previous, but such a big scale factor will degrade feature matching scores dramatically. On the other hand, too close to 1 scale factor will mean that to cover certain scale range you will need more pyramid levels and so the speed will suffer.nlevels - The number of pyramid levels. The smallest level will have linear size equal to input\_image\_linear\_size/pow(scaleFactor, nlevels - firstLevel).edgeThreshold - This is size of the border where the features are not detected. It should roughly match the patchSize parameter.firstLevel - The level of pyramid to put source image to. Previous layers are filled with upscaled source image.WTA\_K - The number of points that produce each element of the oriented BRIEF descriptor. The default value 2 means the BRIEF where we take a random point pair and compare their brightnesses, so we get 0/1 response. Other possible values are 3 and 4. For example, 3 means that we take 3 random points (of course, those point coordinates are random, but they are generated from the pre-defined seed, so each element of BRIEF descriptor is computed deterministically from the pixel rectangle), find point of maximum brightness and output index of the winner (0, 1 or 2). Such output will occupy 2 bits, and therefore it will need a special variant of Hamming distance, denoted as NORM\_HAMMING2 (2 bits per bin). When WTA\_K=4, we take 4 random points to compute each bin (that will also occupy 2 bits with possible values 0, 1, 2 or 3). (the score is written to KeyPoint::score and is used to retain best nfeatures features); FAST\_SCORE is alternative value of the parameter that produces slightly less stable keypoints, but it is a little faster to compute. pyramid layers the perceived image area covered by a feature will be larger. Returns:automatically generated

#### create public static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) create(int nfeatures, float scaleFactor, int nlevels, int edgeThreshold, int firstLevel, int WTA\_K, int scoreType) The ORB constructorParameters:nfeatures - The maximum number of features to retain.scaleFactor - Pyramid decimation ratio, greater than 1. scaleFactor==2 means the classical pyramid, where each next level has 4x less pixels than the previous, but such a big scale factor will degrade feature matching scores dramatically. On the other hand, too close to 1 scale factor will mean that to cover certain scale range you will need more pyramid levels and so the speed will suffer.nlevels - The number of pyramid levels. The smallest level will have linear size equal to input\_image\_linear\_size/pow(scaleFactor, nlevels - firstLevel).edgeThreshold - This is size of the border where the features are not detected. It should roughly match the patchSize parameter.firstLevel - The level of pyramid to put source image to. Previous layers are filled with upscaled source image.WTA\_K - The number of points that produce each element of the oriented BRIEF descriptor. The default value 2 means the BRIEF where we take a random point pair and compare their brightnesses, so we get 0/1 response. Other possible values are 3 and 4. For example, 3 means that we take 3 random points (of course, those point coordinates are random, but they are generated from the pre-defined seed, so each element of BRIEF descriptor is computed deterministically from the pixel rectangle), find point of maximum brightness and output index of the winner (0, 1 or 2). Such output will occupy 2 bits, and therefore it will need a special variant of Hamming distance, denoted as NORM\_HAMMING2 (2 bits per bin). When WTA\_K=4, we take 4 random points to compute each bin (that will also occupy 2 bits with possible values 0, 1, 2 or 3).scoreType - The default HARRIS\_SCORE means that Harris algorithm is used to rank features (the score is written to KeyPoint::score and is used to retain best nfeatures features); FAST\_SCORE is alternative value of the parameter that produces slightly less stable keypoints, but it is a little faster to compute. pyramid layers the perceived image area covered by a feature will be larger. Returns:automatically generated

#### create public static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) create(int nfeatures, float scaleFactor, int nlevels, int edgeThreshold, int firstLevel, int WTA\_K, int scoreType, int patchSize) The ORB constructorParameters:nfeatures - The maximum number of features to retain.scaleFactor - Pyramid decimation ratio, greater than 1. scaleFactor==2 means the classical pyramid, where each next level has 4x less pixels than the previous, but such a big scale factor will degrade feature matching scores dramatically. On the other hand, too close to 1 scale factor will mean that to cover certain scale range you will need more pyramid levels and so the speed will suffer.nlevels - The number of pyramid levels. The smallest level will have linear size equal to input\_image\_linear\_size/pow(scaleFactor, nlevels - firstLevel).edgeThreshold - This is size of the border where the features are not detected. It should roughly match the patchSize parameter.firstLevel - The level of pyramid to put source image to. Previous layers are filled with upscaled source image.WTA\_K - The number of points that produce each element of the oriented BRIEF descriptor. The default value 2 means the BRIEF where we take a random point pair and compare their brightnesses, so we get 0/1 response. Other possible values are 3 and 4. For example, 3 means that we take 3 random points (of course, those point coordinates are random, but they are generated from the pre-defined seed, so each element of BRIEF descriptor is computed deterministically from the pixel rectangle), find point of maximum brightness and output index of the winner (0, 1 or 2). Such output will occupy 2 bits, and therefore it will need a special variant of Hamming distance, denoted as NORM\_HAMMING2 (2 bits per bin). When WTA\_K=4, we take 4 random points to compute each bin (that will also occupy 2 bits with possible values 0, 1, 2 or 3).scoreType - The default HARRIS\_SCORE means that Harris algorithm is used to rank features (the score is written to KeyPoint::score and is used to retain best nfeatures features); FAST\_SCORE is alternative value of the parameter that produces slightly less stable keypoints, but it is a little faster to compute.patchSize - size of the patch used by the oriented BRIEF descriptor. Of course, on smaller pyramid layers the perceived image area covered by a feature will be larger. Returns:automatically generated

#### create public static [ORB](http://docs.google.com/org/opencv/features2d/ORB.html) create(int nfeatures, float scaleFactor, int nlevels, int edgeThreshold, int firstLevel, int WTA\_K, int scoreType, int patchSize, int fastThreshold) The ORB constructorParameters:nfeatures - The maximum number of features to retain.scaleFactor - Pyramid decimation ratio, greater than 1. scaleFactor==2 means the classical pyramid, where each next level has 4x less pixels than the previous, but such a big scale factor will degrade feature matching scores dramatically. On the other hand, too close to 1 scale factor will mean that to cover certain scale range you will need more pyramid levels and so the speed will suffer.nlevels - The number of pyramid levels. The smallest level will have linear size equal to input\_image\_linear\_size/pow(scaleFactor, nlevels - firstLevel).edgeThreshold - This is size of the border where the features are not detected. It should roughly match the patchSize parameter.firstLevel - The level of pyramid to put source image to. Previous layers are filled with upscaled source image.WTA\_K - The number of points that produce each element of the oriented BRIEF descriptor. The default value 2 means the BRIEF where we take a random point pair and compare their brightnesses, so we get 0/1 response. Other possible values are 3 and 4. For example, 3 means that we take 3 random points (of course, those point coordinates are random, but they are generated from the pre-defined seed, so each element of BRIEF descriptor is computed deterministically from the pixel rectangle), find point of maximum brightness and output index of the winner (0, 1 or 2). Such output will occupy 2 bits, and therefore it will need a special variant of Hamming distance, denoted as NORM\_HAMMING2 (2 bits per bin). When WTA\_K=4, we take 4 random points to compute each bin (that will also occupy 2 bits with possible values 0, 1, 2 or 3).scoreType - The default HARRIS\_SCORE means that Harris algorithm is used to rank features (the score is written to KeyPoint::score and is used to retain best nfeatures features); FAST\_SCORE is alternative value of the parameter that produces slightly less stable keypoints, but it is a little faster to compute.patchSize - size of the patch used by the oriented BRIEF descriptor. Of course, on smaller pyramid layers the perceived image area covered by a feature will be larger.fastThreshold - the fast threshold Returns:automatically generated

#### getDefaultName public java.lang.String getDefaultName() **Description copied from class:**[**Algorithm**](http://docs.google.com/org/opencv/core/Algorithm.html#getDefaultName()) Returns the algorithm string identifier. This string is used as top level xml/yml node tag when the object is saved to a file or string.**Overrides:** [getDefaultName](http://docs.google.com/org/opencv/features2d/Feature2D.html#getDefaultName()) in class [Feature2D](http://docs.google.com/org/opencv/features2d/Feature2D.html) Returns:automatically generated

#### getEdgeThreshold public int getEdgeThreshold()

#### getFastThreshold public int getFastThreshold()

#### getFirstLevel public int getFirstLevel()

#### getMaxFeatures public int getMaxFeatures()

#### getNLevels public int getNLevels()

#### getPatchSize public int getPatchSize()

#### getScaleFactor public double getScaleFactor()

#### getScoreType public int getScoreType()

#### getWTA\_K public int getWTA\_K()

#### setEdgeThreshold public void setEdgeThreshold(int edgeThreshold)

#### setFastThreshold public void setFastThreshold(int fastThreshold)

#### setFirstLevel public void setFirstLevel(int firstLevel)

#### setMaxFeatures public void setMaxFeatures(int maxFeatures)

#### setNLevels public void setNLevels(int nlevels)

#### setPatchSize public void setPatchSize(int patchSize)

#### setScaleFactor public void setScaleFactor(double scaleFactor)

#### setScoreType public void setScoreType(int scoreType)

#### setWTA\_K public void setWTA\_K(int wta\_k)

* [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/features2d/MSER.html)
* [Next Class](http://docs.google.com/org/opencv/features2d/Params.html)
* [Frames](http://docs.google.com/index.html?org/opencv/features2d/ORB.html)
* [No Frames](http://docs.google.com/ORB.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* [Field](#3znysh7) |
* Constr |
* [Method](#2et92p0)
* Detail:
* [Field](#4d34og8) |
* Constr |
* [Method](#26in1rg)

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