```
/** @brief The base class for stereo correspondence algorithms.
  */
class CV_EXPORTS_W StereoMatcher : public Algorithm
{
public:
    enum { DISP_SHIFT = 4,
           DISP_SCALE = (1 << DISP_SHIFT)
         };

    /** @brief Computes disparity map for the specified stereo pair

    @param left Left 8-bit single-channel image.
    @param right Right image of the same size and the same type as the left one.
    @param disparity Output disparity map. It has the same size as the input images. Some
algorithms,
    like StereoBM or StereoSGBM compute 16-bit fixed-point disparity map (where each disparity
value
    has 4 fractional bits), whereas other algorithms output 32-bit floating-point disparity map.
      */
    CV_WRAP virtual void compute( InputArray left, InputArray right,
                                  OutputArray disparity ) = 0;

    CV_WRAP virtual int getMinDisparity() const = 0;
    CV_WRAP virtual void setMinDisparity(int minDisparity) = 0;

    CV_WRAP virtual int getNumDisparities() const = 0;
    CV_WRAP virtual void setNumDisparities(int numDisparities) = 0;

    CV_WRAP virtual int getBlockSize() const = 0;
    CV_WRAP virtual void setBlockSize(int blockSize) = 0;

    CV_WRAP virtual int getSpeckleWindowSize() const = 0;
    CV_WRAP virtual void setSpeckleWindowSize(int speckleWindowSize) = 0;

    CV_WRAP virtual int getSpeckleRange() const = 0;
    CV_WRAP virtual void setSpeckleRange(int speckleRange) = 0;

    CV_WRAP virtual int getDisp12MaxDiff() const = 0;
    CV_WRAP virtual void setDisp12MaxDiff(int disp12MaxDiff) = 0;
};
```

```
/** @brief Class for computing stereo correspondence using the block matching algorithm,
introduced and
contributed to OpenCV by K. Konolige.
 */
class CV_EXPORTS_W StereoBM : public StereoMatcher
{
public:
    enum { PREFILTER_NORMALIZED_RESPONSE = 0,
           PREFILTER_XSOBEL              = 1
         };

    CV_WRAP virtual int getPreFilterType() const = 0;
    CV_WRAP virtual void setPreFilterType(int preFilterType) = 0;

    CV_WRAP virtual int getPreFilterSize() const = 0;
    CV_WRAP virtual void setPreFilterSize(int preFilterSize) = 0;

    CV_WRAP virtual int getPreFilterCap() const = 0;
    CV_WRAP virtual void setPreFilterCap(int preFilterCap) = 0;

    CV_WRAP virtual int getTextureThreshold() const = 0;
    CV_WRAP virtual void setTextureThreshold(int textureThreshold) = 0;

    CV_WRAP virtual int getUniquenessRatio() const = 0;
    CV_WRAP virtual void setUniquenessRatio(int uniquenessRatio) = 0;

    CV_WRAP virtual int getSmallerBlockSize() const = 0;
    CV_WRAP virtual void setSmallerBlockSize(int blockSize) = 0;

    CV_WRAP virtual Rect getROI1() const = 0;
    CV_WRAP virtual void setROI1(Rect roi1) = 0;

    CV_WRAP virtual Rect getROI2() const = 0;
    CV_WRAP virtual void setROI2(Rect roi2) = 0;

    /** @brief Creates StereoBM object

    @param numDisparities the disparity search range. For each pixel algorithm will find the best
    disparity from 0 (default minimum disparity) to numDisparities. The search range can then be
    shifted by changing the minimum disparity.
    @param blockSize the linear size of the blocks compared by the algorithm. The size should be
odd
    (as the block is centered at the current pixel). Larger block size implies smoother, though less
    accurate disparity map. Smaller block size gives more detailed disparity map, but there is higher
    chance for algorithm to find a wrong correspondence.

    The function create StereoBM object. You can then call StereoBM::compute() to compute
disparity for
    a specific stereo pair.
     */
    CV_WRAP static Ptr<StereoBM> create(int numDisparities = 0, int blockSize = 21);
};

/** @brief The class implements the modified H. Hirschmuller algorithm @cite HH08 that differs
from the original
one as follows:

-   By default, the algorithm is single-pass, which means that you consider only 5 directions
instead of 8. Set mode=StereoSGBM::MODE_HH in createStereoSGBM to run the full variant of the
algorithm but beware that it may consume a lot of memory.
-   The algorithm matches blocks, not individual pixels. Though, setting blockSize=1 reduces the
blocks to single pixels.
-   Mutual information cost function is not implemented. Instead, a simpler Birchfield-Tomasi
sub-pixel metric from @cite BT98 is used. Though, the color images are supported as well.
-   Some pre- and post- processing steps from K. Konolige algorithm StereoBM are included, for
example: pre-filtering (StereoBM::PREFILTER_XSOBEL type) and post-filtering (uniqueness
check, quadratic interpolation and speckle filtering).

@note
   -   (Python) An example illustrating the use of the StereoSGBM matching algorithm can be
found
        at opencv_source_code/samples/python/stereo_match.py
 */
class CV_EXPORTS_W StereoSGBM : public StereoMatcher
{
public:
    enum
    {
        MODE_SGBM = 0,
        MODE_HH    = 1,
        MODE_SGBM_3WAY = 2,
        MODE_HH4    = 3
    };

    CV_WRAP virtual int getPreFilterCap() const = 0;
    CV_WRAP virtual void setPreFilterCap(int preFilterCap) = 0;

    CV_WRAP virtual int getUniquenessRatio() const = 0;
    CV_WRAP virtual void setUniquenessRatio(int uniquenessRatio) = 0;

    CV_WRAP virtual int getP1() const = 0;
    CV_WRAP virtual void setP1(int P1) = 0;

    CV_WRAP virtual int getP2() const = 0;
    CV_WRAP virtual void setP2(int P2) = 0;

    CV_WRAP virtual int getMode() const = 0;
    CV_WRAP virtual void setMode(int mode) = 0;

    /** @brief Creates StereoSGBM object

    @param minDisparity Minimum possible disparity value. Normally, it is zero but sometimes
    rectification algorithms can shift images, so this parameter needs to be adjusted accordingly.
    @param numDisparities Maximum disparity minus minimum disparity. The value is always greater than
    zero. In the current implementation, this parameter must be divisible by 16.
    @param blockSize Matched block size. It must be an odd number \>=1 . Normally, it should be
    somewhere in the 3..11 range.
    @param P1 The first parameter controlling the disparity smoothness. See below.
    @param P2 The second parameter controlling the disparity smoothness. The larger the values are,
    the smoother the disparity is. P1 is the penalty on the disparity change by plus or minus 1
    between neighbor pixels. P2 is the penalty on the disparity change by more than 1 between neighbor
    pixels. The algorithm requires P2 \> P1 . See stereo_match.cpp sample where some reasonably good
    P1 and P2 values are shown (like 8\*number_of_image_channels\*blockSize\*blockSize and
    32\*number_of_image_channels\*blockSize\*blockSize , respectively).
    @param disp12MaxDiff Maximum allowed difference (in integer pixel units) in the left-right
    disparity check. Set it to a non-positive value to disable the check.
    @param preFilterCap Truncation value for the prefiltered image pixels. The algorithm first
    computes x-derivative at each pixel and clips its value by [-preFilterCap, preFilterCap] interval.
    The result values are passed to the Birchfield-Tomasi pixel cost function.
    @param uniquenessRatio Margin in percentage by which the best (minimum) computed cost
function
    value should "win" the second best value to consider the found match correct. Normally, a value
    within the 5-15 range is good enough.
    @param speckleWindowSize Maximum size of smooth disparity regions to consider their noise
speckles
    and invalidate. Set it to 0 to disable speckle filtering. Otherwise, set it somewhere in the
    50-200 range.
    @param speckleRange Maximum disparity variation within each connected component. If you
do speckle
    filtering, set the parameter to a positive value, it will be implicitly multiplied by 16.
    Normally, 1 or 2 is good enough.
    @param mode Set it to StereoSGBM::MODE_HH to run the full-scale two-pass dynamic
programming
    algorithm. It will consume O(W\*H\*numDisparities) bytes, which is large for 640x480 stereo
and
    huge for HD-size pictures. By default, it is set to false .

    The first constructor initializes StereoSGBM with all the default parameters. So, you only have to
    set StereoSGBM::numDisparities at minimum. The second constructor enables you to set each
parameter
    to a custom value.
     */
    CV_WRAP static Ptr<StereoSGBM> create(int minDisparity = 0, int numDisparities = 16, int
blockSize = 3,
                                          int P1 = 0, int P2 = 0, int disp12MaxDiff = 0,
                                          int preFilterCap = 0, int uniquenessRatio = 0,
                                          int speckleWindowSize = 0, int speckleRange = 0,
                                          int mode = StereoSGBM::MODE_SGBM);
};
```