```
#include <opencv2/core.hpp>
#include <opencv2/calib3d.hpp>

namespace cv {
namespace ximgproc {

//! @addtogroup ximgproc_filters
//! @{

/** @brief Main interface for all disparity map filters.
 */
class CV_EXPORTS_W DisparityFilter : public Algorithm
{
public:

    /** @brief Apply filtering to the disparity map.

    @param disparity_map_left disparity map of the left view, 1 channel, CV_16S type. Implicitly assumes that disparity
    values are scaled by 16 (one-pixel disparity corresponds to the value of 16 in the disparity map). Disparity map
    can have any resolution, it will be automatically resized to fit left_view resolution.

    @param left_view left view of the original stereo-pair to guide the filtering process, 8-bit single-channel
    or three-channel image.

    @param filtered_disparity_map output disparity map.

    @param disparity_map_right optional argument, some implementations might also use the disparity map
    of the right view to compute confidence maps, for instance.

    @param ROI region of the disparity map to filter. Optional, usually it should be set automatically.

    @param right_view optional argument, some implementations might also use the right view of the original
    stereo-pair.
     */
    CV_WRAP virtual void filter(InputArray disparity_map_left, InputArray left_view, OutputArray
```

```
filtered_disparity_map, InputArray disparity_map_right = Mat(), Rect ROI = Rect(), InputArray right_view = Mat()) = 0;
};

/** @brief Disparity map filter based on Weighted Least Squares filter (in form of Fast Global Smoother that
is a lot faster than traditional Weighted Least Squares filter implementations) and optional use of
left-right-consistency-based confidence to refine the results in half-occlusions and uniform areas.
  */
class CV_EXPORTS_W DisparityWLSFilter : public DisparityFilter
{
public:
    /** filter parameters */

    /** @brief Lambda is a parameter defining the amount of regularization during filtering. Larger values force
    filtered disparity map edges to adhere more to source image edges. Typical value is 8000.
     */
    CV_WRAP virtual double getLambda() = 0;
    /** @see getLambda */
    CV_WRAP virtual void setLambda(double _lambda) = 0;
    /** @brief SigmaColor is a parameter defining how sensitive the filtering process is to source image edges.
    Large values can lead to disparity leakage through low-contrast edges. Small values can make the filter too
    sensitive to noise and textures in the source image. Typical values range from 0.8 to 2.0.
     */
    CV_WRAP virtual double getSigmaColor() = 0;
    /** @see getSigmaColor */
    CV_WRAP virtual void setSigmaColor(double _sigma_color) = 0;

    /** confidence-related parameters */

    /** @brief LRCthresh is a threshold of disparity difference used in left-right-consistency check during
    confidence map computation. The default value of 24 (1.5 pixels) is virtually always good enough.
     */
    CV_WRAP virtual int getLRCthresh() = 0;
    /** @see getLRCthresh */
    CV_WRAP virtual void setLRCthresh(int _LRC_thresh) = 0;
```

```
    /** @brief DepthDiscontinuityRadius is a parameter used in confidence computation. It defines the size of
    low-confidence regions around depth discontinuities.
     */
    CV_WRAP virtual int getDepthDiscontinuityRadius() = 0;
    /** @see getDepthDiscontinuityRadius */
    CV_WRAP virtual void setDepthDiscontinuityRadius(int _disc_radius) = 0;
    /** @brief Get the confidence map that was used in the last filter call. It is a CV_32F one-channel image
    with values ranging from 0.0 (totally untrusted regions of the raw disparity map) to 255.0 (regions containing
    correct disparity values with a high degree of confidence).
     */
    CV_WRAP virtual Mat getConfidenceMap() = 0;
    /** @brief Get the ROI used in the last filter call
     */
    CV_WRAP virtual Rect getROI() = 0;
};

/** @brief Convenience factory method that creates an instance of DisparityWLSFilter and sets up all the relevant
filter parameters automatically based on the matcher instance. Currently supports only StereoBM and StereoSGBM.

@param matcher_left stereo matcher instance that will be used with the filter
*/
CV_EXPORTS_W
Ptr<DisparityWLSFilter> createDisparityWLSFilter(Ptr<StereoMatcher> matcher_left);

/** @brief Convenience method to set up the matcher for computing the right-view disparity map
that is required in case of filtering with confidence.

@param matcher_left main stereo matcher instance that will be used with the filter
*/
CV_EXPORTS_W
Ptr<StereoMatcher> createRightMatcher(Ptr<StereoMatcher> matcher_left);

/** @brief More generic factory method, create instance of DisparityWLSFilter and execute basic
initialization routines. When using this method you will need to set-up the ROI, matchers and
other parameters by yourself.
```

```
@param use_confidence filtering with confidence requires two disparity maps (for the left and right views) and is
approximately two times slower. However, quality is typically significantly better.
*/
CV_EXPORTS_W
Ptr<DisparityWLSFilter> createDisparityWLSFilterGeneric(bool use_confidence);

////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

/** @brief Function for reading ground truth disparity maps. Supports basic Middlebury
and MPI-Sintel formats. Note that the resulting disparity map is scaled by 16.

@param src_path path to the image, containing ground-truth disparity map

@param dst output disparity map, CV_16S depth

@result returns zero if successfully read the ground truth
 */
CV_EXPORTS_W
int readGT(String src_path,OutputArray dst);

/** @brief Function for computing mean square error for disparity maps

@param GT ground truth disparity map

@param src disparity map to evaluate

@param ROI region of interest

@result returns mean square error between GT and src
 */
CV_EXPORTS_W
double computeMSE(InputArray GT, InputArray src, Rect ROI);

/** @brief Function for computing the percent of "bad" pixels in the disparity map
(pixels where error is higher than a specified threshold)

@param GT ground truth disparity map

@param src disparity map to evaluate
```

@param ROI region of interest

@param thresh threshold used to determine "bad" pixels

@result returns mean square error between GT and src
*/
CV_EXPORTS_W
double computeBadPixelPercent(InputArray GT, InputArray src, Rect ROI, int thresh=24/*1.5 pixels*/);

/** @brief Function for creating a disparity map visualization (clamped CV_8U image)

@param src input disparity map (CV_16S depth)

@param dst output visualization

@param scale disparity map will be multiplied by this value for visualization
*/
CV_EXPORTS_W
void getDisparityVis(InputArray src,OutputArray dst,double scale=1.0);

//! @}
}
}