```
	numberOfDisparities = numberOfDisparities > 0 ? numberOfDisparities : ((img_size.width / 8) +
15) & -16;

	bm->setROI1(roi1);
	bm->setROI2(roi2);
	bm->setPreFilterCap(31);
	bm->setBlockSize(SADWindowSize > 0 ? SADWindowSize : 9);
	bm->setMinDisparity(0);
	bm->setNumDisparities(numberOfDisparities);
	bm->setTextureThreshold(10);
	bm->setUniquenessRatio(15);
	bm->setSpeckleWindowSize(100);
	bm->setSpeckleRange(32);
	bm->setDisp12MaxDiff(1);

	sgbm->setPreFilterCap(63);
	int sgbmWinSize = SADWindowSize > 0 ? SADWindowSize : 3;
	sgbm->setBlockSize(sgbmWinSize);

	int cn = img1.channels();

	sgbm->setP1(8 * cn * sgbmWinSize * sgbmWinSize);
	sgbm->setP2(32 * cn * sgbmWinSize * sgbmWinSize);
	sgbm->setMinDisparity(0);
	sgbm->setNumDisparities(numberOfDisparities);
	sgbm->setUniquenessRatio(10);
	sgbm->setSpeckleWindowSize(100);
	sgbm->setSpeckleRange(32);
	sgbm->setDisp12MaxDiff(1);
	if (alg == STEREO_HH)
		sgbm->setMode(StereoSGBM::MODE_HH);
	else if (alg == STEREO_SGBM)
		sgbm->setMode(StereoSGBM::MODE_SGBM);
	else if (alg == STEREO_HH4)
		sgbm->setMode(StereoSGBM::MODE_HH4);
	else if (alg == STEREO_3WAY)
		sgbm->setMode(StereoSGBM::MODE_SGBM_3WAY);

	Mat disp, disp8;
	//Mat img1p, img2p, dispp;
	//copyMakeBorder(img1, img1p, 0, 0, numberOfDisparities, 0, IPL_BORDER_REPLICATE);
```

```
        //copyMakeBorder(img2, img2p, 0, 0, numberOfDisparities, 0, IPL_BORDER_REPLICATE);

        int64 t = getTickCount();
        float disparity_multiplier = 1.0f;
        if (alg == STEREO_BM) {
                bm->compute(img1, img2, disp);
                if (disp.type() == CV_16S)
                        disparity_multiplier = 16.0f;
        } else if (alg == STEREO_SGBM || alg == STEREO_HH || alg == STEREO_HH4 || alg ==
STEREO_3WAY) {
                sgbm->compute(img1, img2, disp);
                if (disp.type() == CV_16S)
                        disparity_multiplier = 16.0f;
        }
        t = getTickCount() - t;
        printf("Time elapsed: %fms\n", t * 1000 / getTickFrequency());

        //disp = dispp.colRange(numberOfDisparities, img1p.cols);
        if (alg != STEREO_VAR)
                disp.convertTo(disp8, CV_8U, 255 / (numberOfDisparities * 16.));
        else
                disp.convertTo(disp8, CV_8U);

        Mat disp8_3c;
        if (color_display)
                cv::applyColorMap(disp8, disp8_3c, COLORMAP_TURBO);

        if (!disparity_filename.empty())
                imwrite(disparity_filename, color_display ? disp8_3c : disp8);

        if (!point_cloud_filename.empty()) {
                printf("storing the point cloud...");
                fflush(stdout);
                Mat xyz;
                Mat floatDisp;
                disp.convertTo(floatDisp, CV_32F, 1.0f / disparity_multiplier);
                reprojectImageTo3D(floatDisp, xyz, Q, true);
                cv::Mat colors;
                colors = img1_color;
//                cv::cvtColor(img1_color, colors, cv::COLOR_BGR2RGB);
                vi_cloud_saveXYZ(point_cloud_filename.c_str(), xyz, colors, cv::Mat());
                printf("\n");
        }

        if (!no_display) {
                std::ostringstream oss;
                oss << "disparity    " << (alg == STEREO_BM ? "bm" :
                                            alg == STEREO_SGBM ? "sgbm" :
                                            alg == STEREO_HH ? "hh" :
                                            alg == STEREO_VAR ? "var" :
                                            alg == STEREO_HH4 ? "hh4" :
                                            alg == STEREO_3WAY ? "sgbm3way" : "");
                oss << "    blocksize:" << (alg == STEREO_BM ? SADWindowSize : sgbmWinSize);
                oss << "    max-disparity:" << numberOfDisparities;
                std::string disp_name = oss.str();

                namedWindow("left", cv::WINDOW_NORMAL);
                imshow("left", img1);
                namedWindow("right", cv::WINDOW_NORMAL);
                imshow("right", img2);
                namedWindow(disp_name, cv::WINDOW_NORMAL);
                imshow(disp_name, color_display ? disp8_3c : disp8);

                printf("press ESC key or CTRL+C to close...");
                fflush(stdout);
                printf("\n");
                while (1) {
                        if (waitKey() == 27) //ESC (prevents closing on actions like taking screenshots)
                                break;
                }
        }
```