Stereo Calibration and Disparity Map Computation

April 14, 2014

Contents

We introduce how to capture image and video sequence using stereo camera in Hardware documentation Section FlyCapture 2 SDK. However, these data are not ready for depth information extraction. We need to first estimate the parameters of stereo camera model to project image pairs onto a common image plane, then compute the depth value for each world point in terms of disparity. The first step related to camera calibration and image rectification. The second step related to disparity computation. In this documentation, we will discuss these two steps in Section 1 and 2. Then we will introduce an implementation of real time disparity map computation using OpenCV in Section 3.

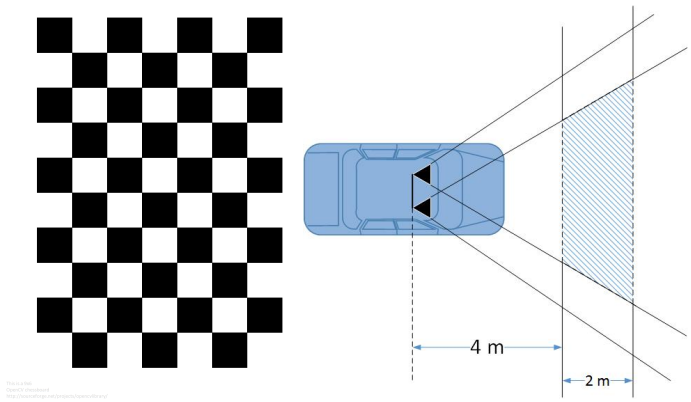
1 Calibration and image rectification

The aim of calibration is computing the relative location and orientation of the camera as well as the intrinsic parameters of the camera. Then these parameters are used for image rectification.

The main steps of calibration is: 1. take pictures of a calibration object through different view. 2. Compute the intrinsic and extrinsic parameters of left and right cameras. 3. Compute the rectified images using calibration parameters.

1.1 Calibration object

We use flat chessboard patterns shown in Fig. 1(a) as calibration object and the size of squares are $70mm \times 70mm$. This image can be found in `Data/Chessboard/OpenCV_Chessboard.png` and also can be downloaded from OpenCV online documentation here. To provide multiple views, the chessboard need to be rotated, translated and cover most of range of interest in the camera image plane. For our application, the range of interest is the area of road. Also the distance between the camera and chessboard should not be too large, since these image will give relatively high reprojection error during calibration. Fig. 1(b) shows the range for calibration. In order to obtain enough information for calibration, 20 - 30 images are needed and the more variety of orientations and positions of the chessboard in the images are better. Fig. 2 shows sample chessboard images we taken for calibration.



(a) Chessboard [**?**] (b) Range of calibration

Figure 1

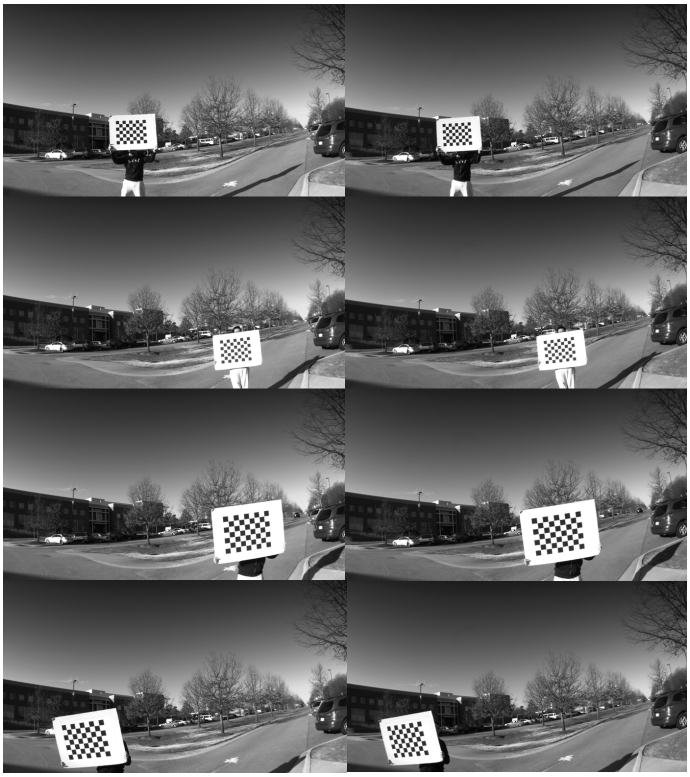


Figure 2: Chessboard images. First and second column are the images taken by left and right camera, respectively.

1.2 Calibration toolbox

We use Camera Calibration Toolbox for Matlab [**?**] to do the stereo calibration.

Matlab toolbox requires selecting four extreme corners manually. Each camera is calibrated separately and then process stereo calibration. The toolbox can be found `Src/Calibration - MATLAB/toolbox_calib.zip` and also can be downloaded here. We will briefly introduce calibration steps, for more details, please refer to the documentation of this toolbox here.

First, we calibrate the left camera. Run `calib_gui` and Fig. 3, the camera calibration toolbox window is shown:

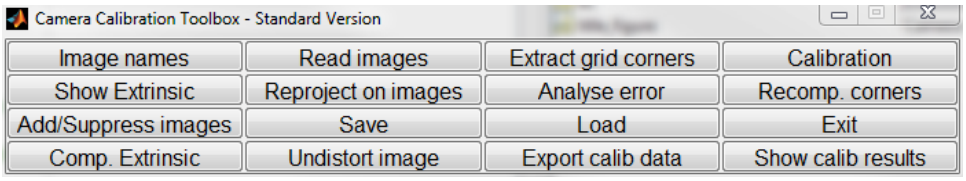


Figure 3: Calibration window

Click `Image names`, enter the name and format of chessboard images. After loading images, click `Extract grid corners` and enter the number of image to be processed. Then select the default parameters until the first chessboard image is shown. This toolbox requires pointing four extreme corners of chessboard manually. The first selected corner is the origin point of the reference frame attached to the grid and the selection order of four selected corners must be consistent for all the chessboard images. An example of corner selection is shown in left image of Fig. 4. Then the toolbox asks for the size of squares, for our case, the value is 70 *mm*. Then the estimated corner position is given in the middle image of Fig. 4. If all the estimated corners are close enough to the real corners, skip the following step. If not, please refer to here for more details about initial distortion guess. Then corner extraction result is shown as the right image of Fig. 4. Repeat these steps for the rest of images to extract corner for all the chessboard images.

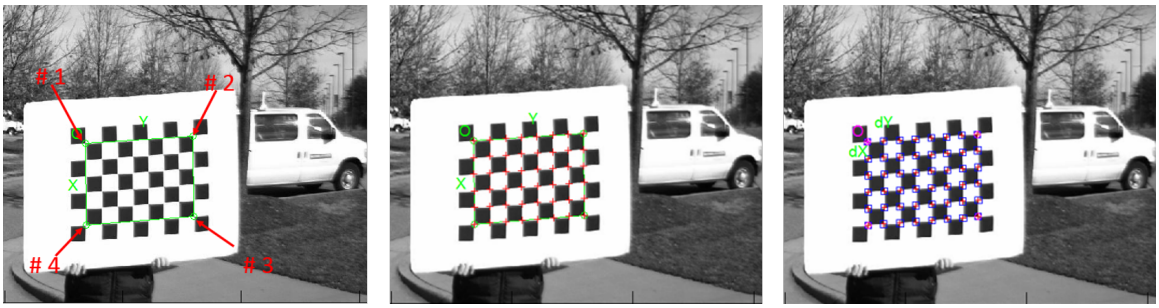


Figure 4: Corner extraction

After corner extraction, click `Calibration` in Fig. 3 to run the calibration procedure. The result is shown as below and the description of the calibration parameters are explained here.

```
 Calibration results after optimization (with uncertainties):
Focal Length:  fc = [ 1243.34836 1240.19370 ] ± [ 47.75369 47.67433 ]
Principal point:  cc = [ 965.71382 553.14746 ] ± [ 17.84984 20.64580 ]
Skew:  alpha_c = [ 0.00000 ] ± [ 0.00000 ] => angle of pixel axes = 90.00000
0.00000 degrees
Distortion:  kc = [ -0.37994 0.16183 0.00063 0.00080 0.00000 ] ± [ 0.01805
0.01900 0.00262 0.00213 0.00000 ]
Pixel error:  err = [ 0.45062 0.35574 ]
```

This toolbox will not remove the images with large re-projection error automatically, so the result is computed from all the chessboard images. To inspect which image corresponding to large error, click `Analyse error` in Fig. 3 and the re-projection error is shown as Fig. 5. Left click the point with large error and it will show the image index correspond to that point in Matlab command window. In order to get smaller error, click `Extract grid corners` to extract corners for this image again or click `Add/Suppress images` to simply remove this image.

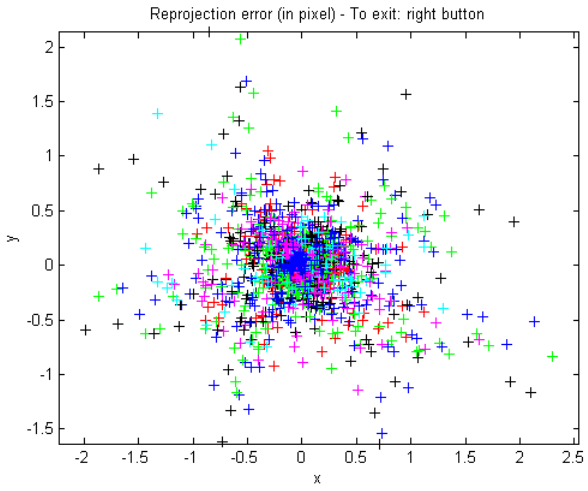


Figure 5: Re-projection error

After removal of all images with large error, run `Calibration` again to compute the new result and click `Show Extrinsic`, the 3D positions of the chessboard with respect to the camera is shown as Fig. 6. Then click `Save` to save the calibration result in file `Calib_Results.mat`. Do the same thing to the right camera images and save the calibration results in file `Calib_Results_left.mat` and `Calib_Results_right.mat` separately. Notice that the images used for calibration should be consistent for both cameras. That is, same set of images should be used to calibrate left and right camera.

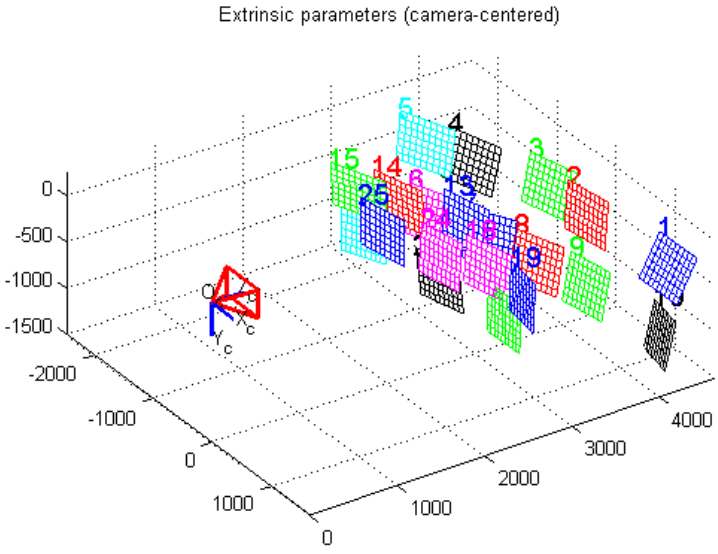


Figure 6: 3D position of chessboard with respect to the camera

The next step is stereo calibration. Run `stereo_gui` and Fig. 7, the stereo camera calibration window is shown.

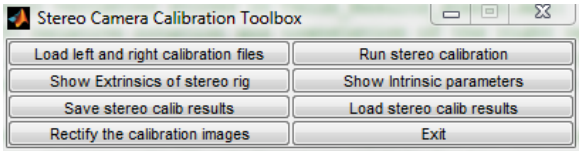


Figure 7: Stereo calibration window

Click `Load left and right calibratio files` to load the calibration results of left and right cameras then click `Run stereo calibration` to run the stereo calibration procedure. The result is shown as below and the descriptions of these parameter is explained here. This result is the calibration parameter used to rectify image pairs. Note that the intrinsic parameters of left and right cameras are re-computed. Also, the 3×3 rotation matrix stored in variable `R` is not shown here. This parameter is required as input of OpenCV image rectification which will be mentioned in Section 1.3.

```
 Intrinsic parameters of left camera:
Focal Length:  fc_left = [ 1176.49201 1172.90938 ]  [ 18.07460 18.22083 ]
Principal point:  cc_left = [ 955.32713 571.88448 ]  [ 9.94857 7.82188 ]
Skew:  alpha_c_left = [ 0.00000 ]  [ 0.00000 ] => angle of pixel axes = 90.00000
0.00000 degrees
Distortion:  kc_left = [ -0.35544 0.13716 -0.00173 0.00144 0.00000 ]  [ 0.00769
0.00793 0.00104 0.00116 0.00000 ]

Intrinsic parameters of right camera:
Focal Length:  fc_right = [ 1149.78447 1148.91459 ]  [ 18.20346 18.38693 ]
Principal point:  cc_right = [ 933.79059 558.19366 ]  [ 9.65664 8.22484 ]
Skew:  alpha_c_right = [ 0.00000 ]  [ 0.00000 ] => angle of pixel axes = 90.00000
0.00000 degrees
Distortion:  kc_right = [ -0.34594 0.12215 -0.00222 0.00302 0.00000 ]  [ 0.00815
0.00736 0.00114 0.00124 0.00000 ]

Extrinsic parameters (position of right camera wrt left camera):
Rotation vector:  om = [ -0.01382 0.03758 -0.03424 ]  [ 0.00197 0.00507 0.00073 ]
Translation vector:  T = [ -482.54751 37.10367 -5.64928 ]  [ 0.90891 0.75024
4.16050 ]
```

Click `Show Extrinsics of stereo rig`, the 3D position of chessboard with respect to left and right cameras is shown as Fig. 8. Click `Save stereo calib results` to save the stereo calibration result in `Calib_Results_stereo.mat`.

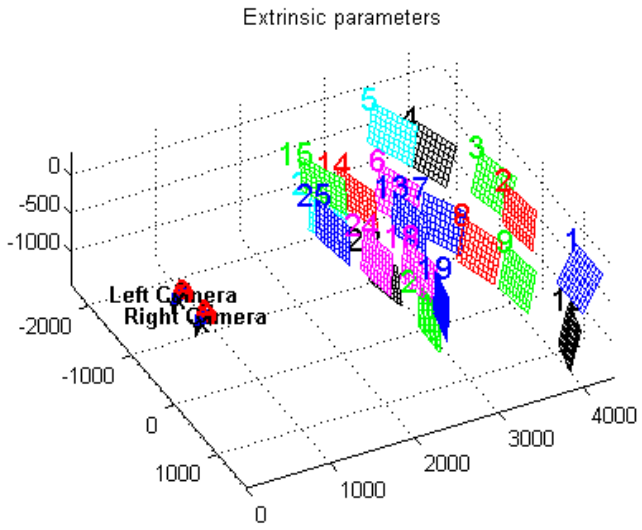


Figure 8: 3D position of chessboard with respect to two cameras

Finally, we can use the calibration result to rectify the images by clicking `Rectify the calibration images`. One example of rectified images is shown in Fig. 9.

Figure 9: Rectified images by using Matlab toolbox

1.3 Image rectification using OpenCV

Last subsection we introduce how to use Matlab calibration toolbox to obtain rectified image. The next step is rectifying images using OpenCV through the calibration result from Matlab toolbox. The source code can be found in `Src/Disparity Computation - C++ and MATLAB/C++/` combined with disparity map and confidence cue computation which will be introduced in Section 2. Input of this is sequence of image pairs or video pair, configuration parameters, calibration parameters and disparity map parameters. Video and image capture is mentioned in Hardware documentation Section FlyCapture 2 SDK. Image and video input configuration will be introduced in Section 3.

Disparity map parameters will be discussed in Section 2. Here we will talk about how to set the calibration parameter using Matlab toolbox results.

The input calibration parameters for OpenCV are saved in files `intrinsics.yml` and `extrinsics.yml` which store the intrinsic and extrinsic parameters for the stereo camera. An example of these two files corresponding to stereo calibration result shown in Section 1 are shown below and also can be found in `Src/Disparity Computation - C++ and MATLAB/C++/input/intrinsics.yml` and `Src/Disparity Computation - C++ and MATLAB/C++/input/extrinsics.yml`.

```
%YAML:1.0
%intrinsics.yml
M1: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 1176.49201, 0., 955.32713, 0.,1172.90938, 571.88448, 0., 0., 1. ]
D1: !!opencv-matrix
   rows: 1
   cols: 5
   dt: d
   data: [ -0.35544, 0.13716, 0.,0., -0.00173 ]
M2: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 1149.78447, 0., 933.79059, 0., 1148.91459, 558.19366, 0., 0., 1. ]
D2: !!opencv-matrix
   rows: 1
   cols: 5
   dt: d
   data: [ -0.34594, 0.12215, 0., 0., -0.00222 ]
```

```
%YAML:1.0
%extrinsics.yml
R: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 0.9987, 0.0340, 0.0378, -0.0345, 0.9993, 0.0132, -0.0373, -0.0145, 0.9992]
T: !!opencv-matrix
   rows: 3
   cols: 1
   dt: d
   data: [ -48.254751, 3.710367, 0.564928 ]
ImageSizeHeight: 1080
ImageSizeWidth: 1920
```

For intrinsic parameters, `M1` and `M2` are 3×3 intrinsic matrix of left and right cameras, `D1` and `D2` are 1×5 distortion vector of left and right cameras. The relationship between Matlab and OpenCV parameters is:

```
M1 = [fc_left[1], 0, cc_left[1], 0, fc_left[2], cc_left[2], 0, 0, 1]
D1 = [kc_left[1], kc_left[2], kc_left[3], kc_left[4], kc_left[5]]
M2 = [fc_right[1], 0, cc_right[1], 0, fc_right[2], cc_right[2], 0, 0, 1]
D2 = [kc_right[1], kc_right[2], kc_right[3], kc_right[4], kc_right[5]]
```

where `fc_left, cc_left, kc_left, fc_right, cc_right` and `kc_right` are Matlab stereo calibration results shown in Section 1.2.

For extrinsic parameters, `R` is the 3×3 rotation matrix, `T` is 1×3 translation vector, `ImageSizeHeight` and `ImageSizeWidth` are resolution of input sequence. The relationship between Matlab and OpenCV parameters is:

```
R = R_matlab
T = T_matlab/10
```

where `R_matlab` and `T_matlab` are rotation matrix and translation vector computed by Matlab mentioned in Section 1.2. Note that `T_matlab` is shown as `Translation vector` in Matlab stereo calibration result and `R_matlab` stored in variable `R` is not shown in the result.

Rectified and cropped image computed by OpenCV corresponding to Fig. 9 is shown in Fig. 10.

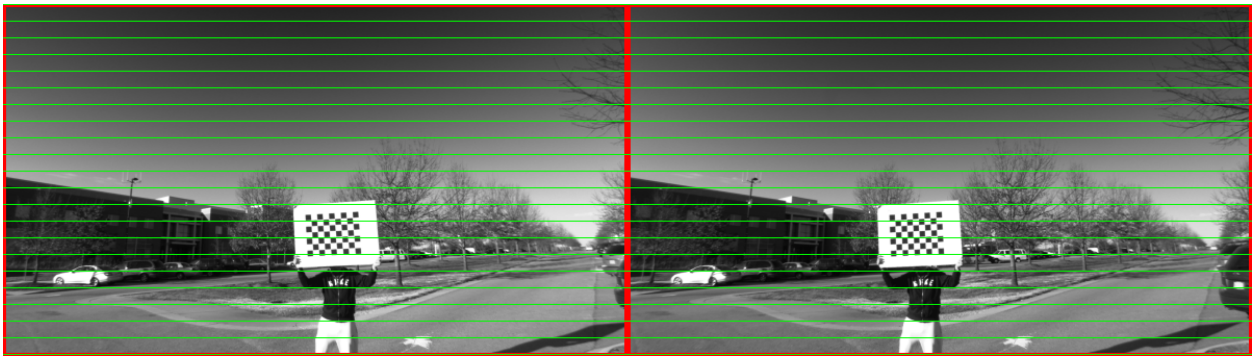


Figure 10: Rectified images

2 Disparity map computation

Depth information is extracted in terms of disparity. Disparity is the difference in location of a world point in left and right image pair. In this section we will discuss how to compute disparity map and its corresponding confidence cues using OpenCV. The source code which is the same code mentioned in Section 1.3 can be found in `Src/Disparity Computation - C++ and MATLAB/C++/`.

2.1 Disparity map

We use semi-global block matching (SGBM) [**?**] implemented in OpenCV (version 2.4.5) to compute disparity map. The input of this algorithm are rectified left and right image pair after cropping. The parameters for SGBM is stored in file `sgbmPara.yml`. The parameters we are using is shown below

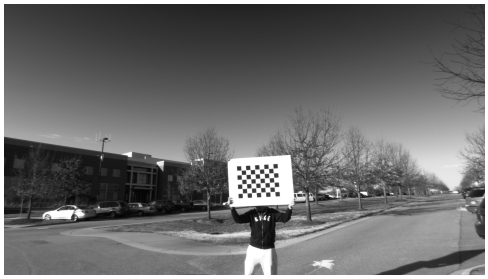
and also can be found in `Src/Disparity Computation - C++ and MATLAB/C++/input/sgbmPara.yml`.

```
%YAML:1.0
preFilterCap: 1
SADWindowSize: 3
P1: 36
P2: 288
minDisparity: 0
numberOfDisparities: 240
uniquenessRatio: 0
speckleWindowSize: 200
speckleRange: 32
disp12MaxDiff: 1
fullDP: 1
```

`minDisparity` is the minimum possible disparity value based on left image. `numberOfDisparities` is maximum disparity minus minimum disparity and this value must be divisible by 16. For explanation of other parameters please refer to OpenCV documentation here.

The disparity map result corresponding to Fig. 10 is shown in Fig. 11.

(a) Rectified left image



(b) Rectified right image

(c) Disparity map

Figure 11: Disparity map computation

2.2 Confidence

We also compute the confidence of stereo matching during disparity map computation using the metrics mentioned in [**?**]. The cost term for confidence computation is the stereo matching cost of SGBM implemented in OpenCV, that is, the accumulated costs with cost metric mentioned in [**?**].

Three confidence metrics are [**?**]:

$$PKRN = \frac{C_2 + \epsilon}{C_1 + \epsilon} - 1 \tag{1}$$

$$MLM = \frac{e^{-C_1/2\sigma^2}}{\sum e^{-C_i/2\sigma^2}} \tag{2}$$

$$LC = \frac{max(C_+, C_-) - C_1}{\gamma} \tag{3}$$

where C_1 and C_2 are the minimum and second minimum stereo matching cost, C_+ and C_- are the cost adjacent to C_1, ϵ and γ are constant for a nicely spread distribution of confidence values. Confidence cues corresponding to Fig. 11 is shown in Fig. 12.

(a) PKRN

(b) MLM

(c) LC

Figure 12: Confidence cues using three metrics

3 Real time disparity map computation

After calibration and testing the calibration result by image rectification and disparity map computation, the next step is real time image capturing and disparity computation using calibration and SGBM parameters mentioned in last two sections. The source code can be found in `Src/Disparity Computation - C++ and MATLAB/C++/`. Note that now, the disparity computation part is not real time. We will upload real time version later. The block diagram is show in Fig. 13. The input are image or video sequence and five XML and YML files storing sets of parameters. Parameter files must be in the same folder as executable file. Output are disparity map and confidence cues. For each image pair, disparity map is stored in in PNG files, which contains normalized disparity value and TXT files, which contains disparity value after multiplying 16. Confidence cues are stored in four TXT files. To show the confidence cues, the Matlab code `Src/Disparity Computation - C++ and MATLAB/MATLAB/Run_ShowCost.m` is used. When running, sequence is captured as mentioned in Hardware documentation Section FlyCapture 2 SDK. The unrectified image pairs are read according to configuration files `Config.xml` and `ImgList.xml`. Then by using the calibration parameters stored in `intrinsics.yml` and `extrinsics.yml`, rectified image pairs are obtained. At last, the disparity map is computed by SGBM using parameters stored in `sgbmPara.yml`. Image rectification, disparity map computation and the corresponding input files are discussed in Section 1.3 and 2. In this section, we will introduce the configuration input files: `Config.xml` and `ImgList.xml`.

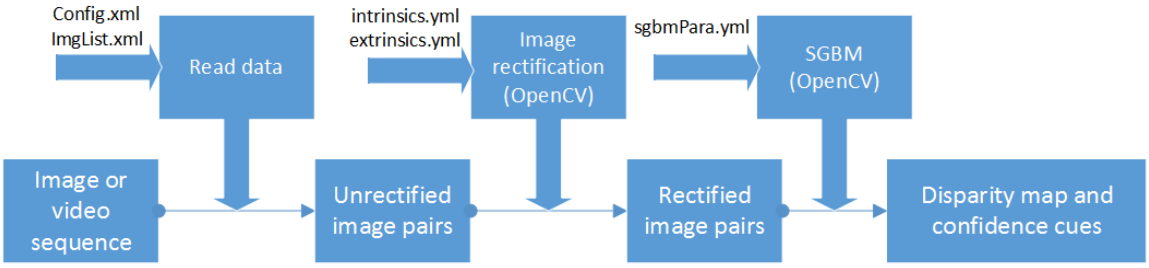


Figure 13: Block diagram of disparity computation

An example of `Config.xml` is shown below and also can be found in `Src/Disparity Computation - C++ and MATLAB/C++/input/Config.xml`. This file stores flag values and data save path, where "1" indicates "true". By using this file, all the results files will be saved in folder `output`.

```
<?xml version="1.0"?>
<opencv_storage>
  <ShowRectify>
    1
  </ShowRectify>
  <ShwoDisparity>
    0
  </ShwoDisparity>
  <SaveDisparity>
    0
```

```
  </SaveDisparity>
  <ComputeConfidence>
    0
  </ComputeConfidence>
  <SavePath>
    "output/"
  </SavePath>
</opencv_storage>
```

An example of `ImgList.xml` is shown below and also can be found in `Src/Disparity Computation - C++ and MATLAB/C++/input/ImgList.xml`. This file stores the information of input file type and input file names. If `imageorvideo` is "1", the input is image sequence otherwise input is video. `leftvideo` and `rightvideo` are left and right video file names. `inputpath` indicates the path of image data. `leftname` and `rightname` are the basename of left and right images without index suffix. `imageformat` is the format of input image sequence and `imagelist` is the list of image index to be processed. By using this file, the input is image sequence with three image pairs and the files are: `left_1.jpg`, `right_1.jpg`, `left_2.jpg`, `right_2.jpg`, `left_3.jpg` and `right_3.jpg` stored in folder `input`.

```
<?xml version="1.0"?>
<opencv_storage>
  <imageorvideo>
   1
  </imageorvideo>
  <leftvideo>
    "leftvideo.avi"
  </leftvideo>
  <rightvideo>
    "rightvideo.avi"
  </rightvideo>
  <inputpath>
    "input/"
  </inputpath>
  <leftname>
    "left_"
  </leftname>
  <rightname>
    "right_"
  </rightname>
  <imageformat>
    "jpg"
  </imageformat>
  <imagelist>
    "1"
    "2"
    "3"
  </imagelist>
```

```
</opencv_storage>
```