Technical Manual for Calibration of the Teleimmersion System

(ver. 0.31)

Dr. Gregorij Kurillo
Teleimmersion Lab, University of California, Berkeley *

May 15, 2009

Contents

*email: `gregorij@eecs.berkeley.edu`

1 Introduction

The calibration of the Teleimmersion system is performed using hierarchical approach. First, the cameras inside each cluster are calibrated using Zhang's method with checkerboards where the internal camera parameters and the pose of the cameras in the cluster are obtained. The parameters are optimized using non-linear optimization to further reduce the errors. Finally, the position and orientation of all the clusters inside the Teleimmersion system with regard to the selected reference camera are determined using LED calibration. The calibration is performed with a calibration bar consisting of two bright LEDs on each end. The algorithm first pairwise calibrates neighboring cameras using fundamental matrix decomposition . In the last step, the parameters are optimized using sparse bundle adjustment with Levenberg-Marquardt optimization method.

The calibration package for the Teleimmersion system consists of the following programs:

- `TICalibImages.exe` - graphic user interface for camera setting and cluster calibration
- `Calib_new.exe` - geometric cluster calibration
- `CalibLED.exe` - client program for LED data collection
- `CalibExtrinsic.exe` - extrinsic calibration of the clusters
- `ConvertCamCfg.exe` - conversion of calibration parameters between `results.yml` and `camcfg.ini` file formats

The following files are also created or used for the calibration:

- `camcfg.ini` - configuration file for `INTI` 3D reconstruction program
- `calibration.yml` - configuration file for `TICalibImages`
- `results.yml` - results of geometric calibration of camera cluster (array), produced by `Calib_new.exe`
- `colors.txt` - color calibration reference values
- `results_color.yml` - results of the color calibration, produced by `Calib_new.exe`
- `led_c??_?` - results of LED marker capturing, data input for `CalibExtrinsic.exe`

To run the calibration programs the following OpenCV DLL file shave to be installed and registered: `cv100.dll`, `cxcore100.dll`, and `highgui100.dll`. Note, if you are using newer OpenCv version, these files should be updated accordingly. Additionally, to use `TiCalibImages`, you must install PointGrey FlyCap driver (ver 1.7). The applications should be installed with the provided setup file. Some files may link to Microsoft dynamic libraries (DLL files), such as assemblies Microsoft.VC80.CRT (module: Microsoft_VC80_CRT_x86.msm) and Microsoft.VC80.MFC (module:

Microsoft_VC80_MFC_x86.msm). In case of manual installation (i.e. copying the files), the necessary DLL files should be copied and registered with the operating system (using regsvr32.exe). If setup package is not used Microsoft Visual C++ 2005 SP1 Redistributable Package (x86) from microsoft.com has to be installed if the computer does not have Visual Studio 2005 installed.

2 Cluster Calibration

2.1 Software

2.1.1 TICalibImages

Program `TiCalibImages` is used to adjust the camera parameters (e.g. gain, shutter, white balance) and to capture checkerboard data for the cluster calibration. To run the program, first edit the file `camcfg.ini` to display correct serial numbers of the cameras on the bus and the correct number of cameras. When running `TiCalibImages`, a dialog box with listed serial number will be displayed. Up to four cameras can be shown simultaneously.

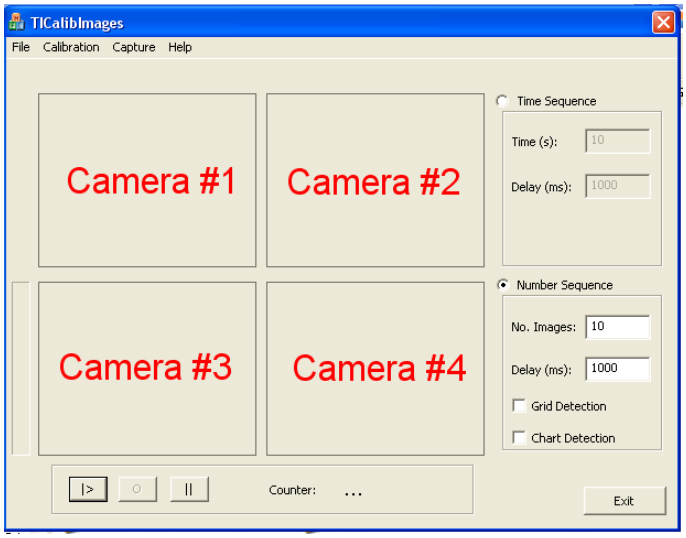


Figure 1: Main dialog window for `TICalibImages`

Graphic user interface for `TiCalibImages` is shown in Figure 1. The buttons on the bottom are similar to VCR controls and allow the user to start live capturing, pause capturing or record the images. When recording images, dialog box for folder selection will appear. Folder name will be automatically created from current date information. Afterwards, image frames will be recorded. The number of images to record can be controlled by changing the settings on the right side of the dialog (Figure 1). Two recording modes are possible: (1) recording a time sequence with specified

time of recording and delay between frames or (2) number sequence recording where the number of images and the delay between the capturing are specified.

Before calibration, the camera parameters should be properly adjusted. Cameras can be adjusted by opening camera controls dialog box (Figure 2) in the menu `Calibration` || `Camera Settings`. The dialog allows control of individual cameras as well as automatic calibration. Camera parameters can also be adjusted to a reference camera values. `Photometric Calibration` runs LM algorithm in real-time loop between the selected camera and the reference camera. The camera images should have similar intensity. `Color Calibration` calibrates the color camera white balance for blue and red based on histogram of the MacBeth checkerboard. Note that these functions are still in experimental phase.

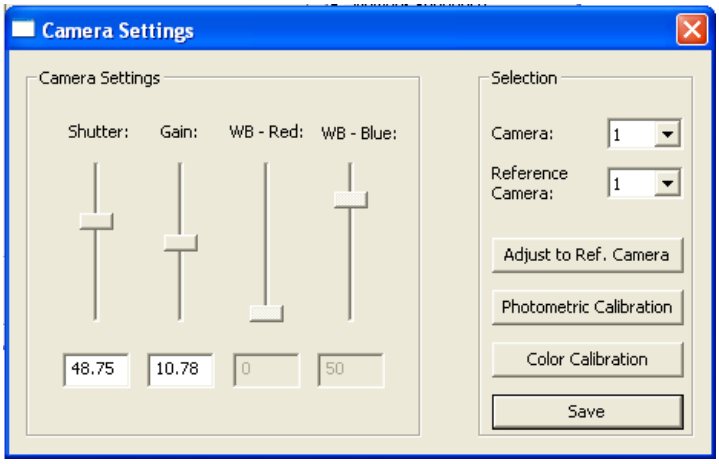


Figure 2: Camera setting dialog.

2.1.2 Calib_new

The geometric calibration of the cluster is performed using the checkerboard images that were collected using the program `TICalibImages` or by running `Calib_new.exe` from the command line. The calibration will provide internal parameters of the cameras (i.e. calibration matrix, distortion parameters) and geometric configuration of the cluster (i.e. relative pose of cameras to the reference camera). These results will be stored in `results.yml` file which can be converted to the standard configuration file for the TI system `camcfg.ini` using `TICalibImages` application or a separate executable `ConvertCamCfg.exe` which is now included in the INTI project.

The calibration program can be configured via the configuration file `calibration.yml` which contains the following information (see *Appendix* for an example):

- `number of cameras` ... Number of cameras used for the calibration

- `camera list`* ... Specifies list of camera indices. The list should match the names of 'list' files.
- `color bayer`* ... Specifies if the camera images are stored as the raw bayer [bggr] format (1) or as regular rgb color format.
- `refcam` ... Index of the reference cameras (starts with 0).
- `height` ... Height of the checkerboard (number of squares in vertical direction).
- `width` ... Weight of the checkerboard (number of squares in horizontal direction).
- `size` ... Size of the square of the checkerboard (in mm)
- `search window`* ... Size of the search window for grid refinement process (in pixels). Typical value is 10, low resolution images or small checkerboards may require smaller search window.
- `color calibration`* ... Color camera can be corrected for color if colored checkerboard was used for the image collection. The process will require `colors.txt` file to define individual colors.
- `optimize distortion` ... Set to 1 to optimize distortion of the lens; else set to 0.
- `optimize cameras` ... Set to 1 to optimize internal parameters of cameras before global calibration of the cluster; else set to 0.
- `optimize internal` ... Set to 1 to include the internal parameters of the cameras in the global optimization; else set to 0 (better accuracy).
- `error threshold`* ... Error threshold for the maximal reprojection error allowed on each image set. If the maximal reprojection is bigger, the images will be excluded from the calibration for that frame number. Note the change of the key from the original calibration code `error_thresh`!
- `color threshold`* ... The parameter controls color calibration threshold where the value defines diagonal elements in the color transformation matrix. Typical value can be set between 2.0 and 2.5.
- `path` ... Folder path where the images and the corresponding 'list' files are stored.

The parameters in the configuration file can be set manually and some can be set automatically through the menus of `TICalibImages`. In the image path, the program searches for text files named `listX.txt` (where `X` is index of camera, `X` $= 1, 2, 3...$) which contain the list of images used for calibration. The order and number of images has to correspond among the cameras. The program sequentially loads the images and automatically detects the grid in the checkerboard. If the

checkerboard cannot be detected reliably, the particular frame will be omitted in the calculations for all the cameras. The error threshold can be set to eliminate images where the errors are too large.

The calibration program will first find initial guess on internal and external camera parameters and run the necessary nonlinear optimization to refine these parameters.

2.2 Data Collection

When capturing checkerboard images for calibration, check "Grid Detection" to automatically record only images where the calibration grid has been found in all cameras simultaneously. The settings for the calibration grid size can be changed through dialog (Figure 4) located in the menu `Calibration` || `Calibration Settings`. The width, height and the size of the checkerboard square should correspond to the actual checkerboard used during the calibration. Note that the same settings are later used by the calibration program. The dialog also controls some of the options for the calibration algorithm such as distortion optimization, separate calibration of cameras, optimization of internal parameters during global cluster optimization, and the error threshold (images with higher error will be removed from the set).

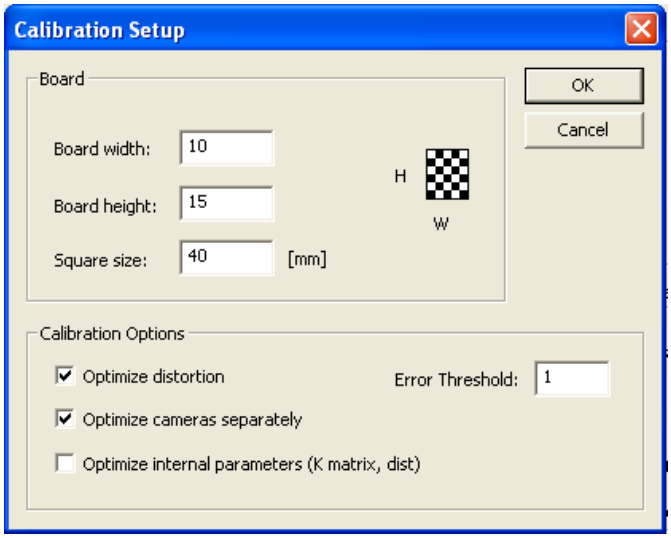


Figure 3: Calibration setup dialog.

After the cameras are adjusted properly, images can be captured. To start image capturing, select either `Time Sequence` or `Number Sequence` on the main dialog window. For checkerboards acquisition the later selection should be made. By checking `Grid Detection`, the checkerboard will be automatically segmented from the image and the frame will be captured only when the grid is detected in all the cameras. The recording of the images is initiated by clicking on the

recording button and setting the destination folder for the images. The images will be saved with the filename as `image_0X_cY_f000Z.pgm` where `X` represents cluster number, `Y` camera number in the cluster, and `Z` consecutive frame number. The folder will also contain a copy of the configuration file `calibration.yml`, which can be used to run calibration on the same set of images at a later time.

Image files can also be collected manually using a custom developed software for any type of camera. User must save their own `listX.txt` files for each of the cameras X to point the calibration program to the image data. In this case `calib_new` application should be run manually from the command prompt.

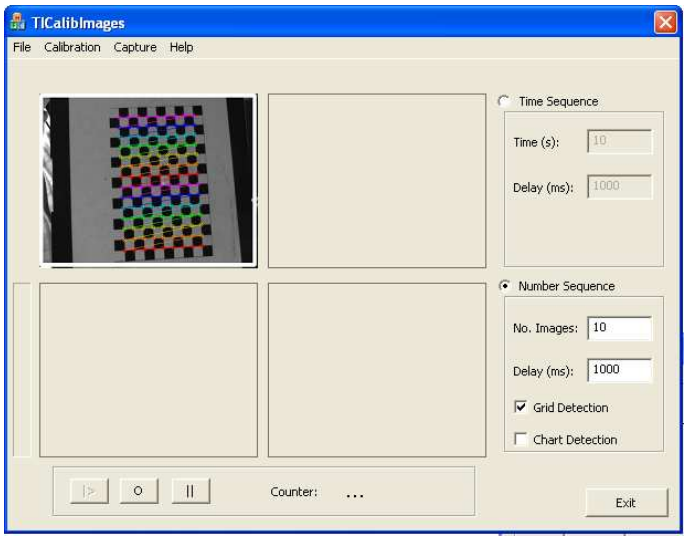


Figure 4: Automatic grid detection during calibration.

2.3 Results

The calibration of the cluster can be started in the menu `Calibration` || `Calibrate` or by running `Calib_new.exe` from command prompt. The calibration program will run through all the images and calculate the camera parameters. Press `<Enter>` key to proceed through the program. Finally, the results will be saved in `results.yml` file (see *Appendix* section for an example). To get proper formating for `titrino` program used by the Tele-immersion system, the parameters have to be converted. Conversion can be performed in the menu `Calibration` || `Convert File` and saved as `camcfg.ini`.

The results of the calibration can be checked in the `results.yml` file. Mean reprojection error should typically range between 0.10 (or less) to 0.15 pixels (e.g. size of image 640x480). The size of the error however depends on the image resolution and type of imager (color or grayscale). Images

with errors larger than specified threshold (parameter `error threshold`) will be removed from the set. The value of this threshold can be set between 0.8 and 2 pixels for good results. If the average or the maximal error is still too large, new set of images should be collected and the calibration should be repeated.

Calibration program will display final results in a form of histograms of error distribution. The histograms are displayed for each of the cameras as shown in Figure 5.

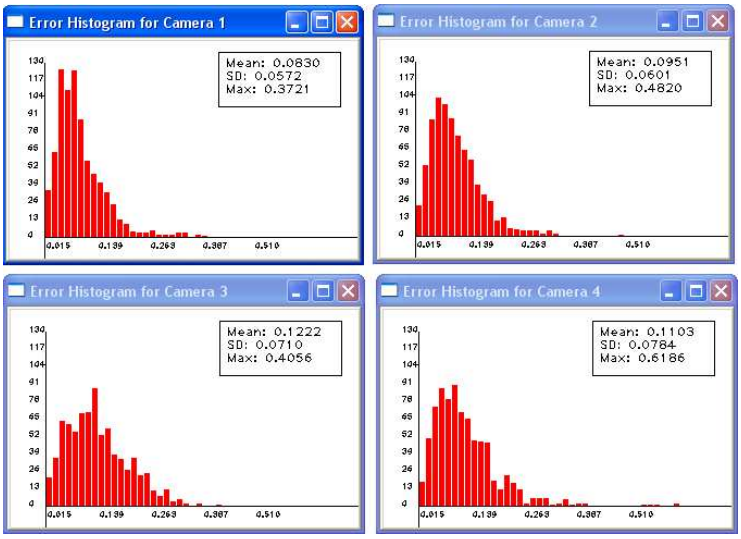


Figure 5: Results of cluster calibration are presented with reprojection error histograms for each camera.

3 External Calibration

3.1 Software

3.1.1 CalibLED

`CalibLED` is the client program for LED marker based calibration. The program works with Point-Grey (Dragonfly) cameras using the same c++ library as the reconstruction software. The algorithm searches for one or more markers (default is 2) in each captured frame and stores the position of the marker in image space. The image is analyzed by several filters and ellipse-finding algorithm to detect light emitting markers. The exact position of the marker is calculated using squared gray scale centroid method to obtain sub-pixel location. For the global calibration of the teleimmersion system, 2 markers with constant distance are used. This calibration bar should include two bright LEDs with wide emitting angle (red LED are preferred since they work well with gray-scale cameras). The location of the markers is saved for each frame into ASCII file named `led_c0X_Y.txt`, where `X` represents cluster number and `Y` represents camera index inside the cluster. Each line of the file represents one captured frame. The first column represents the frame number, the next two columns are x and y image coordinates for each detected marker. In case one or both markers are not visible, [-1 -1] values are entered for the current frame.

When calibrating the cameras, the trigger server has to be started first to synchronize capturing on multiple clusters (cameras):

```
tiserver.exe num: 1 eframes: 2
```

Next `CalibLED` can be started on cluster machine with the following parameters which define the gain and shutter of the cameras:

```
CalibLED.exe gain: 2.0 shutter: 5.0 trigger: 169.229.144.106
```

The LED calibration also allows for real-time viewing of the marker detection by executing the application with the following parameter:

```
CalibLED.exe show: 1
```

Background subtraction can be turned on (default) or off:

```
CalibLED.exe background: 1
```

The selection of `gain` and `shutter` parameters depends on the camera setting and illumination. Additional parameter `markers` can be used to specify number of markers to be detected (in case of other calibration schemes).

3.1.2 CalibExtrinsic

Geometric calibration (i.e. determination of position and orientation) of the clusters is performed by `CalibExtrinsic` program. The algorithm assumes that two markers were used for data collection. The data point files on all cameras have to be properly synchronize to find the correspondence among different views. The extrinsic calibration is based on fundamental matrix decomposition and graph theory to find initial guess of the parameters. Finally the parameters are refined using sparse Levenberg-Marquardt algorithm. The program checks for the configuration file named `calib_ex.yml` (see *Appendix* for example) which stores the parameters for the calibration. The parameters can be adjusted by editing this file. The program loads LED data files with the name `led_c0X_Y.txt` (where `X` represents camera number and `Y` the cluster number).

The following command line is used to calibrate multiple cameras (e.g. cameras 1, 2, 3, and 4, where 3 is reference camera):

```
CalibExtrinsic.exe 3 1 2 4
```

Note that the first camera index will be used as the reference camera. The cluster numbers should match the numbers saved by the capture program `CalibLED`. Configuration file `camcfgXX.ini` has to be available in the same folder to obtain internal camera parameters needed for the calibration. The results of the calibration will be saved automatically in `camcfgXX.ini` file located in the same folder.

3.2 Data Collection

To find the proper gain and shutter setting for `CalibLED`, run `TICalibImages` and adjust the cameras until mainly the LED markers will be visible in the image without any other point (or circular) light sources in the screen. In general the lights can stay turned on since the detection algorithm will threshold the image, but in some cases turning some of the lights off may help the detection. The recorded data files should be checked to see if the marker positions are changing from frame to frame. If the same number is recorded in several frames, the captured "marker" was probably part of the background. Reduce illumination, change camera settings, and/or eliminate background objects that disturb marker tracking. It is not necessary for all the cameras to see the markers simultaneously. When calibrating, make sure that different camera (cluster) pairs have sufficient number of common points (frames with visible markers). The calibration bar should be kept more or less vertically and positioned inside the entire viewing volume. The calibration error will be smaller if several cameras share the same space.

3.3 Results

Successful calibration will determine in camera pose relative to the selected reference camera. If any of the cameras is unable to calibrate, its translation and rotation parameters are set to $[0, 0, 0, 0, 0, 0]$. In case of unsuccessful calibration, the camera should be removed from the set or the system should

be re-calibrated. The results of the external calibration are automatically saved in `camcfg0X.ini` files which can then be renamed to `camcfg.ini` and copied to the corresponding cluster machines.

4 Matlab Scripts

Matlab scripts are allow review of the results of the cluster and external calibration. The scripts read the ascii files generated by the calibration programs. In most cases, m-script has to be modified to display the correct path and the name of the data file.

Cluster calibration can be reviewed by running `calib_cluster.m` script. To run the file, point the path to folder containing `calibration.yml` (calibration parameters), `results.yml` (calibration results), and `matrx.txt` (data points generated by the calibration).

External calibration can be reviewed by running `ext_calib.m` script. The results will show the reconstructed 3D path of the calibration bar and the reconstructed length before and after running non-linear optimization. The script will load files `results3D.txt` and `results3D_2.txt` which contain 3D points. Every other line of the captured two markers represents a new frame inside the file.

Calibrated cluster position can be reviewed by running `cam_layout.m` script. The script loads `camcfgXX.ini` files from designated path. By specifying `CameraVector` variable, selection can be made on what cameras should be loaded. The script can also compare results of the previous calibration (or similar) by activating flag `bCompareOn`.

Note, that the scripts were generated mainly for debugging purposes.

5 Appendix

5.1 Configuration file `calibration.yml`

The calibration file `calibration.yml` has the format shown below. Note that some of these features (*) are only supported in the newest release of the camera calibration software.

```
------------------
%YAML:1.0
number of cameras: 4
camera list: "1 2 3 4"
color bayer: "0 0 0 0"
refcam: 0
number of images: 20
height: 6
width: 9
size: 35
search window: 10
color calibration: 0
optimize distortion: 1
optimize cameras: 1
optimize internal: 0
error threshold: 2.0
color threshold: 2.5
path: ../../Data/Capture_2009_3_4/
------------------
```

Explanation of parameters:

- `number of cameras` ... Number of cameras used for the calibration.
- `camera list`* ... Specifies list of camera indices. The list should match the names of 'list' files.
- `color bayer`* ... Specifies if the camera images are stored as the raw bayer [bggr] format (1) or as regular rgb color format.
- `refcam` ... Index of the reference cameras (starts with 0).
- `height` ... Height of the checkerboard (number of squares in vertical direction).
- `width` ... Weight of the checkerboard (number of squares in horizontal direction).

- `size` ... Size of the square of the checkerboard (in mm).
- `search window`* ... Size of the search window for grid refinement process (in pixels). Typical value is 10, low resolution images or small checkerboards may require smaller search window.
- `color calibration`* ... Color camera can be corrected for color if colored checkerboard was used for the image collection. The process will require `colors.txt` file to define individual colors.
- `optimize distortion` ... Set to 1 to optimize distortion of the lens; else set to 0.
- `optimize cameras` ... Set to 1 to optimize internal parameters of cameras before global calibration of the cluster; else set to 0.
- `optimize internal` ... Set to 1 to include the internal parameters of the cameras in the global optimization; else set to 0 (better accuracy).
- `error threshold`* ... Error threshold for the maximal reprojection error allowed on each image set. If the maximal reprojection is bigger, the images will be excluded from the calibration for that frame number. Note the change of the key from the original calibration code `error_thresh`!
- `color threshold`* ... The parameter controls color calibration threshold where the value defines diagonal elements in the color transformation matrix. Typical value can be set between 2.0 and 2.5.
- `path` ... Folder path where the images and the corresponding 'list' files are stored.

5.2 Configuration file `camcfg.ini`

Example of calibration file used for `camcfg.ini` used by `titrino.exe`. Note, that "..." symbols mean that the parameters should continue in the same line.

```
;-------------------------------
;
; Cluster Information
;

[cluster]
;
; Cameras Information
;
camnum=4
order=4130122 4130113 4130104 4010948
hostnum=11
Rc=-0.793160475 -0.019095664 0.608713247 0.055736734 0.993037561 ...
... 0.103777741 -0.606456823 0.116240091 -0.786573813
Tc=-1051.679602834 26.132155767 3333.748736629
Rw=-0.997099 0.0756258 0.00862902 -0.00961586 -0.0126945...
... -0.999873 -0.0755067 -0.997055 0.0133849
Tw=-0.959695 1295.94 1950.42

[camera_0]
shutter=52.500000
gain=16.719999
KK0=555.17620850 0.00000000 332.46914673 0.00000000 555.87188721 ...
... 236.09780884 0.00000000 0.00000000 1.00000000
KKn=554.53369141 0.00000000 391.93820190 0.00000000 555.08721924 ...
... 223.24569702 0.00000000 0.00000000 1.00000000
Rect=0.99991739 -0.01193877 0.02087711 -0.00861281 1.00011337 ...
... 0.01060038 -0.02100603 -0.01077932 0.99992782
kc=-0.28339455 0.14525101 0.00116896 0.00129457 0.0000000000000
tc_rect=-112.63924408 -0.00000001 0.00000026
resolution=640 480
wbalance=50 20
CCmat=1.05651045 -0.24070223 0.06472625 47.93445206 -0.31458977 ...
... 1.76734614 -0.28905934 44.70959473 0.01096128 -0.43597835 2.10874581 28.12573624
;-------------------------------
```

Explanation of parameters:

- `camnum` ... Number of cameras used for the calibration.
- `order` ... List of camera serial numbers in order (last is the color camera).
- `hostnum` ... Host number of the cluster. This number is assigned to cluster number and port number when communicating with the renderer (e.g. hostnum=11, port 3011).
- `Rc` ... The rotation matrix of the cluster with respect to the global central (reference) camera, row by row.
- `Tc` ... The translation vector the cluster with respect to the global central (reference) camera, row style.
- `Rw` ... The rotation matrix of the reference cluster with respect to the global (physical) coordinate system, row by row, as defined by `CalibFromTarget` application.
- `Tw` ... The translation vector of the reference cluster with respect to the global (physical) coordinate system, row style, as defined by `CalibFromTarget` application.

  For each [camera_i] (i=0,1,2,3), the following is defined:

- `shutter` ... Shutter setting for the camera (in ms).
- `gain` ... Gain setting for the camera (in dB).
- `KK0` ... Calibration matrix of the camera, row by row.
- `KKn` ... Corrected calibration matrix as defined by average matrix of the stereo cameras, row by row.
- `Rect` ... Rotational matrix for image rectification.
- `tc_rect` ... Translation vector for image rectification (x component represents distance between cameras, while other components should be very small).
- `kc` ... The vector of distortion coefficients as defined by the calibration (two radial, two tangential components, one zero element).
- `resolution` ... Native resolution of the camera at which the calibration parameters were obtained.
- `wbalance` ... White balance values for red and blue components as defined by the camera.
- `CCmat` ... Color correction matrix (3x4), row by row. Note, color correction has to be turned on inside the INTI file.

5.3 Results file `results.yml`

Example of results file `results.yml` obtained by the cluster calibration:

```
%YAML:1.0
Number of cameras: 4
Reference camera: 1
Number of images: 16
MeanError: 0.1761958748102188
SDError: 0.1057157367467880
MaxError: 0.9091453552246094
#  ************************
Camera_0:
   path: Capture_2009_3_4
   imgWidth: 640
   imgHeight: 480
   K: !!opencv-matrix
      rows: 3
      cols: 3
      dt: f
      data: [ 558.49035645, 0., 334.36587524, 0., 559.26623535,
          233.55244446, 0., 0., 1. ]
   Dist: !!opencv-matrix
      rows: 1
      cols: 4
      dt: f
      data: [ -0.28504518, 0.14290571, 1.65544252e-003, 6.55467040e-004 ]
   PosRel: !!opencv-matrix
      rows: 1
      cols: 6
      dt: f
      data: [ 112.38521576, 1.04459536, 2.53007770, 0.01057885,
          -0.05979063, 1.91509794e-003 ]
#  ************************
...
```

Explanation of parameters:

- `Number of cameras` ... Number of cameras used for the calibration.
- `Reference camera` ... Reference camera index.

- `Number of images` ... Final number of images used after the error thresholding.
- `MeanError` ... Mean reprojection error from all images for all cameras.
- `SDError` ... Standard deviation of the reprojection error from all images for all cameras.
- `MaxError` ... Maximal reprojection error from all images for all cameras.

  For each camera (Camera_i) the following is defined:

- `path` ... Folder path of the stored checkerboard images.
- `imgWidth` ... Image width in pixels.
- `imgHeight` ... Image height in pixels.
- `K` ... Camera calibration matrix in OpenCV matrix format.
- `Dist` ... Lens distortion coefficients OpenCV vector format.
- `PosRel` ... Position and orientation of the camera with regard to the reference camera inside the cluster. First three values describe camera position in units while last three values describe Rodriguez parameters of rotation (see OpenCV manual for conversion between Rodriguez parameters and rotation matrix).

5.4 Results file `results_color.yml`

Example of results file `results.yml` obtained by the cluster calibration:

```
%YAML:1.0
Number of cameras: 4
Number of color cameras: 1
#  ************************
Camera_3:
   path: Capture_2009_3_4
   A_mat: !!opencv-matrix
      rows: 3
      cols: 4
      dt: f
      data: [ 1.11181641, -0.61547977, 0.31481275, 47.56661987,
          -0.33642229, 1.55430436, -0.31427813, 54.81690979,
          2.47441977e-003, -0.54995143, 1.97474146, 41.16526031 ]
#  ************************
```

Explanation of parameters:

- `Number of cameras` ... Total number of cameras used for the calibration.
- `Number of color cameras` ... Number of color cameras calibrated.

  For each camera (Camera_i) the following is defined:
- `A_mat` ... Color correction matrix (3x4).

5.5 Data conversion between `results.yml` and `camcfg.ini`

Explanation of data conversion from `results.yml` to `camcfg.ini` (in Matlab format).

```
function STR = get_tele_param( STR_in )

%===================
%   load a cluster
%===================

% Rc1,...Rc2 - rotational matrix between reference camera and camera i (in ref. c.s.)
% Tc1,...Tc2 - translation vector between reference camera and camera i (in ref. c.s.)
% KK1,...KK2 - calibration matrix of camera i

r1 = STR_in.Rc1; t1 = STR_in.Tc1; kk1 = STR_in.KK1;
r2 = STR_in.Rc2; t2 = STR_in.Tc2; kk2 = STR_in.KK2;
r3 = STR_in.Rc3; t3 = STR_in.Tc3; kk3 = STR_in.KK3;
r4 = STR_in.Rc4; t4 = STR_in.Tc4; kk4 = STR_in.KK4;

% 2 ... index of reference camera

rc21 = r2*inv(r1); tc21 = t2-rc21*t1;
rc22 = r2*inv(r2); tc22 = t2-rc22*t2;
rc23 = r2*inv(r3); tc23 = t2-rc23*t3;
rc24 = r2*inv(r4); tc24 = t2-rc24*t4;

e1 = tc22-tc21; e1 = e1/norm(e1)
e2 = [-e1(2);e1(1);0]; e2 = e2/norm(e2)
e3 = cross(e1,e2)

R2_vir = [e1,e2,e3]

% "Rect" in camcfg.ini
Rrect1 = inv(R2_vir)*(rc21)
Rrect2 = inv(R2_vir)*(rc22)
Rrect3 = inv(R2_vir)*(rc23)
Rrect4 = inv(R2_vir)*(rc24)

% "tc_rect" in camcfg.ini
tc21_rec = inv(R2_vir)*(tc21)
tc22_rec = inv(R2_vir)*(tc22)
```

```
tc23_rec = inv(R2_vir)*(tc23)
tc24_rec = inv(R2_vir)*(tc24)

% "KK0" in camcfg.ini
kk10 = kk1; kk20 = kk2; kk30 = kk3; kk40 = kk4;

%-----------------------------
% calculation camera matrix
%-----------------------------
cx_ = (kk10(1,3)+kk20(1,3)+kk30(1,3))/3;
cy_ = (kk10(2,3)+kk20(2,3)+kk30(2,3))/3;

% "KKn" in camcfg.ini
kk1n=[kk2(1,1),0,cx_;0,kk2(2,2),cy_;0,0,1];
kk2n=[kk2(1,1),0,cx_;0,kk2(2,2),cy_;0,0,1];
kk3n=[kk2(1,1),0,cx_;0,kk2(2,2),cy_;0,0,1];
kk4n=[kk2(1,1),0,cx_;0,kk2(2,2),cy_;0,0,1];
```

5.6 Configuration file `calib_ex.yml`

The calibration file `calib_ex.yml` used to set up the external calibration parameters has the following format:

cm

```
%------------------
%YAML:1.0
ThreshNumberOfPoints: 30
ThreshRansacRatio: 0.30
bSBAConst: 1
bOptimizeBar: 1
ThreshStereoBarError: 0.01
BarDistance: 317.0
CameraIndex: 2
path: x
%------------------
```

Explanation of parameters:

- `ThreshNumberOfPoints` ... Threshold for minimal number of points required for calibration of a camera pair.
- `ThreshRansacRatio` ... Threshold for RANSAC algorithm. The threshold represents maximal percentage of points viewed by any two cameras that can be invalid (e.g. `ThreshRansacRatio: 0.30` means 30% of points can be excluded).
- `bSBAConst` ... (If 1) Use fixed distance between the points during sparse bundle adjustment.
- `bOptimizeBar` ... Optimize the distance of the mean distances acquired for each camera pair during the triangulation.
- `ThreshStereoBarError` ... Threshold for 3D error of the reconstructed distance (in % of distance).
- `BarDistance` ... Length of the calibration bar (distance between the two LEDs).
- `CameraIndex` ... Index of the camera in the cluster. The index is used to properly read the LED data files.
- `path` ... Path to the files. If set to '`x`', the current directory is used.

5.7 Data file `led_cXX_Y.txt`

Example of LED results file `led_cXX_Y.txt` generated by `CalibLED.exe`:

```
1 -1 -1 -1 -1
2 -1 -1 -1 -1
3 -1 -1 -1 -1
4 -1 -1 -1 -1
5 -1 -1 -1 -1
6 261.512268 318.950989 261.527618 165.865646
7 322.571411 315.714325 318.560333 162.680969
8 386.540131 312.483276 378.603729 161.804764
9 439.500061 310.500031 431.500000 164.500015
10 476.600861 309.599274 469.499939 168.499985
11 497.571411 307.714264 484.571381 172.714294
12 491.878052 303.429077 482.672363 173.536667
13 469.541107 300.835480 471.554810 173.781052
14 433.499969 302.500031 445.557434 177.770279
15 -1 -1 -1 -1
16 -1 -1 -1 -1
...
```

5.8 Configuration file `colors.txt`

Color calibration file `colors.txt` should be defined as follows:

```
9 x 6 colored checkerboard, white odd squares, double border
22 x 16 board white to white, 2x2 squares
    x    y  original colors            measured values
    5    7  (  255      0      0)       (199 0 33)
    5    5  (    0    255      0)       (0 125 39)
    5    3  (    0      0    255)       (61 26 92)
    5    1  (   42     60    153)       (64 65 98)
    4    8  (   91     61    105)       (89 70 83)
    4    6  (   88    108     65)       (87 99 73)
    4    4  (  184     42     56)       (141 56 58)
    4    2  (   66    149     71)       (52 108 73)
    4    0  (    0    134    168)       (6 103 136)
    3    7  (   71     92    164)       (75 83 109)
    3    5  (   91    123    155)       (78 100 110)
    3    3  (  199     84     97)       (166 88 78)
    3    1  (  220    124     37)       (201 116 50)
    2    8  (  160    188     63)       (132 150 72)
    2    6  (  192     81    146)       (160 86 100)
    2    4  (  132    127    176)       (109 108 124)
    2    2  (  232    165     42)       (223 148 56)
    2    0  (   99    189    172)       (65 139 135)
    1    7  (  239    200     28)       (237 174 35)
    1    5  (   96     96     96)       (90 91 83)
    1    3  (  144    144    144)       (116 119 109)
    1    1  (  192    192    192)       (170 169 161)
    0    8  (  225    174    137)       (211 164 128)
    0    6  (  212    159    122)       (194 149 112)
    0    4  (  193    144    115)       (169 131 99)
    0    2  (  182    133    100)       (155 120 89)
    0    0  (  162    117     90)       (133 105 79)
```

5.9 Collecting Checkerboard Images

Here is a quick guide for collection of checkerboard images. To get proper calibration, make sure the following:

(a) Checkerboard:

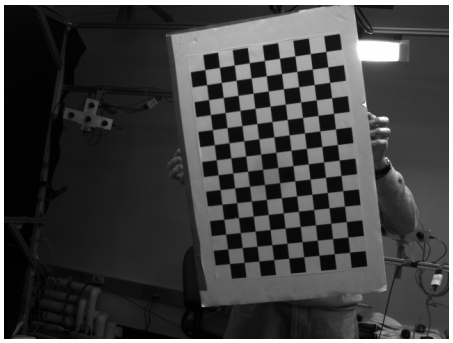
- Checkerboard must have (odd $\times$ even) or (even $\times$ odd) number of squares.
- Preferrably orient the checkerboard to have the dark corner square in the top-left side.
- Use sufficiently large checkerboard ($\geq$ 40mm). More squares will result in more points and better calibration.
- Make sure to capture sufficient number of different positions and orientations (about 20 images works well).
- When positioning your checkerboard, make sure the image plane is covered equally.
- First run the calibration with larger error threshold, then reduce depending on your errors.
- Typical reprojection errors can be about 0.1-0.2 for vga image size (640 $\times$ 480). The error size depends on the image resolution.
- Use Matlab functions to check the distribution of error since some maximal errors may be large but sparsely represented.

(b) Cameras:

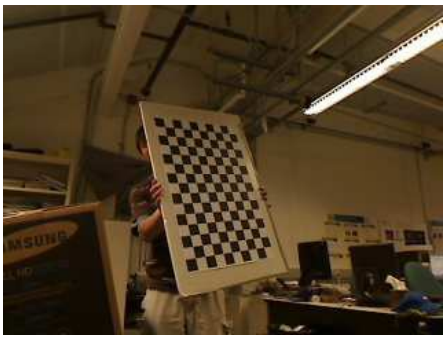
- Use full resolution of the camera to collect the images.
- With lower resolution images (e.g., 320x240) use larger size squares in the checkerboard.
- Make sure the images on the cameras are not too dark or saturated into the white space. White checkerboard squares in captured images should have less than [255 255 255] RGB values.
- Lower the gain parameter on the camera as it will increase the noise (and consequently increase errors).
- Check the focus on the camera with the current camera setting to get reliable images.
- Large shutter setting may result in blurred images if you plan to hold the checkerboard while collecting data. Check camera aperture (if equipped) to control the birghtness of the images.
- When capturing data for several cameras, all data has to be collected at the same time.

(c) Color:

- For the color calibration, set the white blance values close to what appears as natural colors.

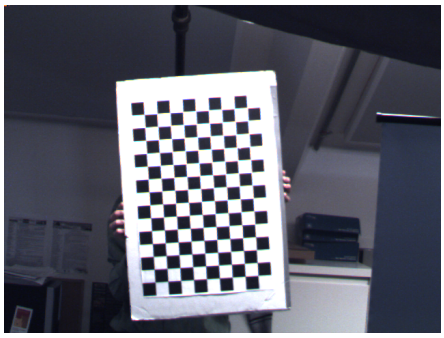


OK Image. Good contrast. White is not washed out.

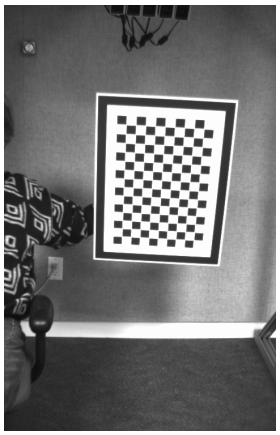


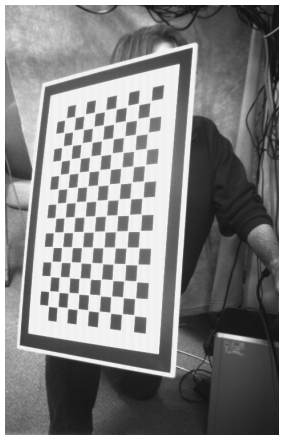
Bad Image. Resolution too low for the size of squares and distance.

Fair Image. Resolution too low. High compression.



Bad Image. Uneven illumination. Camera saturated.





Bad Image. Camera saturated. Gain too high. Distance too large. Large angle.

Figure 6: Sample images of the checkerboard.