CALIBRATION TUTORIAL, SEPT 30TH, 2007

*Performing a multiplane calibration on the dataset '07.07.09ptvsample'*

The sample dataset is calibrated on a planar target imaged at three different heights; the procedure consists in the acquisition of the image coordinates of the target blobs at each level (referred as planes a, b, c) and the following composition of these data in a virtual 3d-target, on which the calibration procedure is performed. The working folder for the calibration is the subfolder 'cal', other paths indicated in the tutorial are children of 'cal'.

## Processing each single plane

Browse into 'cal' and edit the content of the included batch file with the absolute paths of the ptv executable and the *ptv.tcl* file as installed on your machine. Launch the batch file.
In the gui, choose *Change Parameters* $\rightharpoonup$ *Calibration Parameters* and modify the fields as shown in figure 1.

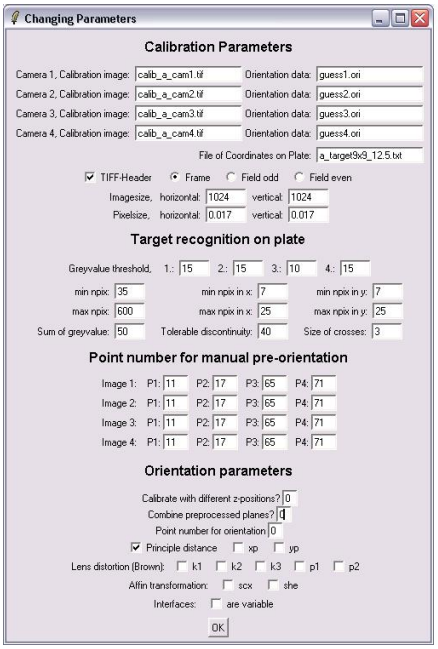


Figure 1: Parameters for plane a.

In 'cal', delete the file *man_ori.dat* and copy *man_ori_a.dat* as *man_ori.dat*. From the gui choose *Calibration* $\rightharpoonup$ *Show Calib. image* and $\rightharpoonup$ *Detection*, and check that all the dots listed in the target file (the central 9x9 dots for plane a) are properly detected and marked with blue crosses in each image. Proceed with $\rightharpoonup$ *Orientation with file*, which uses the previously created *man_ori.dat* file with the coordinates of four manually clicked preorientation points.
Four *guess.ori* files with camera parameters are needed for preorientation. They consist in the position and orientation of the pinhole of each camera and the respective glass window (the refractive interface between air and glass) in the reference frame assigned through the coordinates of the calibration dots in the *taget.txt* files; the principle distance and the offset of the lens axis relative to the sensor centre. Precisely, the file is structured as:

| | | | |
|---|---|---|---|
| $x_{cam}$ | $y_{cam}$ | $z_{cam}$ | pinhole position |
| $\alpha_{cam}$ | $\beta_{cam}$ | $\gamma_{cam}$ | cam rotation vector |
| | $R_{ij}$ | | cam rotation matrix (not used in the guess files) |
| $x_P$ | $y_P$ | | lens offset |
| $f$ | | | principle distance |
| $x_{glass}$ | $y_{glass}$ | $z_{glass}$ | interface position |

Once the four *guess.ori* files are manually edited, it is possible to visualise the reprojections of the grid dots on the images in order to check the quality of the guesses using the command $\rightharpoonup$ *Show initial guess*: the yellow crosses should follow inside the images, if not their virtual image coordinates are anyway displayed in the terminal window.
Now the preorientation can be executed through the command $\rightharpoonup$ *Sortgrid*; check carefully that all the dots listed in the target file (9x9 for plane a) are properly assigned and marked with white crosses and point numbers, as in figure 2.

Finally, $\rightharpoonup$ *Orientation* does the real calibration: the results are printed in the terminal window and in new *.ori* files. If the algorithm does converge and the computed camera parameters are satisfactory (reasonably close to the measured positions used for the *guess.ori* files), the single plane calibration is checked. The image windows display the fitting error (enhanced for visibility) for each dot, and the gui reports the rms error for all the dots, see

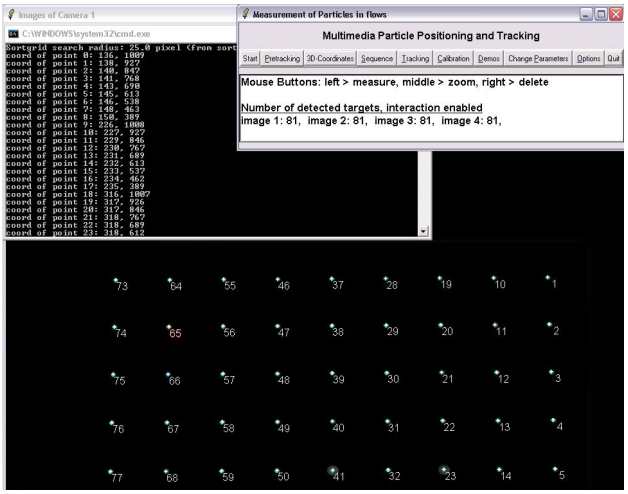


Figure 2: Preorientation results for plane a.

figure 3.

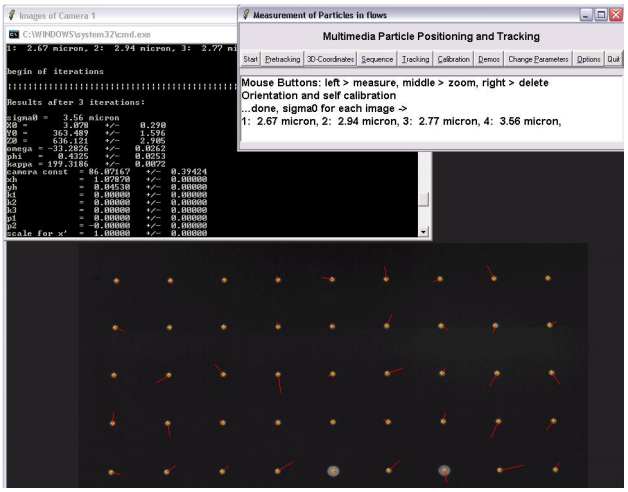


Figure 3: Orientation results for plane a.

Now some files with image and world coordinates of the dots have to be written, in order to perform the following multiplane combination. In $\rightharpoonup$ *Calibration Parameters*, set to 1 the value of $Calibrate\ with\ different\ z-positions$ and execute once more $\rightharpoonup$ *Orientation*. Check that *.crd*, *.pix* and $\_fix$ files are created for each camera.

Now repeat all the previous steps for the remaining calibration planes (b

and c in the example), paying attention to change the appropriate calibration parameters and the *man_ori.dat* file; note that the points for manual preorientation are listed in the corresponding *man_ori.dat* filenames.

## Combining the planes

The last step of the procedure consists in performing the calibration on the virtual 3d-target.
In ⇀ *Calibration Parameters* change the appropriate fields for the plane m (as multiplane), and set both *Calibrate with different z − positions* and *Combine preprocessed planes* to 1, as shown in figure 4.

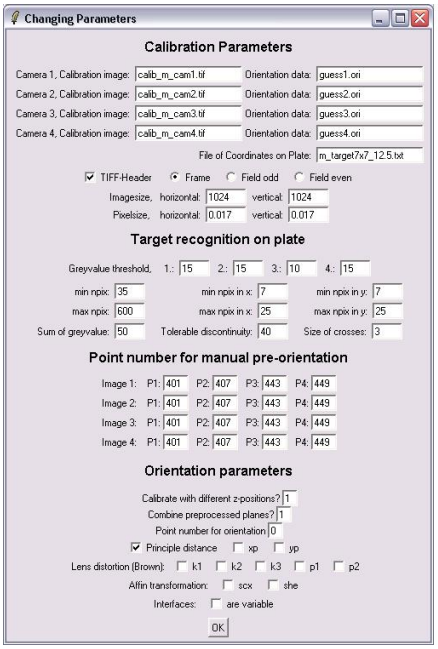


Figure 4: Parameters for plane m (multiplane combination).

Execute once more *Calibration* ⇀ *Show Calib. image*, and black images are shown. Finally ⇀ *Orientation* performs the calibration on the virtual 3d-target. As before, the results are shown in the terminal window and written to files: check that position and orientation of cameras and glass interfaces are consistent.
Now it's time to improve the accuracy, adding one by one some distortion parameters, in order to take into account for lens offset and aberrations: go to ⇀ *Calibration Parameters*, add one parameter, confirm with *ok*, run ⇀ *Orientation*, proceed. Beside looking at the rms error in output, check

constantly how the output *.ori* files are modified: often the procedure tries to correct for wrong position and orientation with the use of $k$ and $p$ aberrations parameters, leading to 'fake' results or not converging at all; it is thus essential that the procedure converges for all the cameras before introducing any lens distortion (as in figure 4).

## *Further suggestions*

The software needs to be restarted before a new set of images is opened (e.g. for each plane) and whenever the orientation procedure does not converge or leads to inconsistent (wrong or NaNs) orientation values. In the latter case, it is always advisable to manually delete the latest *.ori* files and the *raw.ori* ones before launching the batch file.
While optimising the calibration using the distortion parameters, it is convenient to backup the latest good *.ori* and *.addpar* files at each step. The content of consistent *.ori* files can be manually copied into the *guess.ori* files, in order to use it as a starting point for the following steps.
Use the commands $\rightharpoonup Checkpoints$ and $\rightharpoonup AP-figures$ for further checking the calibration results, especially while testing the influence of lens distortion parameters.
In order to fully check the ability of the algorithm in 3d-positioning with the achieved calibration data, it is advisable to set the calibration image files as flow images (*Main Parameters* $\rightharpoonup$ *Name of image* and $\rightharpoonup$ *Basename for sequence*), restart the software and load the images with the *Start* command. Follow the three steps in the menu *Pretracking* and finally press $3D-coordinates$ to write an output file with the recovered 'world' positions of the calibration dots. A shortcut to this is to perform a clean sequence of the commands: *Show Calib. image*, *Detection*, *Orientation with file*, *Sortgrid*, *Orientation*; now it is possible to use the calibration images already loaded following the menu *Pretracking* and then $3D-coordinates$, without changing any parameter. In both cases the code writes in the folder *res* the file *dt_lsq* with the recovered target coordinates in the first three columns, and displays in the command window the actual rms error for 3d-positioning.
Two simple Matlab scripts are included in the sample working folder: *displayOri.m* can be used to display the camera arrangement around the observation volume, useful when dealing with guess files and the first raw calibrations; *displayRt_is.m* plots the recovered positions against the ones assigned in the target files.