AcquisitionAndIndependentCalibration.py

List of Utility Functions

- getch()
- parse_cameras_port_txt(imf)
- capture_images_from_cameras(nFrames)
- square(x)
- validate(intrinsic_matrix,
  distortion_coeffs,
  imagePoints,
  objectPoints,
  pointCounts,
  numFrames,
  board_n
  nbFrames)

Documentation of Utility Functions

Function Name:	getch()
Arguments:	NONE
Return Value:	Returns the ASCII value of the key pressed

Description:

The function getch() works the same way as the function getch() in C++. It waits for a key to be pressed on the keyboard and returns its ASCII value. In the code it's function is to wait for a key press to capture the left and right camera images, if ESC key is pressed then the image acquisition loop is terminated, it's provided as a way out if the code or cameras are stuck for some reason and need to be re-initialized.

Function Name:	parse_camera_port_txt()
Arguments:	imf
Return Value:	Returns the camera port

Description:

The function parse_camera_port_txt() takes as argument the port file where the information that to which port the camera is connected is stored, and extracts and returns the port number from it. It is very important that the port number is given when capturing using the connected camera by using the gphoto2 capture command. It is also important to give the port number while writing the captured file to the disk so that it knows which camera it is talking about.

Function Name:	capture_images_from_cameras()
Arguments:	nFrames
Return Value:	Returns True

Description:

The function capture_images_from_cameras() takes as an argument number of test samples or frames that the cameras will be capturing and returns True once the capture process is finished. In the code the function waits for the user to press any key on the keyboard but ESC to capture the image of the checkerboard and writes that image to the disk. The image is stored depending on the data path given. The images names used for storage purposes are IMG_0000.JPG, IMG_0001.JPG, … and so on. Once the first image is captured by both cameras, the user is prompted again to press any key but ESC to capture and this continues until the user has the required number of frames.

Function Name:	square()
Arguments:	x
Return Value:	Returns the square of x

Description:

The function square() is a simple function that takes as argument the number whose square value will be returned. This function is used in the calibration validation process.

Function Name:	validate()
Arguments:	intrinsic_matrix, distortion_coeffs, imagePoints, objectPoints, pointCounts, numFrames, board_n, nbFrames
Return Value:	Returns the RMS Error in pixels that reflects the calibration accuracy

Description:

The function validate() takes as arguments the calibration data obtained by the calibration process, the total number of samples used for calibration, the board size, and the number of frames or views used for calibration out of the number of frames taken or captured by the capture process. It calculates the difference between the image points in the frame obtained by calibration and the expected image points obtained by projection. The sum of the error is obtained for once frame and then that sum is accumulated over all the sample frames to obtain a Total RMS value that reflects the calibration accuracy in terms of pixels, so a lower value will indicate better calibration.

Main Independent Camera Calibration Procedure

Description:

The calibration procedure starts by obtaining the samples by capturing images from the connected cameras. It is assumed that the connected cameras are matching and are properly aligned so that they more or less have the same view. Once the samples are obtained the calibration process starts bu reading the samples from the disk. For the loaded image file for a left/right camera corner detection process is applied, once the corners are obtained the image is displayed with corners drawn on them for visualization purposes. If the board is good that is maximum corners are found then that view is selected for calibration and the corners obtained for that view is selected as the legitimate data. Then a call to the calibration function (provided by the opencv2.0 open source image processing library) is made to obtain the calibration data for the camera such as intrinsic matrix and distortion coefficients. This data is then saved to the dedicated space and the Total RMS value is given as well to know the calibration accuracy, and then we move to the next camera and the same process is repeated. The calibration process is finished when the message “Independent Camera Calibration Done!” is displayed on the console.

Note: The Total RMS value is just an indicator as to how accurate the calibration has been done. A lower value indicates better calibration as compared to a higher value. The method to obtain the RMS value differs depending on whether we are validating independent camera calibration or stereo calibration.