Real-Time Radius of Gyration Tracker

Original version: August 2011 by Donald J Woodbury, Anselm Hui

Revisions: 4 January 2012 by David Bailey

Windows-only Python software that uses a webcam to measure and plot the radius of gyration of a beaded chain in real time.

"RealTime_ROG" Folder Information

Along with the main RealTimeROG.py file containing the fundamental structure of the program, this folder also includes a supporting RTRogTools.py file and a Video Capture module. VideoCapture (http://videocapture.sourceforge.net/) is an open source, python powered, Windows-only webcam interface that may be downloaded and installed on your machine. Included here is a slightly modified version designed to work specifically with the gyration tracker program; if future edits are made that make use of additional VideoCapture functionality, the original module should be installed to replace the one packaged in this release. Since VideoCapture is a Windows-only module, this distribution will not work on any other operating system. An OpenCV (http://opencv.willowgarage.com) version that will work on Windows, Mac, or Linux has not been completed.

The Algorithm

Radius of gyration is a scalar quantity that characterizes of the locations of the beads in the chain with respect to its centre of mass, defined as:

$$R = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\vec{r}_i - \vec{r}_{CM}\right)^2}$$

where $\vec{r}_i$ is the distance to each bead and $\vec{r}_{CM}$ is the center of mass of the bead.

The primary function of this program is detecting the locations of the beads on the plate. To do this, all pixels above a given brightness threshold are detected and it is assumed that they exclusively belong to beads in the chain. Originally, these pixels were then grouped into individual beads and then the ROG of those beads was found, but due to speed requirements, this step was dropped. Now, every pixel is assumed to be a bead and the radius of gyration of those beads is then found.

Running the Code

This program is designed to work exclusively through a Tkinter Graphical User Interface (GUI). When RealTimeROG.py is run, a Tkinter (http://wiki.python.org/moin/TkInter) window is launched displaying a still image taken from the webcam.

Note: This program requires the Python Imaging Library (PIL). If your computer does not have the PIL module installed, it will return *"ImportError: No module named*

ImageDraw. In this case, go to the *"Python Imaging Library"* website (http://www.pythonware.com/products/pil/) and install the PIL version corresponding to the Python version shown on the first line when the Python shell is launched. Re-start the program and run it again.

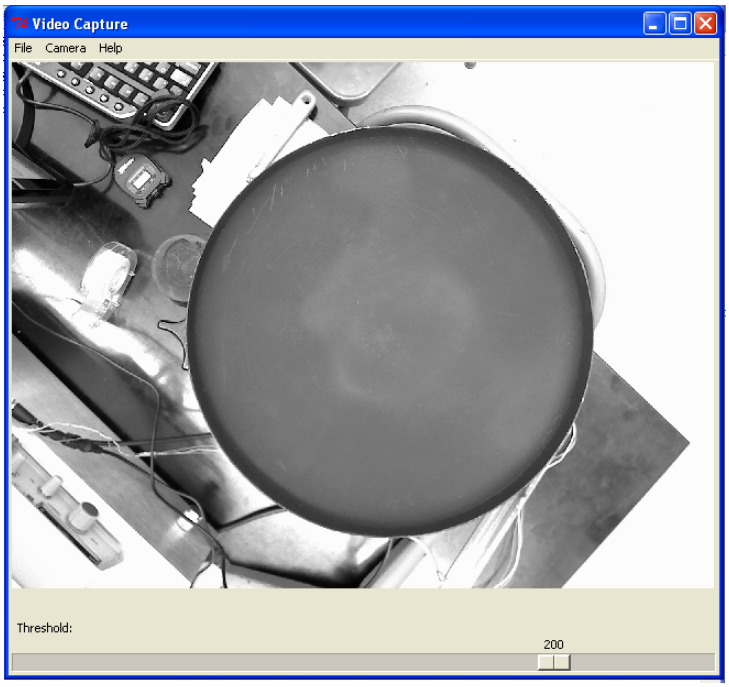

Fig.1 RealTimeROG main window

To begin measuring the ROG, you must first start the webcam by selecting **Start Camera** from the **Camera** menu. The main window should now return live images. Once this is done, you need to ensure that all parameters are correctly set before beginning the ROG tracking.

First, use the **Threshold** slider at the bottom of the Video Capture window to ensure that the threshold is set to an appropriate level to only highlight the beads. The efficiency of this program is dependant on the number of pixels detected, so if the threshold is incorrectly set, the program may run very slowly or crash. The default threshold value is 200, which should work well under most conditions. The user should fine adjust the threshold such that ***only the beads are detected.*** For lower thresholds, the rod connecting the beads may also be detected, which is not ideal but not really a problem, since the rods follow the beads and so have essentially the same radius of gyration. Small shiny particles on the plate surface may need to be removed, but the most likely problem is from glare from overhead lights. If the lights cannot be turned off, a piece of cardboard, cloth, or umbrella overhead should solve the problem.

Next, adjust the height of the camera such that the whole plate is visible in the image. Then, you must define the searchable area of the image by launching the **Calibrate** function under the **Camera** menu. This will prompt the user to select the four edges of the plate, and to specify the diameter of the plate.

Note: Because the webcam is relatively close to the plate, there is significant parallax between the top and bottom of the plate walls. When calibrating, select the edges of the

bottom of the plate, instead of the edge of the outer boundary. Incorrect calibration will cause two major problems. The chain lies on the bottom of the plate, so selecting the outer boundary will underestimate the diameter of the plate. (If desired, you can correct by hand the small parallax effect due to the the bead centres being a bead-radius above the plate.) A less obvious problem is that when the plate vibrates downwards, large blank areas will be included by the program, resulting in larger fluctuations in the readings.

A correct calibration is illustrated by the picture below.

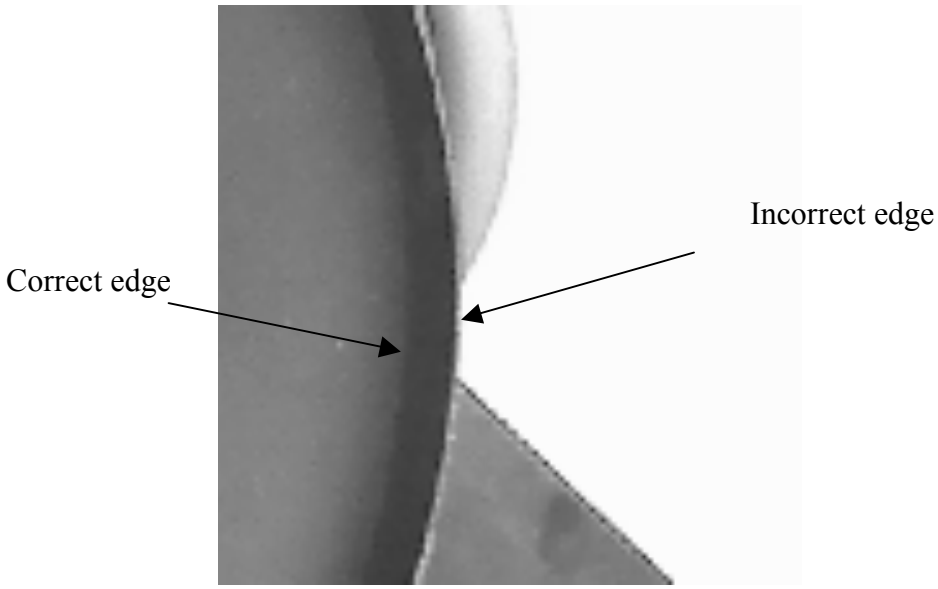

Fig.2 Calibration configuration

The program will then only search within the confines of this area, and will return all ROG data in the same units that the diameter of the plate was entered in. If no calibration is provided, the entire image will be searched for bright pixels and the ROG will be returned in units of pixels.

Once all initialization has taken place, the program is ready to start tracking the ROG. To do this, select the File -> Find Chain. All pixels whose value is greater than the threshold will then be highlighted in red while in the background the ROG of these points is computed. The user may then choose to save this data to a file or plot it in real time. ‘

Here is the description of menu functions that users may use for ROG measurements:

File -> Start Plotting: The function returns a real time plot of the radius of gyration versus time. This function will not store any of the plotted data and should mainly be used by the user to monitor the quality of the data.

File -> Save ROG to file: The function prompts the user to choose a directory and saves the ROG data in a text file. The data saved into the text file is also printed live in the Python shell. The first column corresponds to time and the second column corresponds to ROG. **Save ROG to file** should be the major function the user will need for data analysis.

File -> Take Snapshot: The function saves a snapshot of the currently displayed image under the directory <Directory of Real-Time ROG:\ROG_Snapshots>. The image and

will be named as "ROG_Snapshot_*yyyymmdd_hhmmss_t*", where *yyyymmdd_hhmmss* corresponds to the conventional date and time the snapshot was taken and *t* corresponds to the time elapsed relative to the start of the initialization of **Save ROG to file** or **Start Plotting.** A message will be printed in the shell if a snapshot is taken.

File -> Record Images: The function prompts the user to input a time interval at which images will be taken regularly. Note that there is a few seconds delay between each image. The saving directory, naming methods are identical to **Take Snapshot**. Since ROG measurements are taken at scales of minutes, the time interval should not be too small or else a large amount of snapshots will be generated.

File -> Plot from File: The function can import text files created by **Save ROG to file** and will plot the data using Pylab, which provides a much more finished plot with titles and labels than does the vPython version.

Additional Notes:

Here are some additional notes to some other functions under **Camera**.

Camera -> Webcam Controls: The function launches a setup window that allows the user to change properties of the webcam. Users may wish to unselect **Auto-Focus** after the camera has properly focused as the camera occasionally out-focuses and may yield inaccurate results. If, by any chance, the lighting of the advanced labs are significantly altered and the program does not detect the beads properly under any threshold, the user may wish to go to "Advanced Settings" and alter the exposure, gain , contrast etc. manually to obtain a image with sharp contrast. The user will need to turn off "RightLight" and "Auto" in order to do so. In general, reducing the exposure, gain and brightness and increasing the contrast should yield images with strong contrast.

Camera -> Adjust Resolution: The function launches a setup window that allows the user to change the resolution of the picture. Higher resolutions decrease the speed of the program. The default resolution is set at maximum, 1600 x 1200, and the speed of the program should be sufficiently fast for any purposes. Changing the "Color Space/ Compression" setting is not recommended.