QEndoscopy User Documentation

System Requirements

The provided build is designed for Windows 10 & 11, and requires an Nvidia GPU. The GPU must support CUDA ~~11.6~~2.9 or compute capability ~~3.5~~5.0, which is most nVidia GPUs released since the Maxwell microarchitecture ~~was released~~ in 2014. Computers with AMD graphics cards may still take advantage of hardware acceleration by configuring their Python environment to use PyTorch with RocM. If neither GPU type is available, the AI will fall back to running entirely on the CPU, which will be fully functional, but slow.

The program is designed on Windows with cross platform libraries, so it can also run on Linux, and Mac. However, the Mac version does not support hardware acceleration, which is a limitation of the PyTorch library, which the AI system relies on.

While running, the program utilizes about 2.5GB of RAM.

Installation

~~(todo)~~A build is available on Github that contains a self-contained executable, built with Nuitka. Download and uncompress the 7Zip file, and then execute the "QEndoscopy.exe" inside it.

Alternatively, the program can be run in Python 3.12 using the libraries listed in "requirements.txt." Be advised that PyTorch should be installed from the instructions on its own website, as the version from pip's repository does not include GPU acceleration.

Workflow overview:

This section will walk you through the intended usage of the program. The list below covers the general procedure:

1. Open a Video.
2. Configure ~~s~~Settings.
3. ~~Seek~~ Find ~~to~~ a target video frame.
4. ~~Select a region~~Adjust the slider controls to select a cross section.
5. Record the selection to commit the measurement to a log
6. Repeat 3-5 as needed
7. Review & save measurements

Step by Step

When the program opens and no video is loaded, most features will be disabled. Start by pressing the "Open Video~~.~~" button in the top left corner. A file dialog will appear, which you can use to select a video file. The program is designed for .mp4 files using the H.264 video codec, but the dialog does not filter for any file types. Other video formats may work, and the program supports any video codec supported by OpenCV. Still images also work, although some functionality may be limited.

An alternative way to open a video is to drag a video file onto the executable option, or launch the program with a video path as a command line argument.

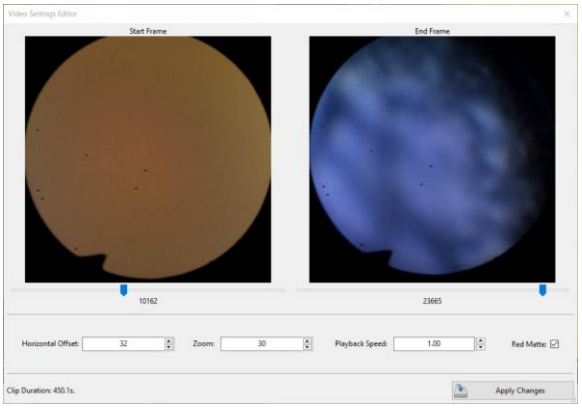
Style Definition: Heading 2: Space Before: 12 pt

Formatted: Heading 3

Video Settings

Understanding the Video Settings panel is essential to producing accurate, usable depth estimations. At first glance, the settings appear to be cosmetic playback options, but a few simple tweaks to each video will greatly increase the quality of results from this program. Therefore, the video settings should be reviewed for every video, and this section of documentation is the most important section to understand.

First make sure a video is opened, because the settings panel will be locked otherwise. All the options listed below are non-destructive parameters that only affect the playback of the video and the analysis parameters, and they do not change or edit the original file.



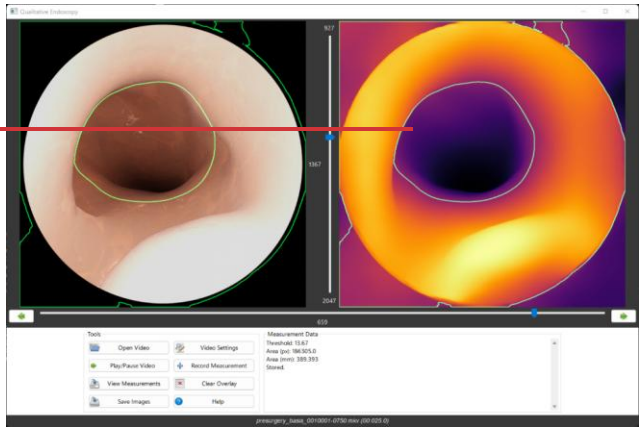
Trim Start/End: If the length of the video is very long or contains sections you do not need to analyze, you can use the trim start and end sliders to isolate a specific section of time that you want to focus on. This enables greater precision and convenience when using the time slider on the main interface. The unit on these sliders is in frames.

Horizontal Offset: The input video is expected to be recorded in a widescreen shape with a circle in the center. Our program always crops the image into a square from the center of the frame, but in many cases, the crop leaves the image circle slightly off center, and this asymmetry leads to reduced accuracy in the depth estimation. To correct this, use this option to slide the region of interest left or right. A correctly aligned image has the same amount of space on both the left and right sides of the image circle.

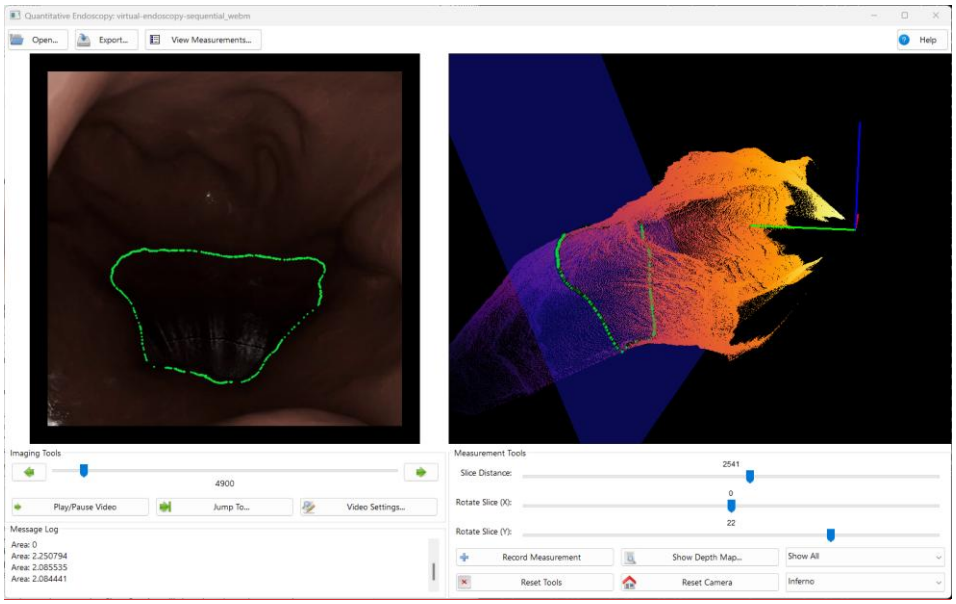
Zoom: Once the image circle is centered in the frame, you can use the zoom setting to maximize the usable area. An ideal setting has the image circle touching all four corners of the frame. Despite the name, this function does not scale the image or simulate a change focal length; it is just reducing the size of the region of interest. The view in the interface appears zoomed because the interface scales to fit its frame, but depth analysis always operates on an unscaled image to avoid scaling artifacts.

Playback Speed: This setting affects the rate of playback when the user presses the "Play/Pause" button in the main interface. This is for the user's convenience and does not affect depth analysis. A setting of 1.0 means the playback matches the original rate, typically 30 frames per second, and 0.5 is half speed.

Red Matte: This option is hard to visualize, because its effect is not shown directly in the depth analysis view. It fills in the black area outside of the image circle with a predefined dark red that is similar to the average color of the source video. In our testing, we have found that this increases the accuracy of depth estimation near the edge of the image, and reduces the amount of artifacts throughout the frame. It is turned by default, but this option exists because it can reduce analysis quality when high zoom levels (>100) are used.



Main Interface



The two images shown in the main interface are different versions of the same frame from the video. The image on the left is the source input image, and the image on the right is the generated point cloud~~depth image~~. The depth image uses a "inferno" colormap to make changes in values more easily discernable, with bright yellow as the closest, followed by orange, then purple, then black as the most distant. This gradient is normalized for each frame in order to provide the most visibility; the absolute values can be seen on the threshold slider, and the measurement data panels.

The horizontal slider ~~below the two images~~below the left image is the **Time Slider.** The left side of the frame is the start frame, and the right edge is the end frame. It can be dragged left or right, and right is

always moving forward in time. The image on the right will not update until the time slider stops changing. The left and right arrows near the slider move the viewer backwards or forwards by one frame. Changing the frame number will ~~clear the overlay.~~recompute the point cloud and the cross-section overlay. The **Jump To** button allows you to type in a specific frame number.

The ~~vertical slider in the middle is the **Threshold Slider**~~horizontal slider below the point cloud is the **Slice Distance**. Moving this slider creates an area selection, which is shown on screen by the green lines. Its minimum and maximum values show the range of values generated by the depth analysis. ~~High values are closer to the camera~~The number represents the distance in millimeters between the camera and the center of the plane. If the camera's view is not perpendicular to the structure, the **Rotate Slice** sliders can be used to fine tune the position of the slicing plane.

To get a better view of the point cloud, you can use the mouse to move the virtual camera. Hold the left mouse button to rotate the view, and shift + left click to pan the camera. The scroll wheel or right click can be used to zoom the camera. The **Reset Camera** button will bring the camera back to its original position, which is always shown by a 3D axis marker.

The depth overlay annotation is always computed using the depth analysis image, then it is copied back onto the source image. ~~By moving this slider, you can select a cross section that is perpendicular to the camera.~~ Alternatively, you can click on the image to specify a threshold visually.

~~If the annotations are in the way of your view, pressing **Clear Overlay** will clear them from the screen. It does not disrupt anything about the depth analysis, or the data log. Clicking on the threshold slider will bring the overlay back.~~

Pressing the **Record Measurement** button will record the current annotation's data into a data log. A notification appears in the status text area to tell you that the data has been saved.

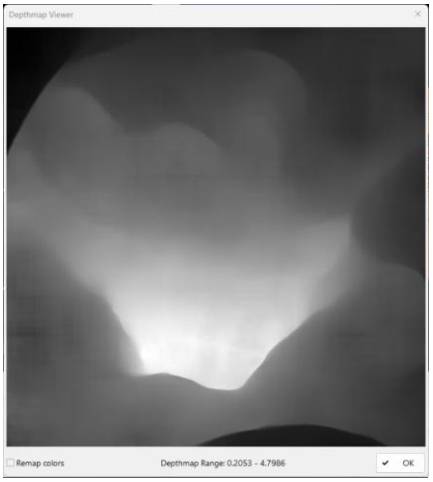
~~The **Save Images** button allows you to export full resolution, high quality images of what you are seeing on screen. It operates in two steps. The first dialog box that appears concerns the image on the left, the input video frame. Specify a location and filename to save this image, or close the window to move onto the next step. The file format is determined automatically by the extension you provide in the file name. The second dialog box concerns the image on the right. If your filename ends in .jpg or .png, the saved file will be the post-processed, colorized version that you see on screen. If your filename ends in .npy, you can export raw numerical data that you can use in another program. NPY is a binary Numpy object, a two dimensional array of 32-bit values for each pixel which can be loaded into a Python script using the numpy.load() function.~~

View Measurement

Pressing "View Measurements" will present a spreadsheet containing all the measurements you have saved so far. The measurements are always sorted by frame number, even if they were sampled out of order. At the bottom of the dialog, you can specify a file location and save the document as an Excel spreadsheet (.xlsx). The Excel version also contains additional rows for metadata.

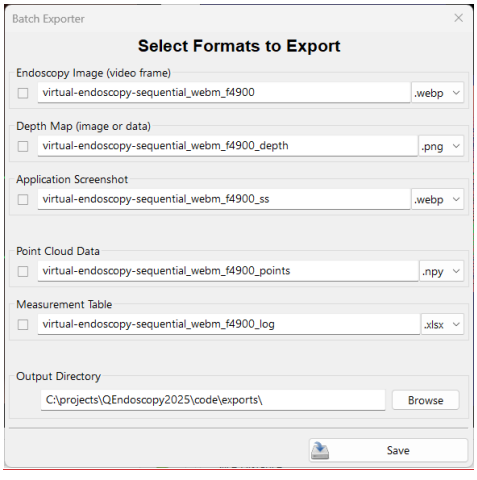
If you need to reset the spreadsheet, re-open the video. The data is cleared when a new video is opened.

Depth Map Viewer



The **Show Depth Map** button creates a subwindow which reveals how the point cloud is generated. This is useful for debugging purposes. **Remap colors** is a convenient toggle switch between a raw data view (white is distant) and a colorized version. Both views are normalized for display, but the actual range is shown in text at the bottom of the window. The color map is selectable in the main dialog. Pressing the OK button closes the window, and does not change any parameters or data.

Export



The **Export** opens a dialog which can export both images and data in a variety of formats in a single step. All types follow the same UI concept. First select the check box on the left if you want that format to be processed. Then review the default file name and adjust it if necessary. Finally, confirm the file format,

Formatted: Normal, Centered

Formatted: Centered

because different formats may have different behaviors. All exported files will be generated in the folder listed at the bottom, in the **Output Directory** field, but other folders can be selected by pressing the **Browse** button. Pressing **Save** will begin the export process. To cancel, press the **X** in the top right corner, or press Save with no check boxes marked.

The depth map and point cloud exporters include the option to export raw data instead of an image. To use this, select a file format of NPY, XYZ, or CSV instead of jpeg, png, or webp.

~~to export full resolution, high quality images of what you are seeing on screen. It operates in two steps. The first dialog box that appears concerns the image on the left, the input video frame. Specify a location and filename to save this image, or close the window to move onto the next step. The file format is determined automatically by the extension you provide in the file name. The second dialog box concerns the image on the right. If your filename ends in .jpg or .png, the saved file will be the post-processed, colorized version that you see on screen. If your filename ends in .npy, you can export raw numerical data that you can use in another program.~~ NPY is a binary Numpy object, a two dimensional array of 32-bit values for each pixel which can be loaded into a Python script using the numpy.load() function. XYZ and CSV are text formats which can be processed by other programs.

Generating Virtual Endoscopy Videos

intro

Blender

Using the plugin

Source Code Documentation

Code Contents

The program is written in Python 3.12~~0~~ using a variety of libraries, such as WxPython, Numpy, OpenCV, and PyTorch. Detailed information about the code's workings are contained within the code comments, and this section only intends to provide a broad overview.

QEndoscopy.py	This file sits in the root code folder, and contains very little code. It is a simple launcher for the rest of the program.
Main.py	This is the main script of the program. It loads in the other modules and layouts and contains the code that routes everything between them. In MVC terminology, this is the Controller.
Main_dialogs.py	Logic code for all popup windows, such as the exporter.
Layouts_v3.fbp	This is the WxFormBuilder project file, which contains the graphic interface and all supporting dialogs.
Layouts.py	This file is automatically generated by WxFormBuilder and should not edited manually. It provides the base classes for main.py's functionality, as well as the graphic interface layout. In MVC terminology, this is the View.
Imaging.py	All functionality for opening video files, reading video file metadata, and other imaging utilities, such as screenshots and annotations.~~the imaging pipeline.~~
Analysis.py	Functionality for the ~~Midas~~ point cloud computation and visualization, using Vispy.~~library, and the image processing that goes a long with that.~~
Depthmap.py	Advanced helper functions for Analysis.py, including point cloud processing and measuring
Logging.py~~Measurement.py~~	~~Functionality for measuring the analysis output, as well as annotating it,~~ contains classes for storing data entries into a database,~~and the logging functionality.~~ and exporting logs.

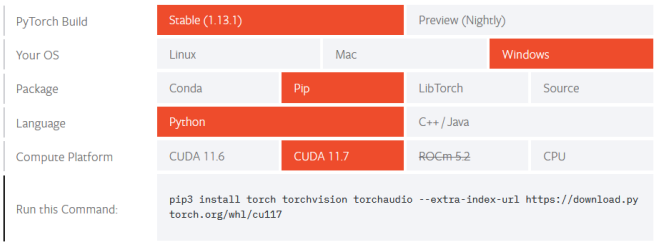
Formatted Table

A tool that may be useful to development is WxFormBuilder. It is not required to run the QEndoscopy program, but it is required to modify the interface file, "modules/layouts.fbp".Version 3.10.1 was the latest release at the time of publication, which can be downloaded here: https://github.com/wxFormBuilder/wxFormBuilder .

Setting Up The Environment

The program's environment is built inside of a virtual environment build using venv, a default Python library. This environment can be activated by running "activate.bat" in qe_venv\Scripts. That distribution is platform-specific, and it may be necessary to rebuild the environment in the future. If that is the case, review "Code\requirements.txt" to find a record of all the libraries used in the project.

Pip can automatically process requirements.txt, with the exception of Torch, which requires an additional step. Torch is the machine learning framework used by this program (via Midas). If it is installed via Pip like the other libraries, Pip will install a version of Torch that does not utilize CUDA hardware acceleration, so it will make analysis significantly slower. To install Torch properly, visit https://pytorch.org/get-started/locally/ to download the latest version. There are several options available, and I recommend the combination shown below:


PyTorch Build
Stable (1.13.1)
Preview (Nightly)
Your OS
Linux
Mac
Windows
Package
Conda
Pip
LibTorch
Source
Language
Python
C++ / Java
Compute Platform
CUDA 11.6
CUDA 11.7
ROCm 5.2
CPU
Run this Command:
pip3 install torch torchvision torchaudio --extra-index-url https://download.pytorch.org/whl/cu117