Qualitative Endoscopy User Documentation

Installation

To install the program, simply unzip the latest “Build” zip file to a writeable location, then run **QEndoscopy.exe** in the generated folder. It does not require administrator privileges to decompress or run. This folder is a compiled, standalone version of the Python code, and does not require Python or any of its libraries to be preinstalled.

Within this folder, take note of the **exports** and **logs** folder. These two folders will accumulate images, data, and log file as the program is used. A shortcut link to the logs folder may be useful. If these folders become cluttered, you can safely delete the contents of these folders without affecting the program.

The Two Versions

There are two versions of the program available: the **demonstration** version, and the **AI** version. The difference between the two is that the demonstration version uses precomputed depth map files, and the AI version computes the depth map on the fly. The version being used for the current phase in the study is the demonstration version.

Both versions are precompiled for Windows 10 & 11, but should also work on older versions. The AI version requires an Nvidia GPU that supports CUDA 11.6 or compute capability 3.5, which is most nVidia GPUs released since the Maxwell microarchitecture was released in 2014. Computers that do not meet this requirement can still use the software by using the Python environment described later in this document.

Workflow overview:

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

1. Open a Video
2. Configure Settings
3. Seek to a target frame
4. Select a region
5. Record the selection
6. Repeat 3-5 as needed
7. Review & save measurements

Open a Video

Start by pressing the "Open Video" button. 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.

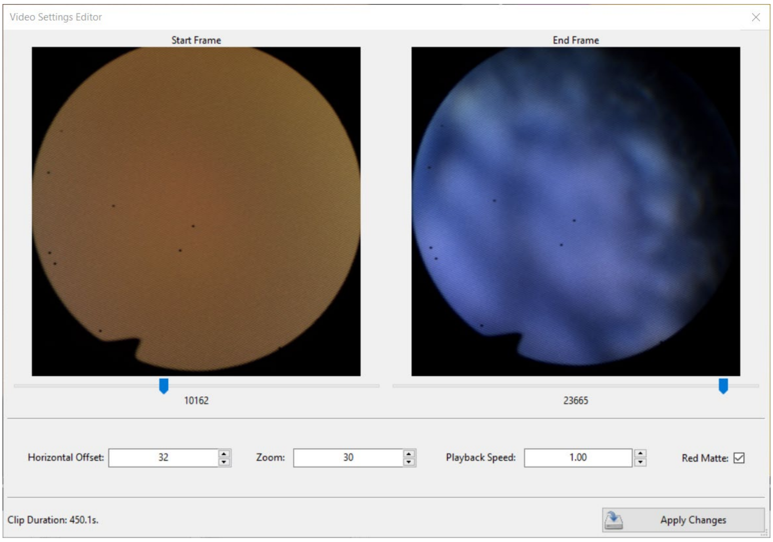
For the demonstration version, an additional depth map video is required. These videos are identified with "_depth" at the end of their file name, and if this file cannot be file, the video will not be opened. You should not open the depth map video directly, and the program will prevent you from doing so.

A CSV file that matches the primary video's filename is also present in the provided video directory, but this file is optional. It is a data file containing information about the camera's spatial position, which it adds to the log file when measurements are made. If the CSV file cannot be found, this step in the log is skipped without error.

Video Settings

In the AI version, 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. In the demonstration version, these functions are optinal.

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.


Video Settings Editor

Start Frame: 10162

End Frame: 23665

Horizontal Offset: 32

Zoom: 30

Playback Speed: 1.00

Red Matte: ✓

Clip Duration: 450.1s.

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.

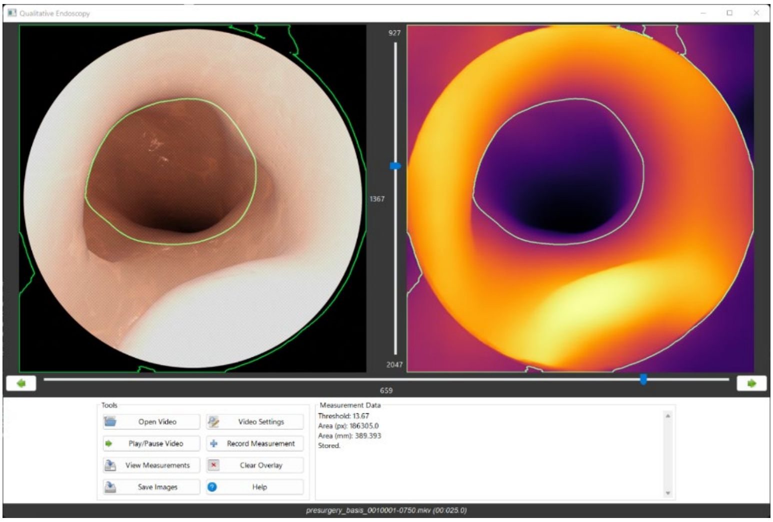
Zoom: Once the image circle is centered in the frame, you can use the zoom setting to maximize the usable area. An ideal setting has the image circle touching all four corners of the frame. Despite the name, this function does not scale the image or simulate a change focal length; it is just reducing 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 depth image. The depth image uses an "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.



Threshold: 13.67
Area (px): 186305.0
Area (mm): 389.393
Stored.

presurgery_basis_0010001-0750.mkv (00:025.0)

The horizontal slider below the two images 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.

The vertical slider in the middle is the **Threshold Slider**. 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 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 spreadsheet. A notification appears in the status text area to tell you that the data has been saved. A screenshot is automatically saved in the logs folder as well.

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.

Review & Save 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), although the default name and location are probably already ideal. 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. Screenshots stored in the log file

Generating Virtual Endoscopy Videos

For this project, we've developed a semi-automated workflow for generating realistic looking virtual endoscopy. The tools used in this are Ansys for mesh creation, and Blender 3.4 for mesh processing and rendering. The source data consists of two files: a GLTF file containing the mesh, and a text file containing the streamline.

We provide a sample Blender project that contains a template environment and a custom plugin that automates most of the steps involved in creating the animation, so swapping in new data can be done easily and consistently. Only basic knowledge of Blender's user interface is required, because as much of the functionality as possible is automated by the script. However, it is possible that the script may need to be adjusted if the data format or desired behavior needs to be changed.

Requirements

The following steps are written for Blender 3.4.1, although future versions should work without issues. A powerful GPU is recommended due to the complexity of the mesh and animation. This project uses Blender's slower and more accurate render engine **Cycles**, due to its support of fisheye lenses and indirect lighting, even though this greatly increases render times. In order to make render times manageable, make sure you have hardware acceleration (such as Cuda or Optix) enabled in Blender's system settings.

Blender

To begin, open the project file, "Virtual Endoscopy Generator.blend" in Blender. There may be a permissions prompt for the plugin as the file loads, which you can permit. The plugin can be triggered through the File menu in the top left. Select *File -> Import -> Import Endoscopy Project*. Then the first of two file selection dialogues will open. On the first one, navigate to your GLTF file, and select it.

Before pressing 'OK,' consider the options provided on the right side of the dialog. All of them are enabled by default, which is probably the correct behavior. **Clear Scene** will erase previous projects loaded into this scene. **Auto-Cleanup** performs several operations to optimize the mesh data's appearance for Blender. These options were designed around our specific data, and they are optional because these functions may not be ideal for your project. Re-run the script with this disabled if you dislike the results. **Import Streamline** triggers the importer to open a second file selection dialog for the next step. Uncheck this option if you do not have a streamline.

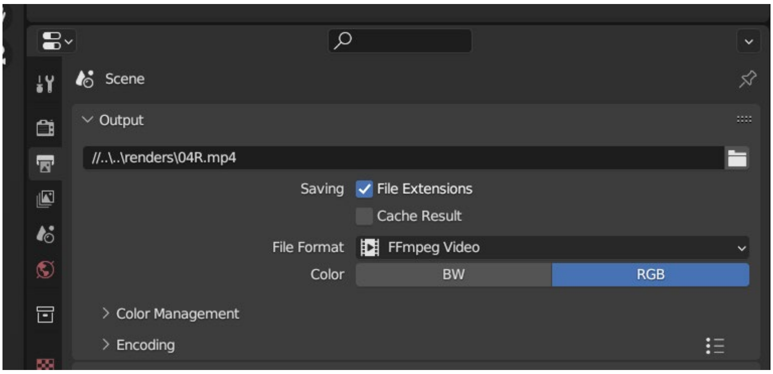
After a moment, a second file selection will appear, requesting the text file containing the streamline matched to your mesh.

There are several options that change the behavior of the importer. **Are Points Contiguous** causes the script to follow the order of vertices in the data, while unchecking it triggers a series of steps that use the 'weld' modifier to bridge nearby vertices into a continuous line. The other option is **Cleanup Needed**, which is turned off by default because it sometimes produces errors with the current data. These options exist because earlier versions of the data required some extra steps, and these may be removed in future versions.

Additional Steps

After the importer is completed, it is recommended to review the generated scene to watch for common errors that may need to be corrected manually. This can include manual edits to the mesh or the streamline. For our animations, we removed the end caps from the mesh and used Blender's proportional edit tool to smoothly redirect the streamline towards the center of the nasopharynx.

If your streamline is excessively dense, it may be useful to filter it using Blender tools such as checker deselect, or merge by distance. Consult the Blender documentation for instructions about these tools.


Scene
Output
//..\..\renders\04R.mp4
Saving ✓ File Extensions
Cache Result
File Format FFmpeg Video
Color BW [✓ RGB]

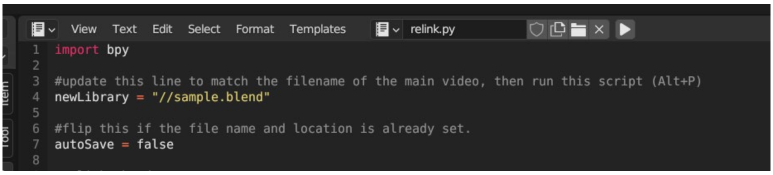
Don't forget to save the project and change the location and file name of the output file. Rendering the movie files at full quality is hardware and time intensive, and may take several hours per clip, depending on graphics card performance.

If your renders appear in a bright magenta, this means the textures could not be loaded. The project references textures in a nearby folder using relative paths, so moving the file may disrupt the references. This can be fixed by using the “Find Missing Files” selection from File -> External Data.

Rendering The Depth Map

Render the depth map involves the same process: Open the template project and run a script. The file name for the template is “Virtual Endoscopy Depthmap.blend.” Like the main project file, the majority of Blender settings are setup in advance. The biggest difference between the two is the compositing node setup, which is reconfigured to output a black and white depth map image.

To use this script, open the depthmap sample and resave it in the same folder as the previous blend file. Then update line 4 in the “relink.py” script to match the file name you specified previously. This script has an **autosave** function which will automatically save the file with the updated information, if desired. Then press the play button in the script window header, or press Alt+P to execute the script.



```
1  import bpy
2
3  #update this line to match the filename of the main video, then run this script (Alt+P)
4  newLibrary = "//sample.blend"
5
6  #flip this if the file name and location is already set.
7  autoSave = false
8
```

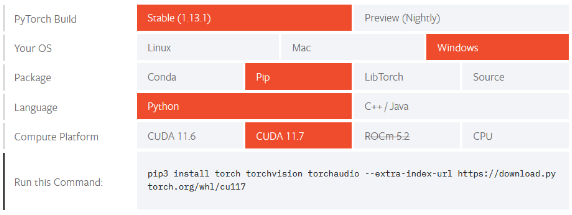
The mesh data and streamline will be relinked, and the data should appear, and the animation is ready to be rendered.

Source Code Documentation

Setting Up The Environment

The program is written in Python 3.10 using a variety of libraries, such as WxPython, Numpy, OpenCV, and PyTorch. 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 completely portable, so 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.

Python's package installer Pip can automatically process requirements.txt to regenerate the virtual environment, except for PyTorch, which requires an additional step. PyTorch is the machine learning framework used by this program, Midas. If PyTorch is installed via Pip like the other libraries, Pip will install a version that does not utilize CUDA hardware acceleration, and analysis will be significantly slower. To install PyTorch 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`			

A tool that may be useful to development is WxFormBuilder. It is not required to run the code, 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 .

Code Contents

Detailed information about the code's workings is 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.
Layouts.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 the imaging pipeline.
Analysis.py	Functionality for the Midas library, and the image processing that goes a long with that.
Measurement.py	Functionality for measuring the analysis output, as well as annotating it, and the logging functionality.
Depthmap.py	Functionality for loading and processing precomputed depthmaps. This bypasses much of analysis.py's functionality in the demonstration version.