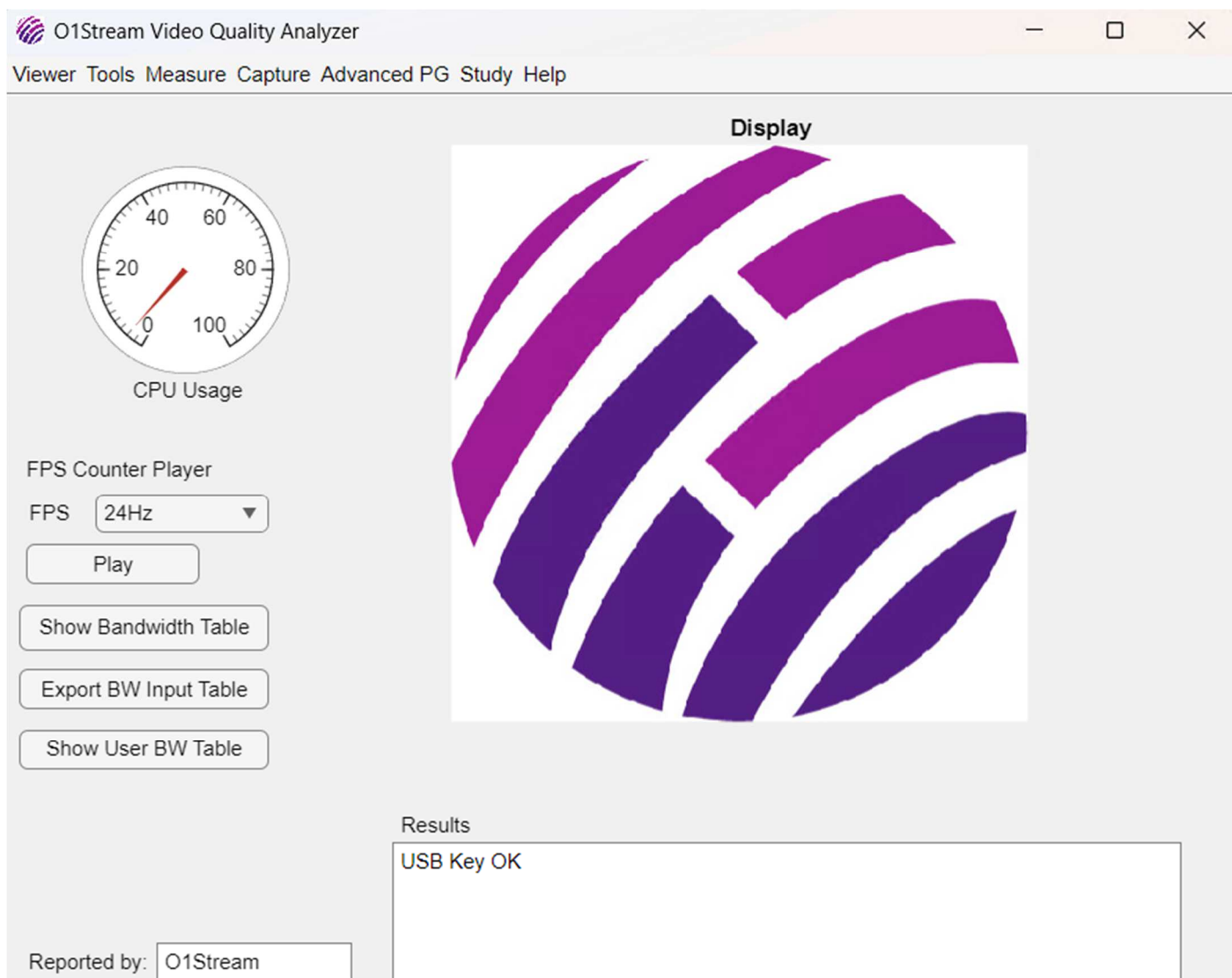


# Video Quality Analyzer

Rev.1.1

Powered by MATLAB  & GoMax Electronics  GoMax



The screenshot shows the O1Stream Video Quality Analyzer application window. The title bar reads "O1Stream Video Quality Analyzer" and includes standard window controls. The menu bar contains "Viewer Tools Measure Capture Advanced PG Study Help". The main interface is divided into several sections:

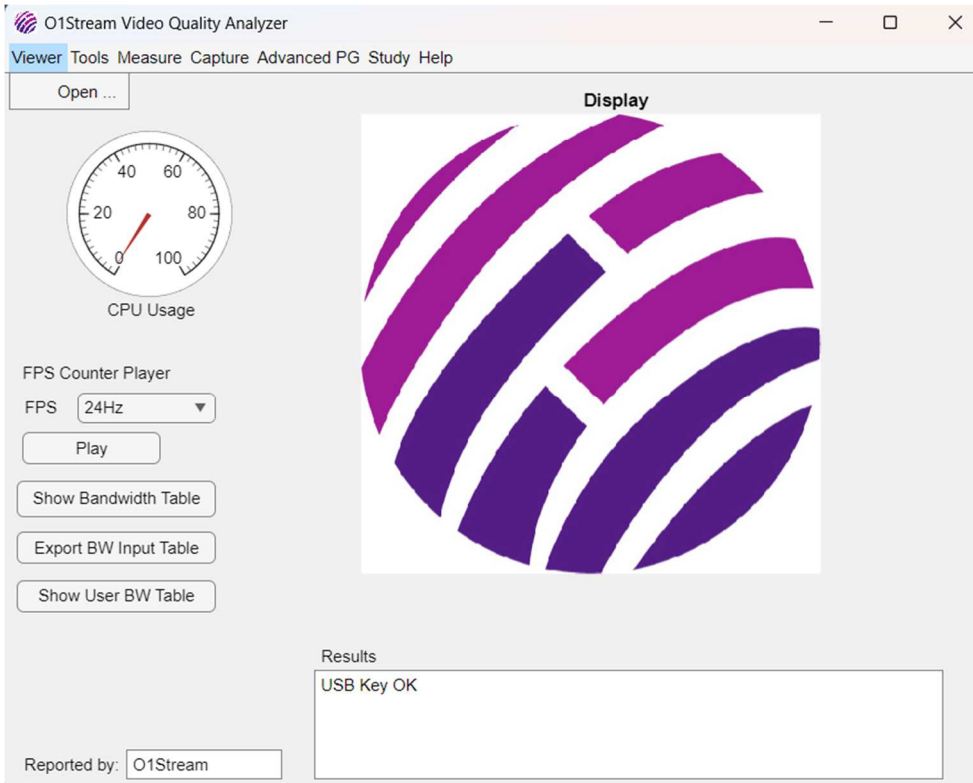
- CPU Usage:** A circular gauge with a scale from 0 to 100. The needle is positioned at approximately 10%.
- FPS Counter Player:** A section with a dropdown menu set to "24Hz", a "Play" button, and three buttons: "Show Bandwidth Table", "Export BW Input Table", and "Show User BW Table".
- Display:** A large central area showing a test pattern of curved, overlapping purple and white stripes.
- Results:** A text box containing the message "USB Key OK".
- Reported by:** A text field containing the name "O1Stream".

**\*\*All features and functions of the O1stream Video Quality Analyzer are subject to change without prior notice.**

## Content

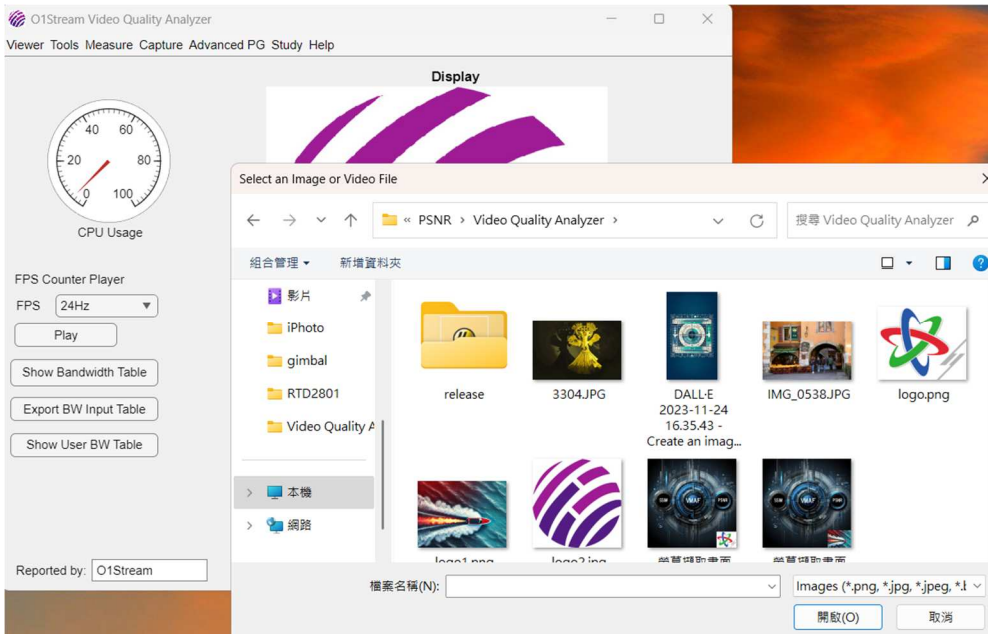
Video Quality Analyzer.....	1
<b>1 Viewer.....</b>	<b>3</b>
<b>1.1 Open.....</b>	<b>3</b>
Open the picture or video file to be reviewed.....	3
<b>2 Tools.....</b>	<b>4</b>
<b>2.1 Picture to Video.....</b>	<b>4</b>
<b>2.1.1 BMP to AVI.....</b>	<b>4</b>
<b>2.1.2 PNG to AVI.....</b>	<b>5</b>
<b>2.1.3 BMP (8bit) to AVI (10bit).....</b>	<b>6</b>
<b>2.2 Video to Picture.....</b>	<b>7</b>
<b>2.2.1 AVI to BMP.....</b>	<b>7</b>
<b>2.2.2 AVI to PNG.....</b>	<b>9</b>
<b>2.3 Generate Frame Counter.....</b>	<b>10</b>
<b>3 Measure.....</b>	<b>12</b>
<b>3.1 VMAF (video).....</b>	<b>12</b>
<b>3.2 PSNR / SSIM (pictures).....</b>	<b>14</b>
<b>3.3 VMAF Comparison.....</b>	<b>16</b>
<b>3.4 PSNR / SSIM Comparison.....</b>	<b>18</b>
<b>4 Capture.....</b>	<b>21</b>
<b>4.1 Preamble.....</b>	<b>21</b>
<b>4.2 Re-Seq.....</b>	<b>22</b>
<b>4.3 Frame drop check.....</b>	<b>23</b>
<b>5 Study.....</b>	<b>25</b>
<b>6 Help.....</b>	<b>32</b>
<b>7 Video Capture Setting Example.....</b>	<b>33</b>
Notes:.....	37

# 1 Viewer



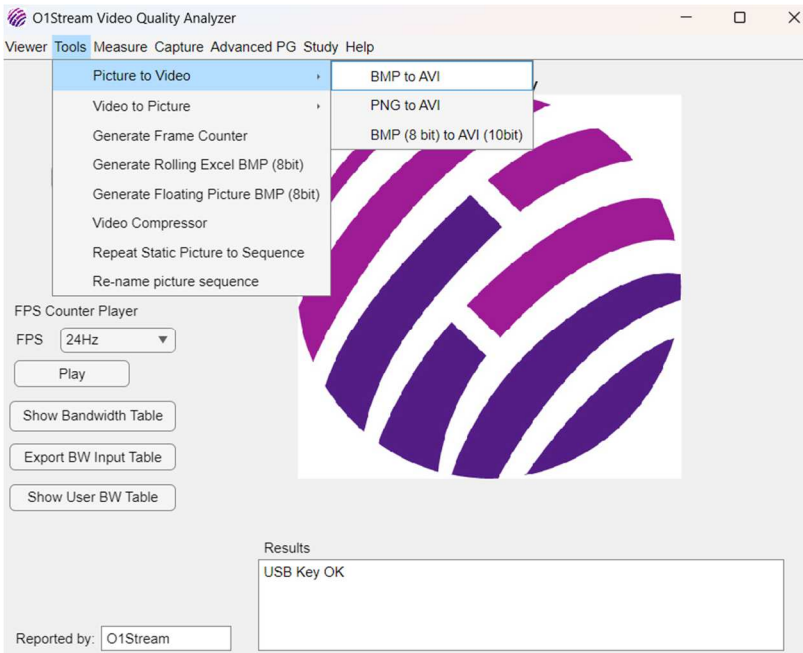
## 1.1 Open

Open the picture or video file to be reviewed.



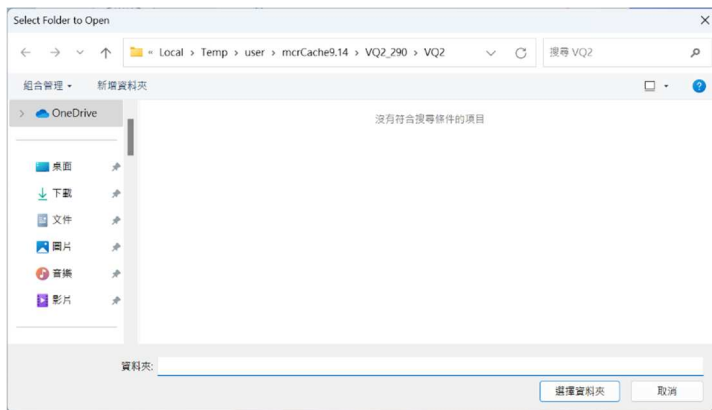
## 2 Tools

### 2.1 Picture to Video

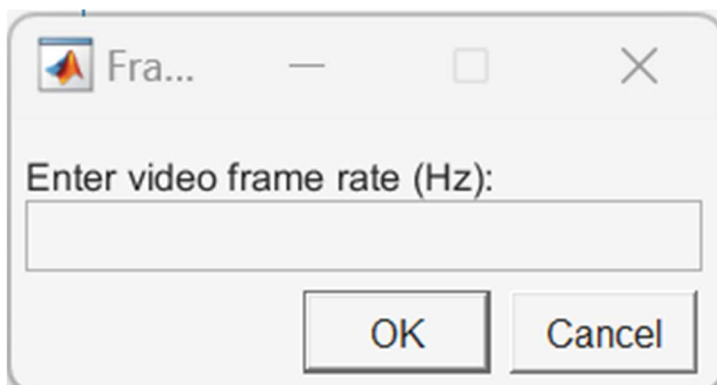


#### 2.1.1 BMP to AVI

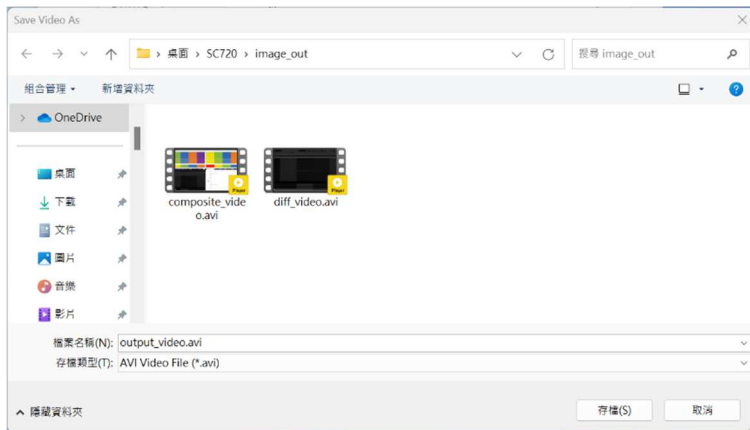
Step 1: Select the image (.bmp) path.



Step 2: Enter the number of frames per second for creating the video.

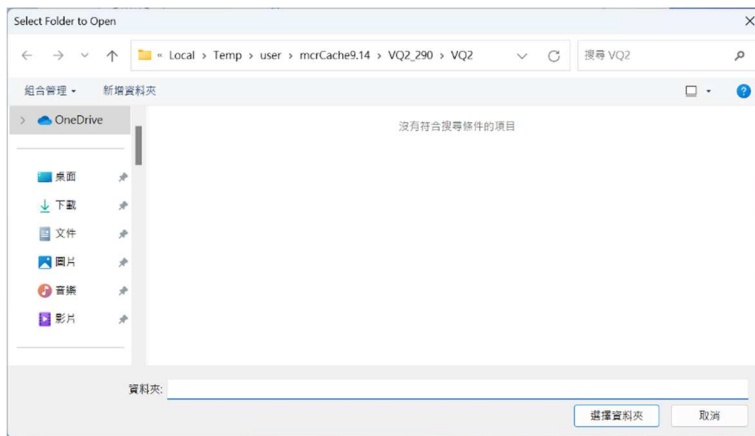


Step 3: Select the save path.

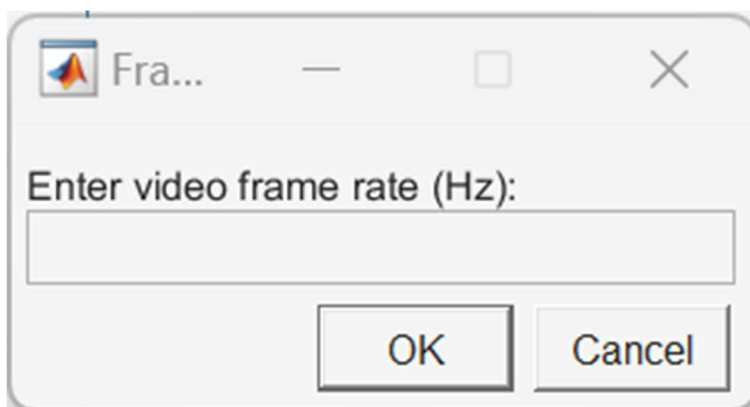


## 2.1.2 PNG to AVI

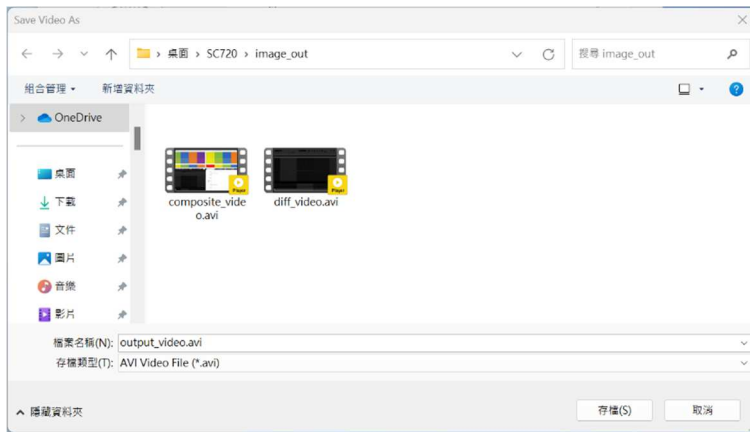
Step 1: Select the image (.png) path.



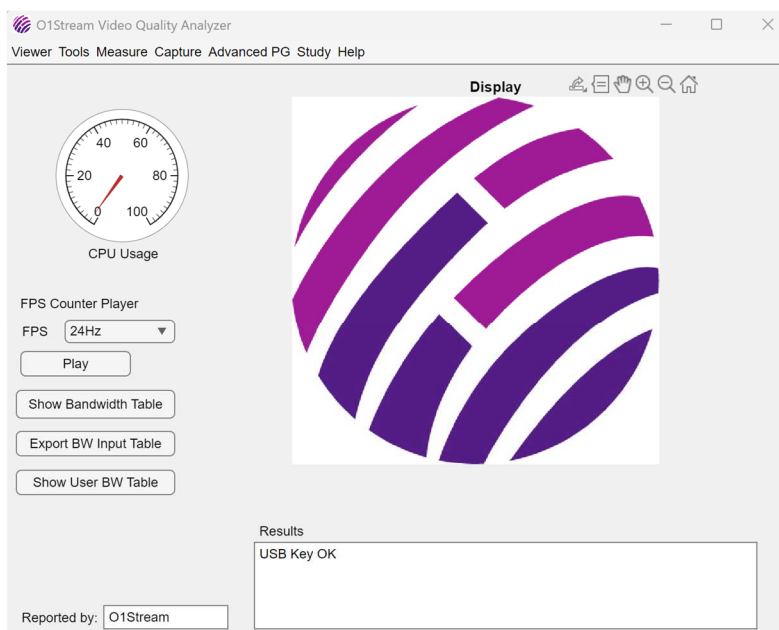
Step 2: Enter the number of frames per second for creating the video.



Step 3: Select the save path.



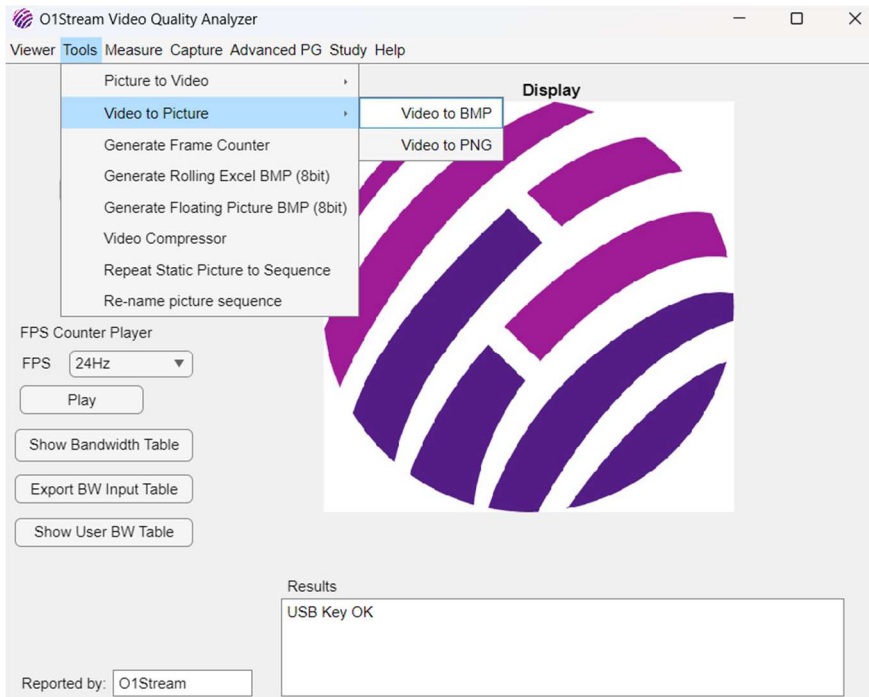
### 2.1.3 BMP (8bit) to AVI (10bit)



The function in this app is designed to streamline the process of converting BMP images into a 10-bit AVI video file. It guides users through a series of interactive steps, starting with folder selection for BMP files, followed by setting the desired frame rate and specifying the output video file name. The function then processes each BMP image by scaling its bit depth from 8-bit to 10-bit and saving the resulting images as intermediate PNG files in a designated subfolder. Utilizing FFmpeg, the function compiles these 10-bit

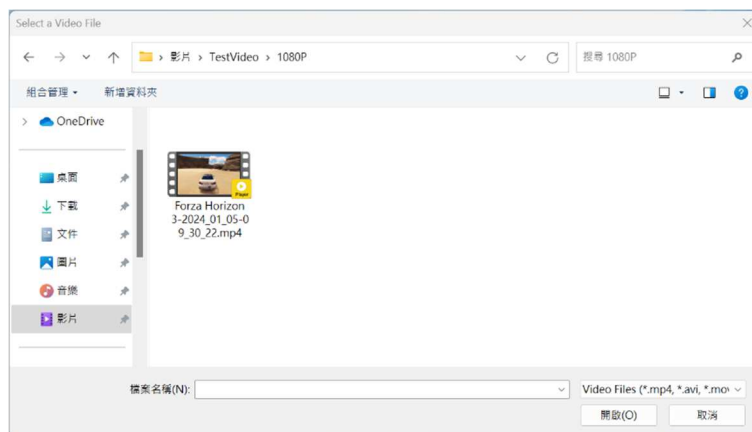
PNG frames into a high-quality AVI video in the YUV444 format. Throughout the process, the app provides real-time updates and handles errors gracefully.

## 2.2 Video to Picture

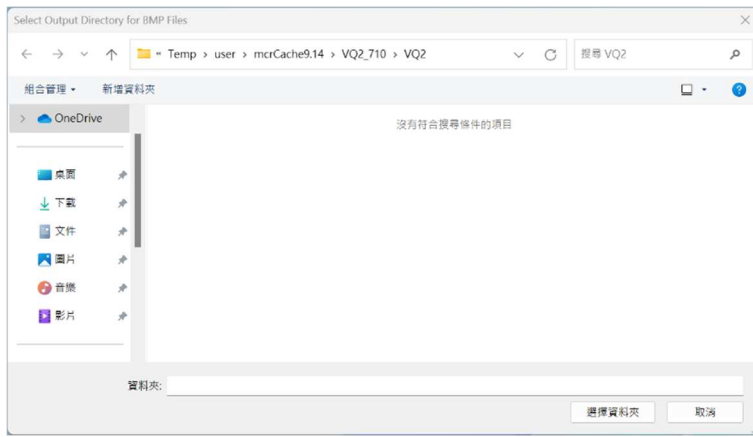


### 2.2.1 AVI to BMP

Step 1: Select the video file path.



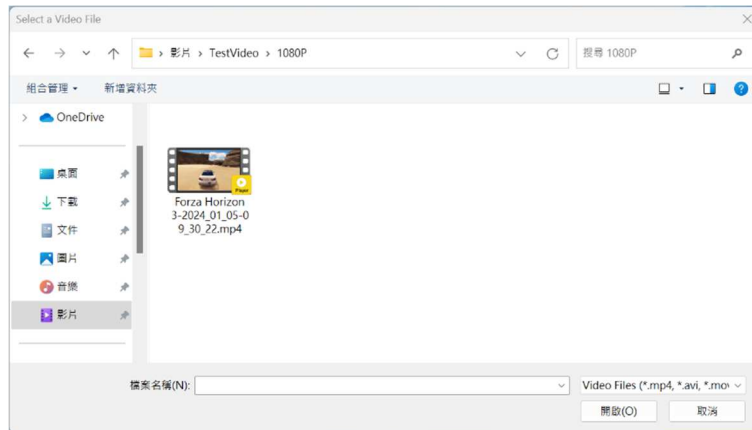
Step 2: Select the save path (.bmp).



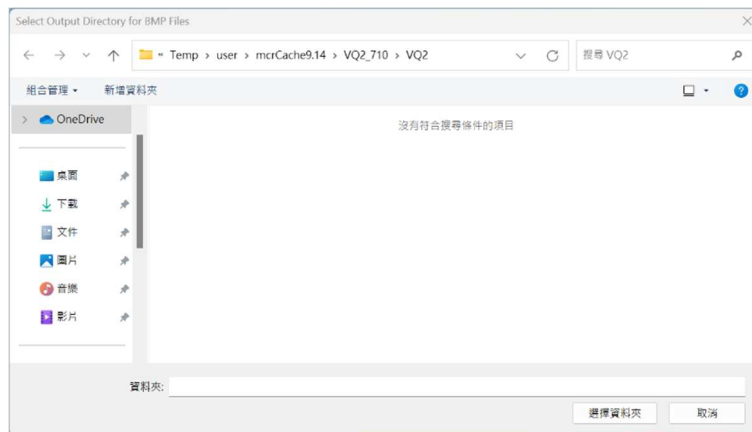


## 2.2.2 AVI to PNG

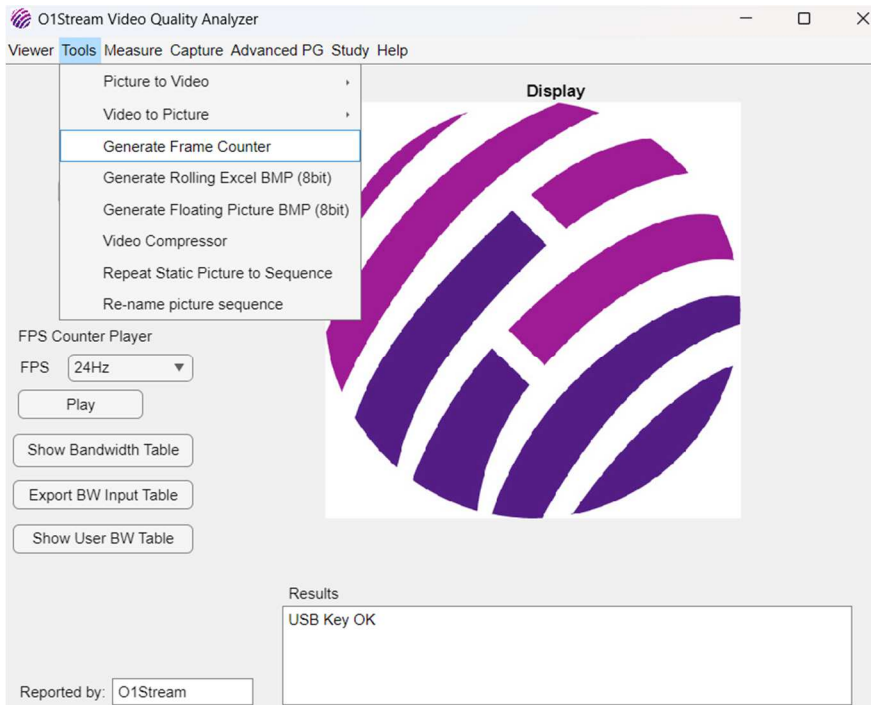
Step 1: Select the video path.



Step 2: Select the save path for images (.png).



## 2.3 Generate Frame Counter



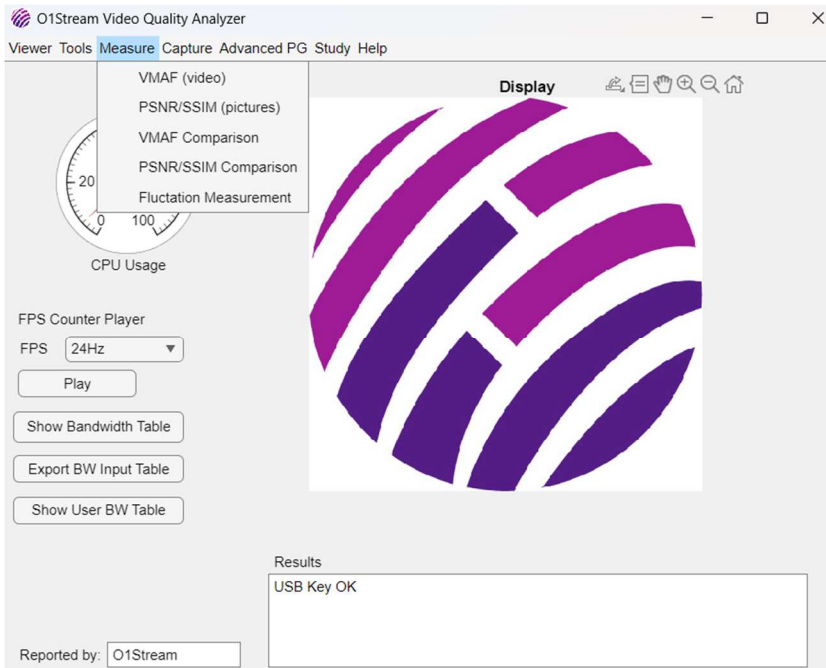
This function is designed to produce a visually dynamic **frame counter video** for testing purposes. It begins by prompting the user for critical parameters, such as the desired frame rate, video duration, and output file location and name. Once these details are provided, the function dynamically generates video frames featuring a prominently displayed frame counter timer in the center of the screen.

The video also incorporates user-defined text overlays, including a primary string entered by the user and a secondary string that adapts dynamically based on the content of the first. Additionally, animated elements such as bouncing squares and overlapping triangles move across the screen, creating a visually engaging output. To enhance visual appeal, the background color changes at regular frame intervals, adding another layer of dynamism to the video. Using MATLAB's VideoWriter class, the function compiles these frames into a high-quality MPEG-4

video file while validating user inputs, providing real-time status updates, and handling errors gracefully.

The primary purpose of this function is to test USB capture devices by generating a precise frame counter video to measure latency. The prominently displayed frame counter allows users to easily identify the number of frames delayed between the input and output of the capture device. The dynamic visual elements, such as moving shapes and changing backgrounds, further test the capture device's ability to handle complex visuals and maintain frame synchronization. With its customizable frame rate and duration, the function is adaptable to various testing scenarios. The resulting video provides a reliable and repeatable tool for evaluating the real-time performance of USB capture devices under different conditions.

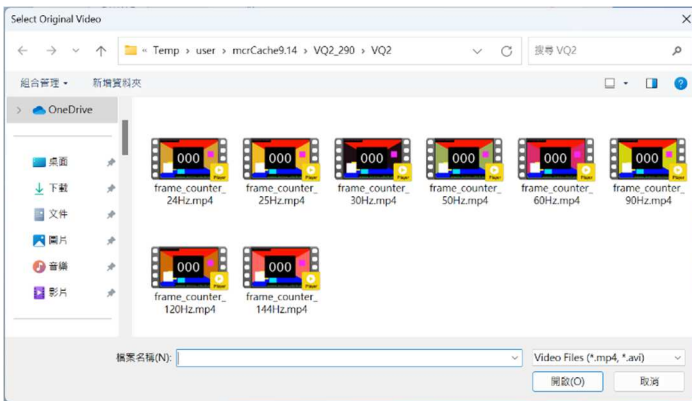
# 3 Measure



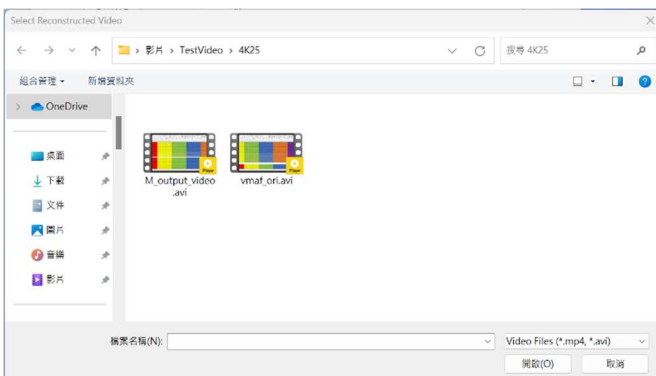
## 3.1 VMAF (video)

Step 1: Select the original video path.

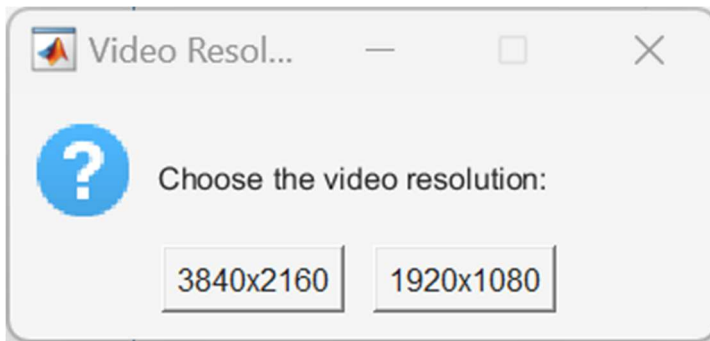
\*Note: To generate a video with the correct frame count displayed in the bottom right corner, use images created with the preamp.



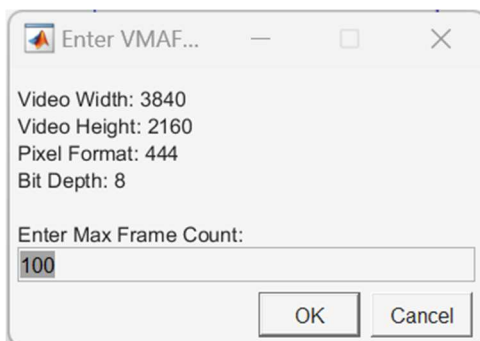
Step 2: Select the path for the extracted video (disturb).



Step 3: Select the video resolution.



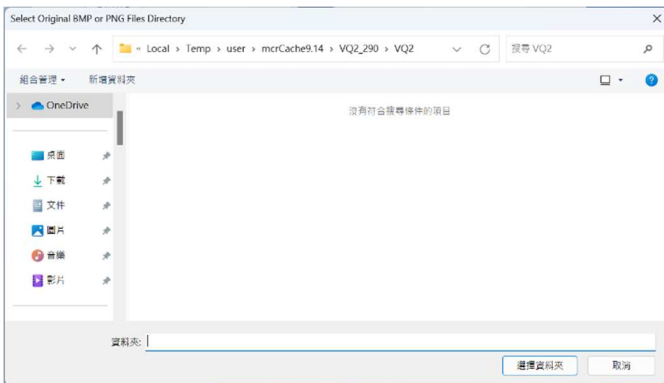
Step 4: Enter the max number of frames for the video to be analyzed.



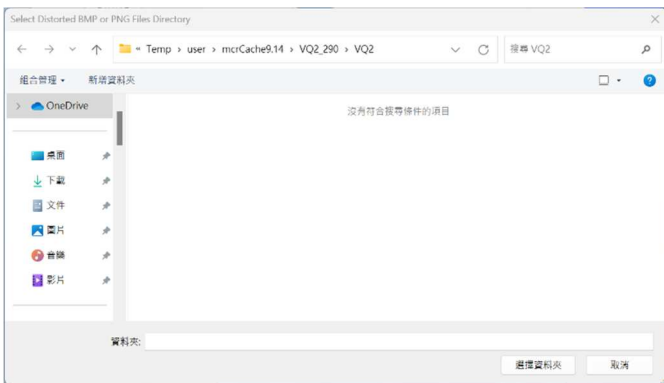
## 3.2 PSNR / SSIM (pictures)

Step 1: Select the original image path.

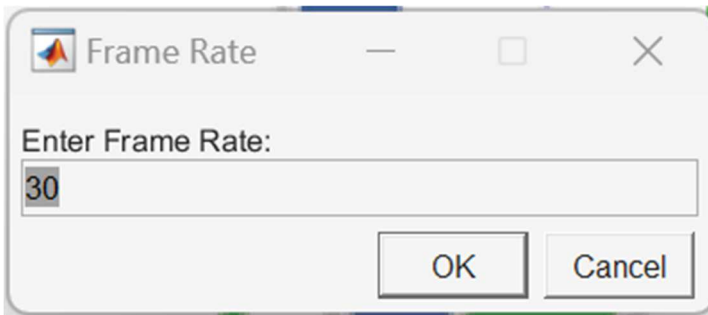
\*Note: Use images generated by the preamp to ensure the frame count is displayed in the bottom right corner.



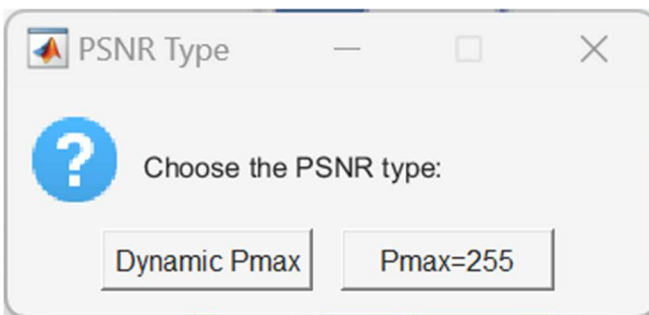
Step 2: Select the path for the extracted images (disturb).



Step 3: Enter the number of frames per second for testing.

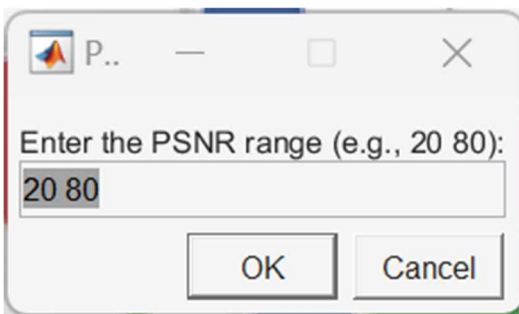


Step 4: Select the PSNR type.



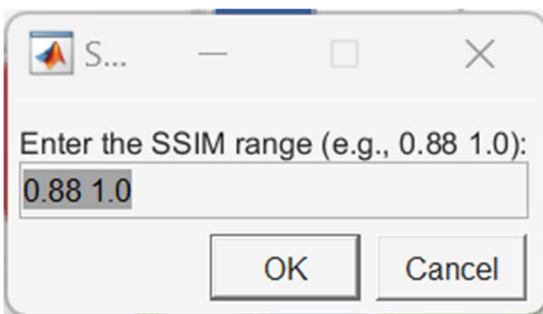
\*Note: Select Pmax = 255 for most cases.

Step 5: Enter the PSNR range to use for plotting the PSNR figure.



\*Note: The test results may be higher or lower than the set value.

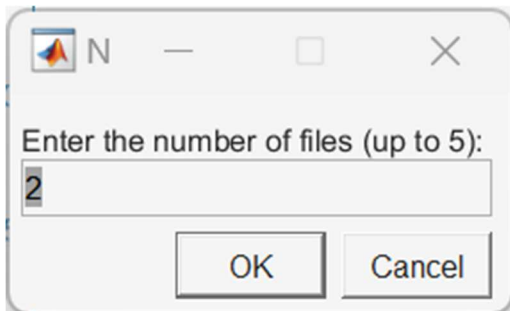
Step 6: Enter the SSIM range.



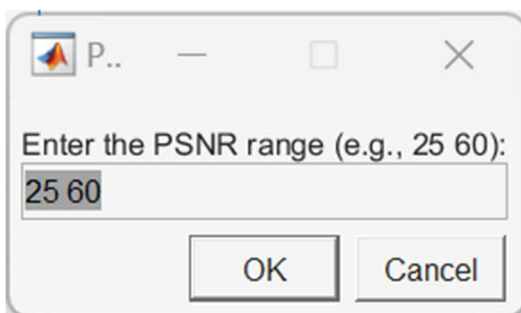
\*Note: The test results may exceed or fall below the set value.

### 3.3 VMAF Comparison

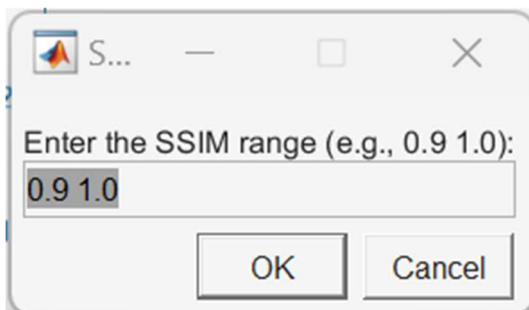
Step 1: Select the number of files to compare.



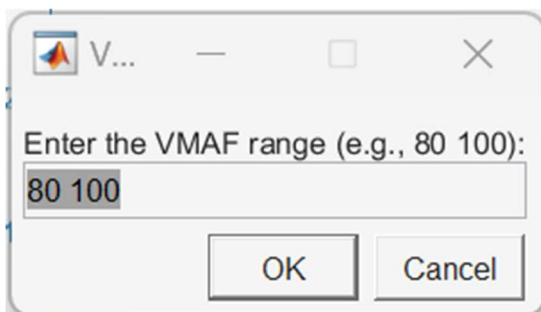
Step 2: Enter the PSNR range.



Step 3: Enter the SSIM range.

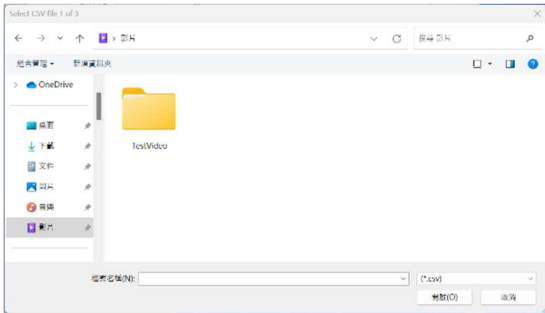


Step 4: Enter the VMAF range.

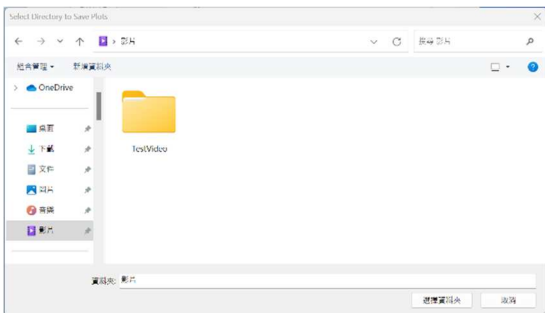




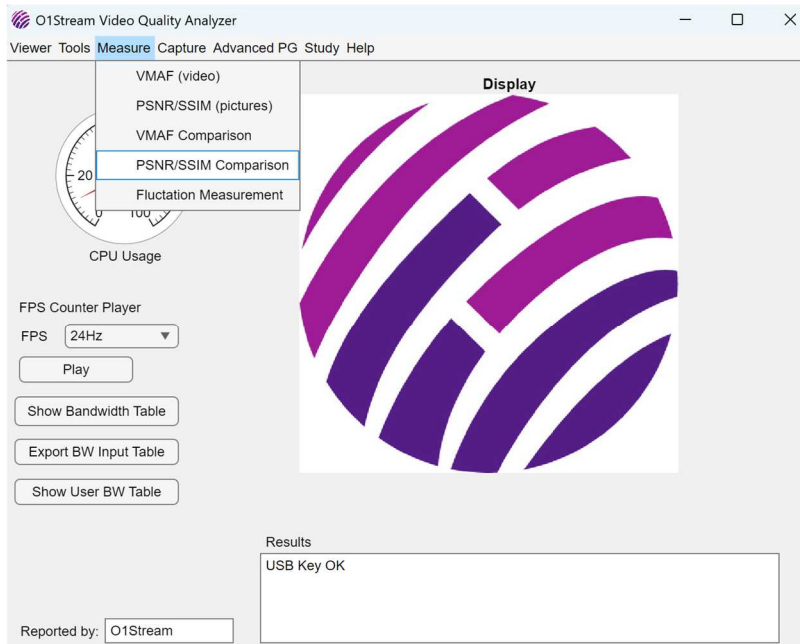
Step 5: Select the data file (.csv).



Step 6: Select the save path for the results.



## 3.4 PSNR / SSIM Comparison



The feature is designed to facilitate the comparison of PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index) metrics across multiple datasets stored in Excel files. It offers a structured, user-friendly process that includes file selection, metric extraction, visualization, and automated report generation. The detailed workflow ensures precise and customizable analysis for video quality comparisons.

### Detailed Process:

#### 1. Input Collection:

- Prompts the user to specify the number of Excel files to compare (up to 5).
- Asks for custom PSNR and SSIM ranges, allowing tailored visualization to fit the data's scale or analysis needs.

#### 2. File Selection:

- Guides the user through selecting individual Excel files containing PSNR and SSIM metrics.
- Validates file selection to ensure all required files are provided before proceeding.

### 3. **Save Location:**

- Prompts the user to select a directory for saving output plots and the final PDF report.
- Defaults to the last selected file's directory if no new location is specified.

### 4. **Data Extraction:**

- Reads PSNR and SSIM data from the selected Excel files.
- Stores the data in structured arrays for plotting and analysis.

### 5. **Visualization and Plotting:**

- Generates individual comparative plots for PSNR and SSIM metrics across all files.
- Saves these plots as high-quality PNG images for future reference.

### 6. **Report Generation:**

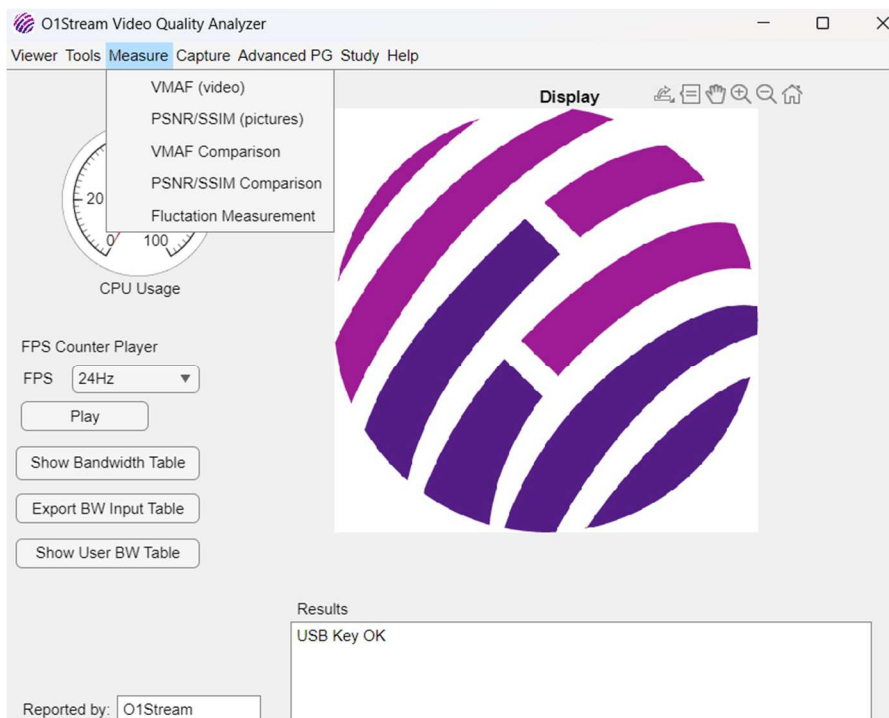
- Combines the plots into a single PDF report annotated with user-provided details such as an author name.
- Includes clear labels and comparisons to ensure the report is professional and easy to interpret.

### 7. **Final Output:**

- A well-organized report containing PSNR and SSIM comparison plots, saved as a PDF in the user-specified directory.
- Clear feedback in the app confirming the process completion and output location.

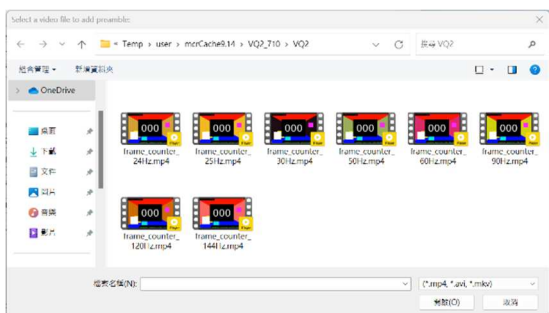
By automating the comparison process and providing customizable options, this function ensures efficient and accurate video quality analysis, making it an essential tool for evaluating and comparing video datasets in a structured manner.

# 4 Capture



## 4.1 Preamble

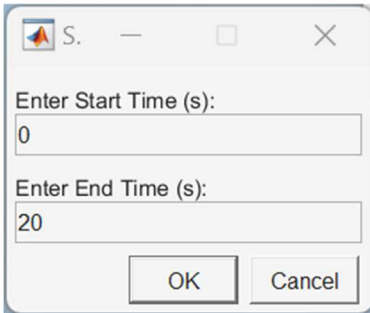
Step 1: Select the video to add the preamp.



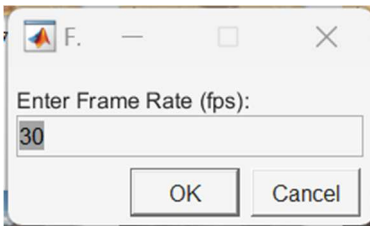
Step 2: Select the save path for the images after adding the preamp.



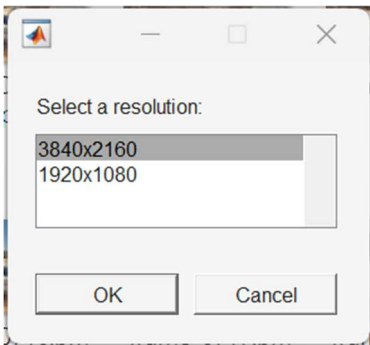
Step 3: Set the start and end times of the original video.



Step 4: Set the number of frames per second for the video.

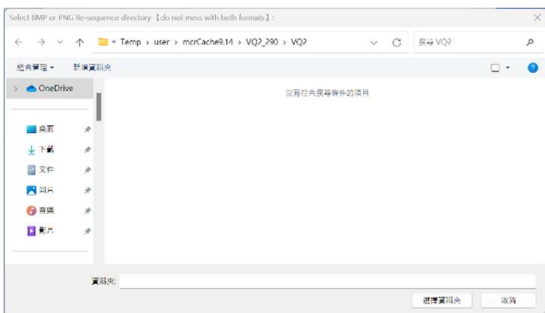


Step 5: Set the video resolution.

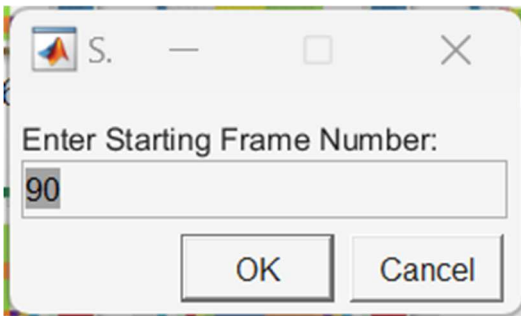


## 4.2 Re-Seq

Step 1: Select the path for the captured images.

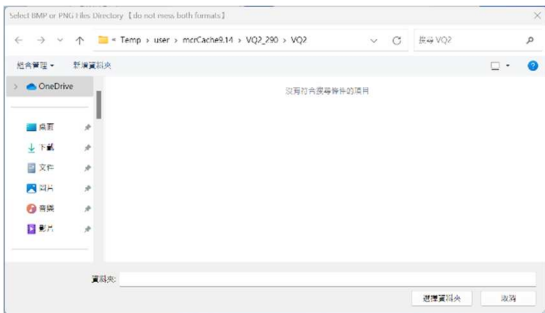


Step 2: Specify from which image number the extracted or captured images should be retained.

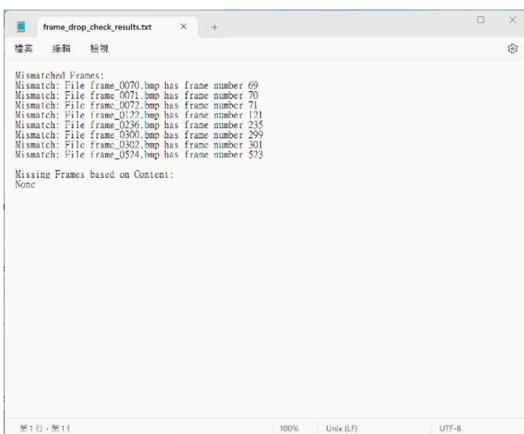
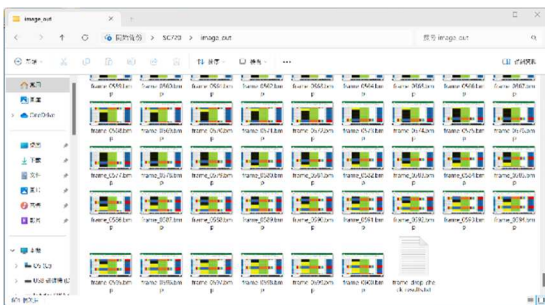


### 4.3 Frame drop check

Step 1: Select the path for the extracted or captured images.



Step 2: Generate the inspection results.



\*note:

1. **Mismatched Frames:** Indicates that the file name of the image does not match the frame number within the image.
2. **Missing Frames Based on Content:** Indicates frame numbers that cannot be found among all the images.



## 5 Study

In this section, we present several examples to illustrate the underlying meaning of PSNR and SSIM, two commonly used metrics for assessing video quality. These measurements are widely recognized and generally provide a reliable indication of quality differences between original and compressed videos or images. For most scenarios, they serve as useful benchmarks for evaluating video fidelity. However, it is important to acknowledge that PSNR and SSIM do not always capture the nuances of perceived video quality and can occasionally produce misleading results, particularly in cases where visual perception differs from mathematical accuracy.

To address these limitations, we encourage users to explore these intuitive examples, which demonstrate scenarios where PSNR and SSIM may not align with human visual judgment. By doing so, users can develop a better understanding of these metrics' strengths and weaknesses. For a more comprehensive evaluation of video quality, it is recommended to use multiple measurement methods in combination, incorporating subjective assessments or other advanced metrics to ensure a well-rounded analysis.

- Simple RGB to NV12 Conversion

Example of RGB to NV12 conversion shows that the best PSNR approach doesn't always yield the best SSIM, highlighting differences in picture quality.



- PSNR paradox

Example of images with comparable PSNR values but distinctly different in visual perception.

**Noisy Original**  
PSNR: 34.36



**Blurred Image**  
PSNR: 34.31



Noisy Original  
PSNR: 39.02



Blurred Image  
PSNR: 39.10



Noisy Original  
PSNR: 32.33

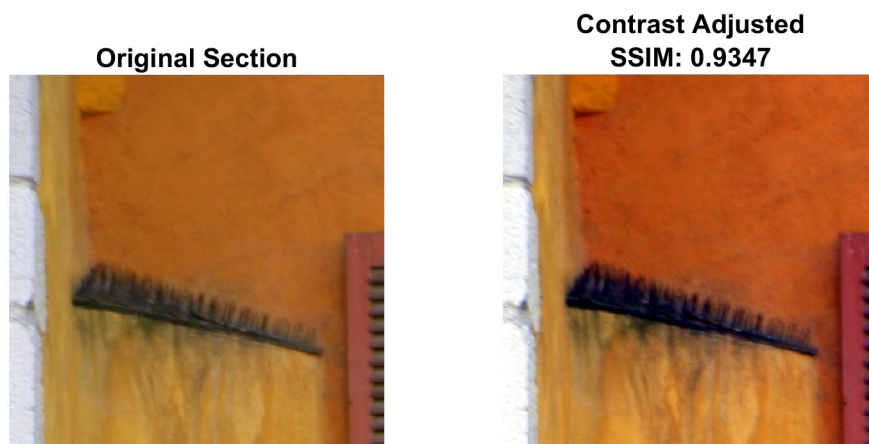


Blurred Image  
PSNR: 32.42



- SSIM paradox

Example of images with comparable SSIM values but distinctly different in visual perception.



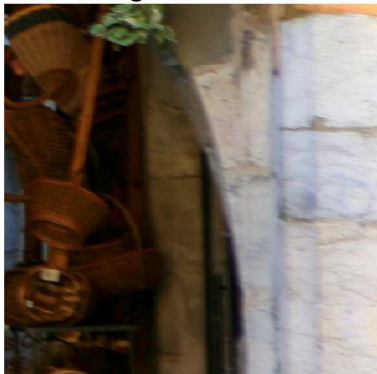
**Original Section**



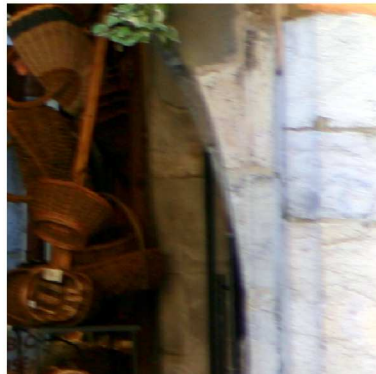
**Contrast Adjusted  
SSIM: 0.6509**



**Original Section**



**Contrast Adjusted  
SSIM: 0.9791**



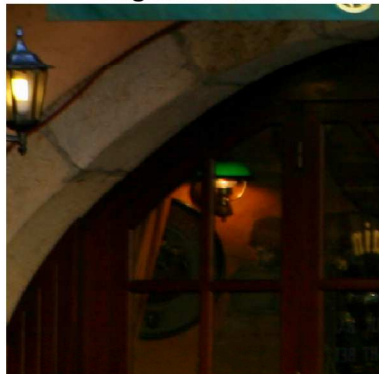
Original Section



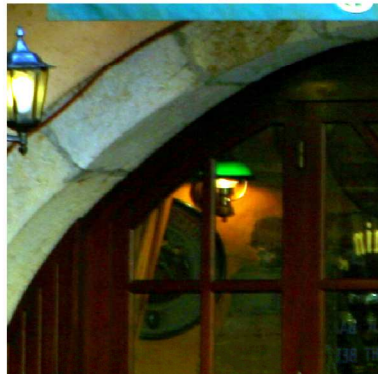
Contrast Adjusted  
SSIM: 0.8221



Original Section



Contrast Adjusted  
SSIM: 0.6169



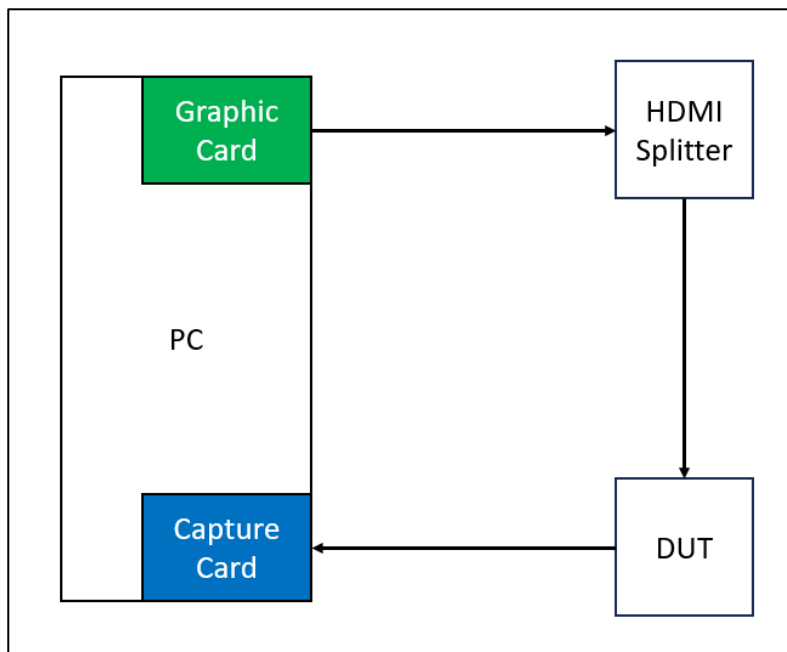
## 6 Help

The help section includes several PDF files, such as the video evaluation flow, video quality analyzer SOP, video metrics analysis guide, and the advanced PG guide. These resources provide valuable insights into how to effectively use the video quality analyzer and explore the key concepts behind this powerful toolkit.



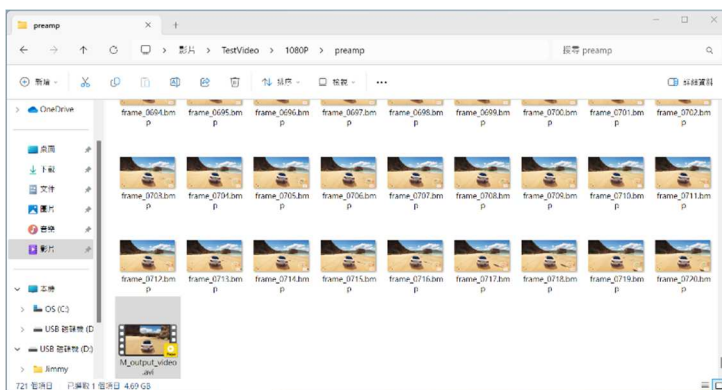
## 7 Video Capture Setting Example

A 1920x1080P60 video is available and processed through the device under test (DUT). Frames are extracted and compared with the original video to evaluate PSNR, SSIM, and VMAF values. The testing framework is as follows:



In our example setup, we use a 4K60-capable PCIe capture card and utilize Python with OpenCV to capture images from the PCIe card.

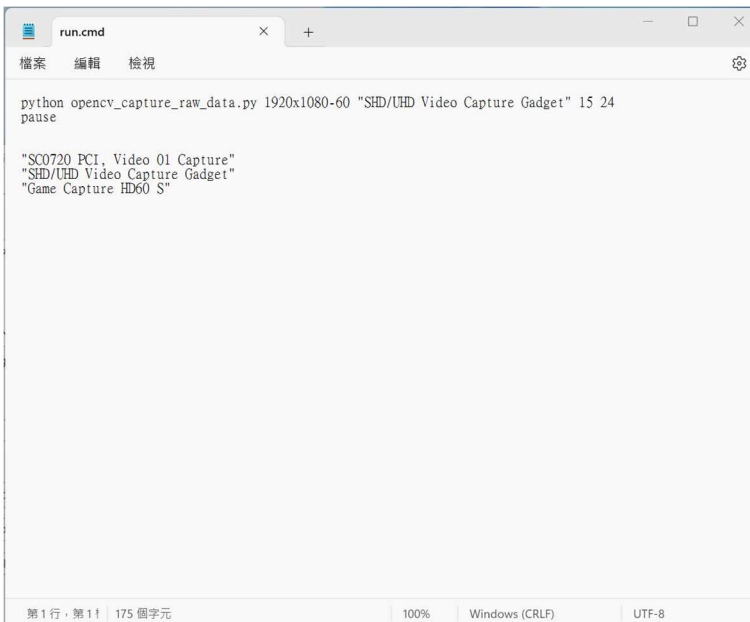
Step 1: Add the preamp to the original video.



Step 2: Verify that the resolution of the graphics card output matches the video resolution.



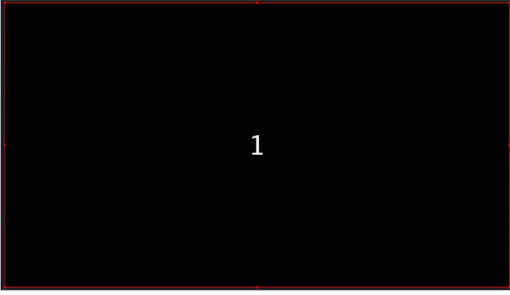
Step 3: Verify the OpenCV command.



\*Note: Explanation of the OpenCV command is as follows:

- **Resolution-Frame Rate:** 1920x1080-60
- **Capture Card Name:** "SHD/UHD Video Capture Gadget"
- **Capture Duration:** 15 seconds
- **Number of Threads:** 24

Step 4: Use VLC Player to play the preamp video and switch to full-screen mode.

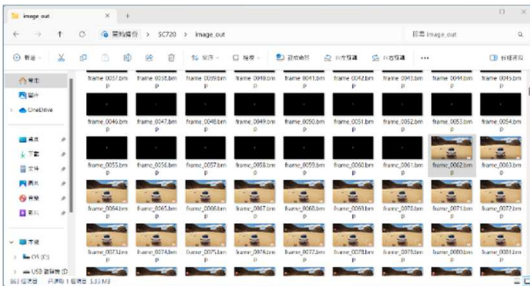


\*Note: Other media players may experience frame drops; it is recommended to use VLC Player.

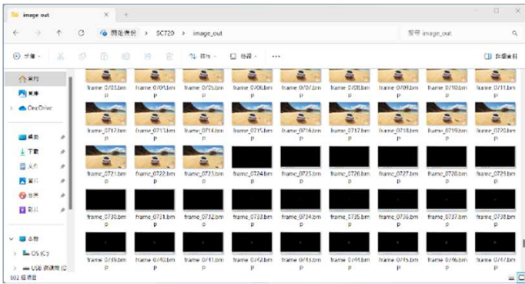
Step 5: Run the batch file to start capturing frames and wait for the process to complete.

```
C:\Windows\system32\cmd.exe
C:\Users\user\Desktop\SC720>python opencv_capture_raw_data.py 1920x1080-60 "SHD/UHD Video Capture Gadget" 15 24
Video Capture Setup
SHD/UHD Video Capture Gadget index = 4
For normal case,
Video Capture Done
Multiple Thread number: 24
Start Capturing
(1080, 1920, 3)
fps: 23 total: 23
fps: 60 total: 83
fps: 61 total: 144
fps: 60 total: 244
fps: 61 total: 265
fps: 60 total: 325
fps: 61 total: 386
fps: 60 total: 447
fps: 61 total: 507
fps: 61 total: 568
fps: 61 total: 629
fps: 61 total: 690
fps: 60 total: 750
fps: 61 total: 811
863
Done.
read pics: 863, capture pics: 863, total fps: 57.53333333333333
C:\Users\user\Desktop\SC720>pause
请按任意键继续 . . .
```

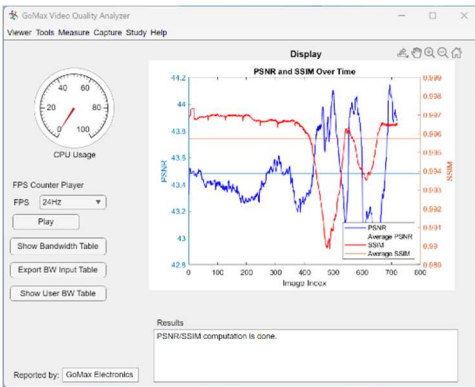
Step 6: Use the Re-Seq function to remove the preamp frames.



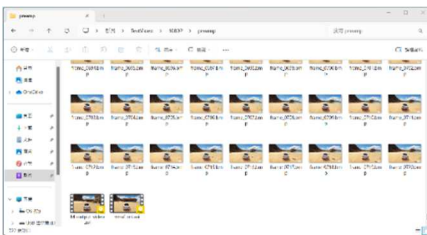
Step 7: Delete the extra images and use the Frame Drop Check function to ensure no frames were dropped.



Step 8: Use the PSNR/SSIM function to evaluate the quality of the frames.

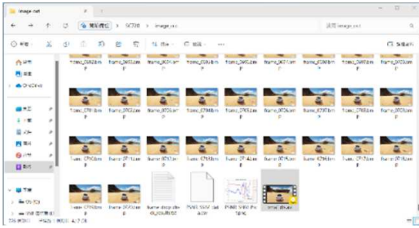


Step 9: Use the BMP to AVI function to generate the **VMAF\_ori** video.



Note: The **VMAF\_ori** video is created from images generated by the preamp (with frame numbers displayed in the bottom right corner).

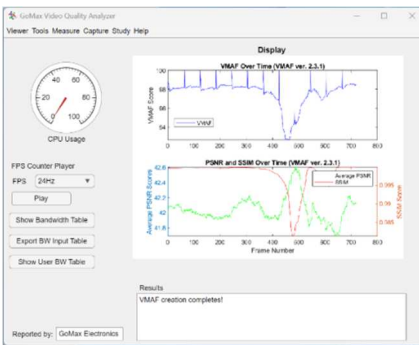
Step 10: Use the BMP to AVI function to generate the **VMAF\_dis** video.



*Note:*

1. If the captured images are in PNG format, use the **PNG to AVI** function.
2. Ensure there are no extra PNG files in the directory (e.g., PSNR\_SSIM\_Plot.png) when processing the PNG images.

Step 11: Use the **VMAF** function to evaluate video quality.



**Notes:**

1. Ensure the **graphics card** and **PCIe capture card** do not drop frames. It is especially possible that a not-qualified PCIe capture card will constantly and frequently loses pictures.
2. Use an qualified HDMI 2.0 Splitter with a fixed EDID to ensure consistent output from the graphics card across all devices under test.
3. Confirm that all testing paths maintain the same resolution, frame rate, color space, and color depth.