Part 06C — MP4 Video Frame Extraction and Image Workflow

Using ChatGPT and Python to Extract Photos from Video Files

Yahya Nazer

2026.06.18

Table of contents

Purpose

In this part, you will learn how to use **ChatGPT and Python** to build a practical tool for photographers who work with video files.

The goal is to create a Python program that can:

- Ask the user to select an MP4 video file

- Extract one image at regular time intervals
- Save the extracted images as high-quality files
- Create a new output folder automatically
- Keep the process simple for Mac and Windows users

This is useful when a photographer records a video and later wants to extract still images for review, printing, documentation, or contact sheets.

What You Will Build

You will build a Python GUI program that extracts still images from an MP4 file.

The program will:

1. Open a graphical file selection window.
2. Let the user select one `.mp4` video.
3. Create a folder next to the video file.
4. Name the folder using the current date and time.
5. Extract one image every selected number of seconds.
6. Save the images as JPEG files.
7. Save the images with 300 DPI metadata.
8. Show a log of what happened.

Why This Project Is Important

Photographers often capture video during events, travel, sports, or family activities. Later, they may want to pull still images from the video.

Doing this manually can be slow. Python can automate the process.

This project teaches several important programming ideas:

- Working with video files
- Reading video metadata
- Using GUI file dialogs
- Creating output folders
- Writing numbered image files
- Logging program activity
- Using ChatGPT to improve and debug Python code

Suggested Folder Structure

Use the standard CDL Python course folder structure.

```
Part-06C-MP4-Video-Frame-Extraction
|
  A-Data
|     sample-video.mp4
|
  B-Engines
|     MP4-Extract-Images-GUI-V01.py
|
  C-Results
|     images-yyyy-mm-dd--hh-mm
|         frame-0001.jpg
|         frame-0002.jpg
```

```
|          frame-0003.jpg
|
  D-Documentation
      Part-06C-MP4-Video-Frame-Extraction.html
```

Required Python Libraries

This project uses the following Python libraries:

```
tkinter
pathlib
datetime
cv2
PIL
```

You may need to install these packages:

```
pip install opencv-python pillow
```

On Mac, use:

```
python3 -m pip install opencv-python pillow
```

On Windows, use:

```
py -m pip install opencv-python pillow
```

Step 1 — Ask ChatGPT for the First Version

You can ask ChatGPT:

> Please write a Python GUI program for Mac and Windows. The program should ask the user to select an MP4 file. It should extract one image every 60 seconds. It should create a folder called images-yyyy-mm-dd–hh-mm next to the selected MP4 file. It should save each image as JPG with 300 DPI. It should show a log window. Please use tkinter, OpenCV, and Pillow.

Step 2 — Basic Python Program

Save the following program as:

```
B-Engines/MP4-Extract-Images-GUI-V01.py
```

```
# ============================================================
# Program: MP4-Extract-Images-GUI-V01.py
# Purpose:
#   Select an MP4 file and extract still images at regular
#   time intervals. Images are saved as 300 DPI JPEG files.
#
# Script requested by user:
#   Build a Mac/Windows friendly GUI that selects an MP4 file,
#   creates an images-yyyy-mm-dd--hh-mm folder, extracts frames,
#   and writes a clear log.
#
# Generated with help from ChatGPT.
# ============================================================
```

```
import tkinter as tk
from tkinter import filedialog, messagebox
from pathlib import Path
from datetime import datetime
import cv2
from PIL import Image


class MP4FrameExtractorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("MP4 Extract Images - 300 DPI")
        self.root.geometry("850x600")
        self.root.configure(bg="#f4f4f4")

        self.video_path = None
        self.interval_seconds = tk.IntVar(value=60)

        self.build_gui()

    def build_gui(self):
        title = tk.Label(
            self.root,
            text="MP4 Extract Images",
            font=("Arial", 22, "bold"),
            bg="#f4f4f4",
            fg="black"
        )
        title.pack(pady=15)

        button_frame = tk.Frame(self.root, bg="#f4f4f4")
        button_frame.pack(pady=10)

        self.select_button = tk.Button(
            button_frame,
            text="1) Select MP4 File",
            command=self.select_video,
            bg="#d9ead3",
            fg="black",
            width=22,
            height=2
        )
        self.select_button.grid(row=0, column=0, padx=8)

        self.extract_button = tk.Button(
            button_frame,
            text="2) Extract Images",
            command=self.extract_images,
            bg="#fce5cd",
            fg="black",
            width=22,
            height=2
        )
```

```
        self.extract_button.grid(row=0, column=1, padx=8)

        interval_frame = tk.Frame(self.root, bg="#f4f4f4")
        interval_frame.pack(pady=10)

        tk.Label(
            interval_frame,
            text="Extract one image every seconds:",
            bg="#f4f4f4",
            fg="black",
            font=("Arial", 12)
        ).grid(row=0, column=0, padx=8)

        tk.Entry(
            interval_frame,
            textvariable=self.interval_seconds,
            width=10,
            font=("Arial", 12)
        ).grid(row=0, column=1, padx=8)

        self.selected_label = tk.Label(
            self.root,
            text="No MP4 file selected.",
            bg="#f4f4f4",
            fg="black",
            font=("Arial", 11)
        )
        self.selected_label.pack(pady=8)

        self.log_text = tk.Text(
            self.root,
            height=22,
            width=100,
            bg="white",
            fg="black",
            font=("Courier New", 10)
        )
        self.log_text.pack(padx=20, pady=15)

    def log(self, message):
        self.log_text.insert(tk.END, message + "\n")
        self.log_text.see(tk.END)
        self.root.update_idletasks()

    def select_video(self):
        file_path = filedialog.askopenfilename(
            title="Select MP4 File",
            filetypes=[("MP4 files", "*.mp4"), ("All files", "*.*")]
        )

        if not file_path:
            self.log("No file selected.")
            return
```

```
        self.video_path = Path(file_path)
        self.selected_label.config(text=str(self.video_path))
        self.log(f"Selected file: {self.video_path}")

    def extract_images(self):
        if not self.video_path:
            messagebox.showerror("Missing File", "Please select an MP4 file first.")
            return

        try:
            interval = int(self.interval_seconds.get())
            if interval <= 0:
                raise ValueError
        except ValueError:
            messagebox.showerror("Invalid Interval", "Please enter a positive number of seconds.")
            return

        timestamp = datetime.now().strftime("%Y-%m-%d--%H-%M")
        output_folder = self.video_path.parent / f"images-{timestamp}"
        output_folder.mkdir(exist_ok=True)

        self.log(f"Output folder: {output_folder}")

        video = cv2.VideoCapture(str(self.video_path))

        if not video.isOpened():
            messagebox.showerror("Video Error", "Could not open the selected video.")
            return

        fps = video.get(cv2.CAP_PROP_FPS)
        total_frames = int(video.get(cv2.CAP_PROP_FRAME_COUNT))
        duration_seconds = total_frames / fps if fps else 0

        self.log(f"Frames per second: {fps:.2f}")
        self.log(f"Total frames: {total_frames}")
        self.log(f"Duration seconds: {duration_seconds:.2f}")

        saved_count = 0
        current_second = 0

        while current_second <= duration_seconds:
            frame_number = int(current_second * fps)
            video.set(cv2.CAP_PROP_POS_FRAMES, frame_number)

            success, frame = video.read()

            if not success:
                self.log(f"Could not read frame at {current_second} seconds.")
                current_second += interval
                continue

            frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            image = Image.fromarray(frame_rgb)
```

```
            saved_count += 1
            output_file = output_folder / f"frame-{saved_count:04d}.jpg"
            image.save(output_file, "JPEG", quality=95, dpi=(300, 300))

            self.log(f"Saved: {output_file.name}")

            current_second += interval

        video.release()

        self.log("")
        self.log(f"Done. Total images saved: {saved_count}")
        messagebox.showinfo("Done", f"Saved {saved_count} images.")


if __name__ == "__main__":
    root = tk.Tk()
    app = MP4FrameExtractorApp(root)
    root.mainloop()
```

Step 3 — Test the Program

To test the program:

1. Open VS Code.
2. Open the project folder.
3. Open the file:

```
B-Engines/MP4-Extract-Images-GUI-V01.py
```

4. Run the program.
5. Select an MP4 video.
6. Click **Extract Images**.
7. Check the new `images-yyyy-mm-dd--hh-mm` folder.

Step 4 — Ask ChatGPT to Improve the Program

After the first version works, ask ChatGPT to improve it.

Example prompt:

> Please improve this program by adding: 1. A selected file label 2. A progress log 3. Better Mac-friendly button colors 4. A start/end time option 5. A user setting for extraction interval 6. Better error handling Please do not change the main algorithm.

Step 5 — Add Start and End Time Options

A useful improvement is to allow the user to extract frames only from part of the video.

For example:

- Start at 60 seconds
- End at 300 seconds
- Extract one image every 30 seconds

This prevents the program from extracting images from unwanted sections.

Step 6 — Add Output Naming Rules

Good file naming is important.

Recommended output file names:

```
video-name-frame-0001.jpg
video-name-frame-0002.jpg
video-name-frame-0003.jpg
```

This helps when multiple video files are processed.

Step 7 — Common Problems

Problem 1 — OpenCV Is Not Installed

Error:

```
ModuleNotFoundError: No module named 'cv2'
```

Solution:

```
pip install opencv-python
```

or on Mac:

```
python3 -m pip install opencv-python
```

Problem 2 — Pillow Is Not Installed

Error:

```
ModuleNotFoundError: No module named 'PIL'
```

Solution:

```
pip install pillow
```

Problem 3 — Video Cannot Be Opened

Possible causes:

- File is not really an MP4 file
- File is damaged
- File path contains unusual characters
- Codec is not supported by OpenCV

Try another MP4 file first.

Step 8 — Student Exercise

Modify the program so that it:

1. Allows JPG or PNG output.
2. Allows the user to choose the interval.
3. Adds the video file name to each image file.
4. Saves a text log file in the output folder.

5. Adds a progress percentage.

Step 9 — Checklist

Before moving to the next part, confirm:

☐ I can select an MP4 file.
☐ I can extract images from the video.
☐ The output folder is created automatically.
☐ The image files are saved with numbered names.
☐ The saved images have 300 DPI metadata.
☐ I understand how ChatGPT helped build and improve the program.
☐ I can explain what OpenCV does.
☐ I can explain what Pillow does.

Summary

In this part, you built a real-world Python tool for extracting still images from MP4 videos.

You learned how to:

- Use ChatGPT to design a Python automation tool
- Create a GUI with Tkinter
- Select a video file
- Read video information with OpenCV
- Extract frames at regular intervals
- Save images with Pillow
- Create an organized output folder
- Document and improve a practical photography workflow

This project prepares you for more advanced image and media automation projects.