

A photograph of three surgeons in an operating room. They are wearing green scrubs, blue surgical caps, and white masks. They are gathered around a patient who is lying on a table, covered with a green drape. A large, circular surgical light is positioned above them, casting a bright glow. In the background, there are medical monitors and equipment. The overall atmosphere is professional and clinical.

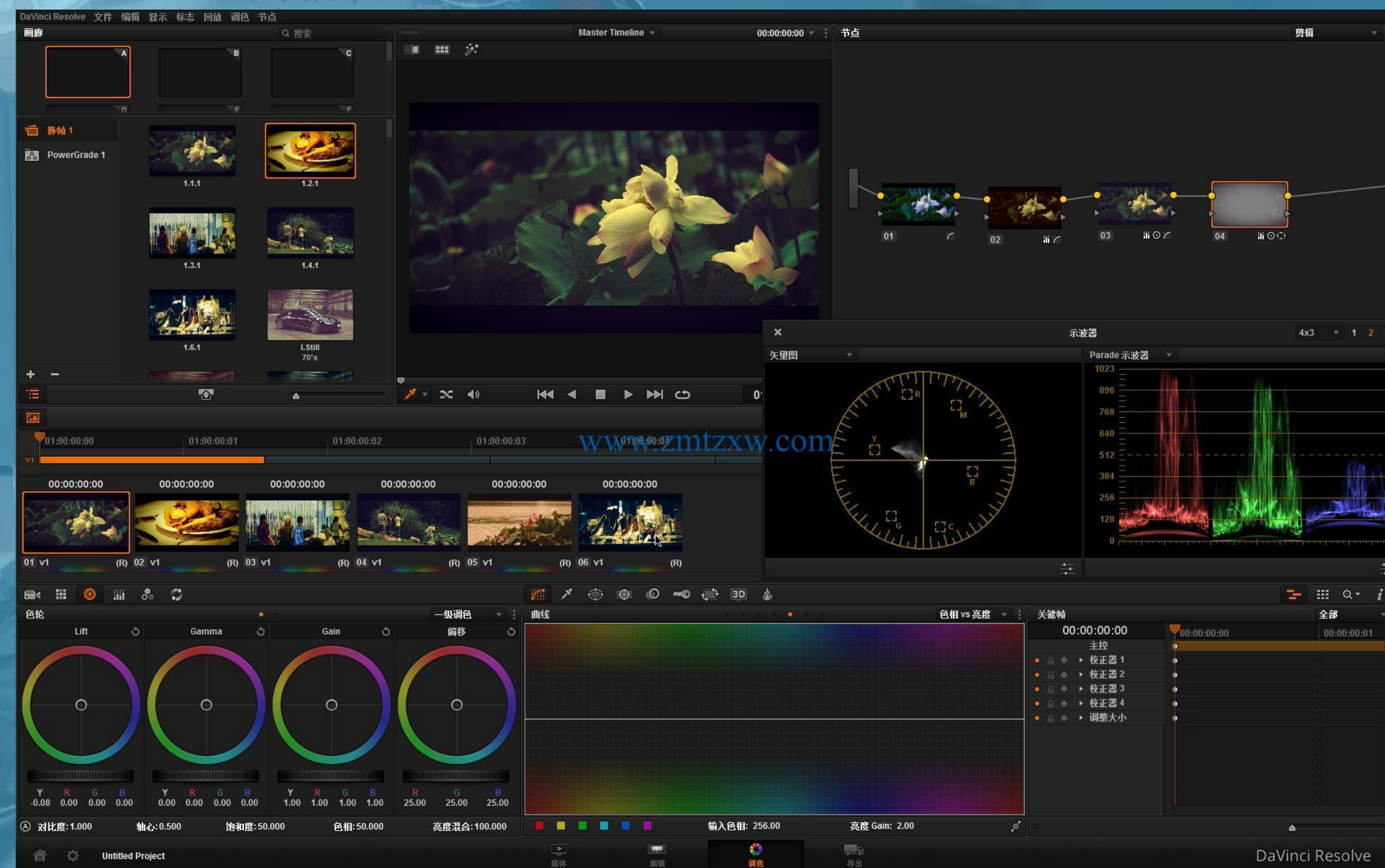
医学手术视频自动剪辑

第 13 小组

工作内容概述

一. 大幅降低手术视频的
数据容量

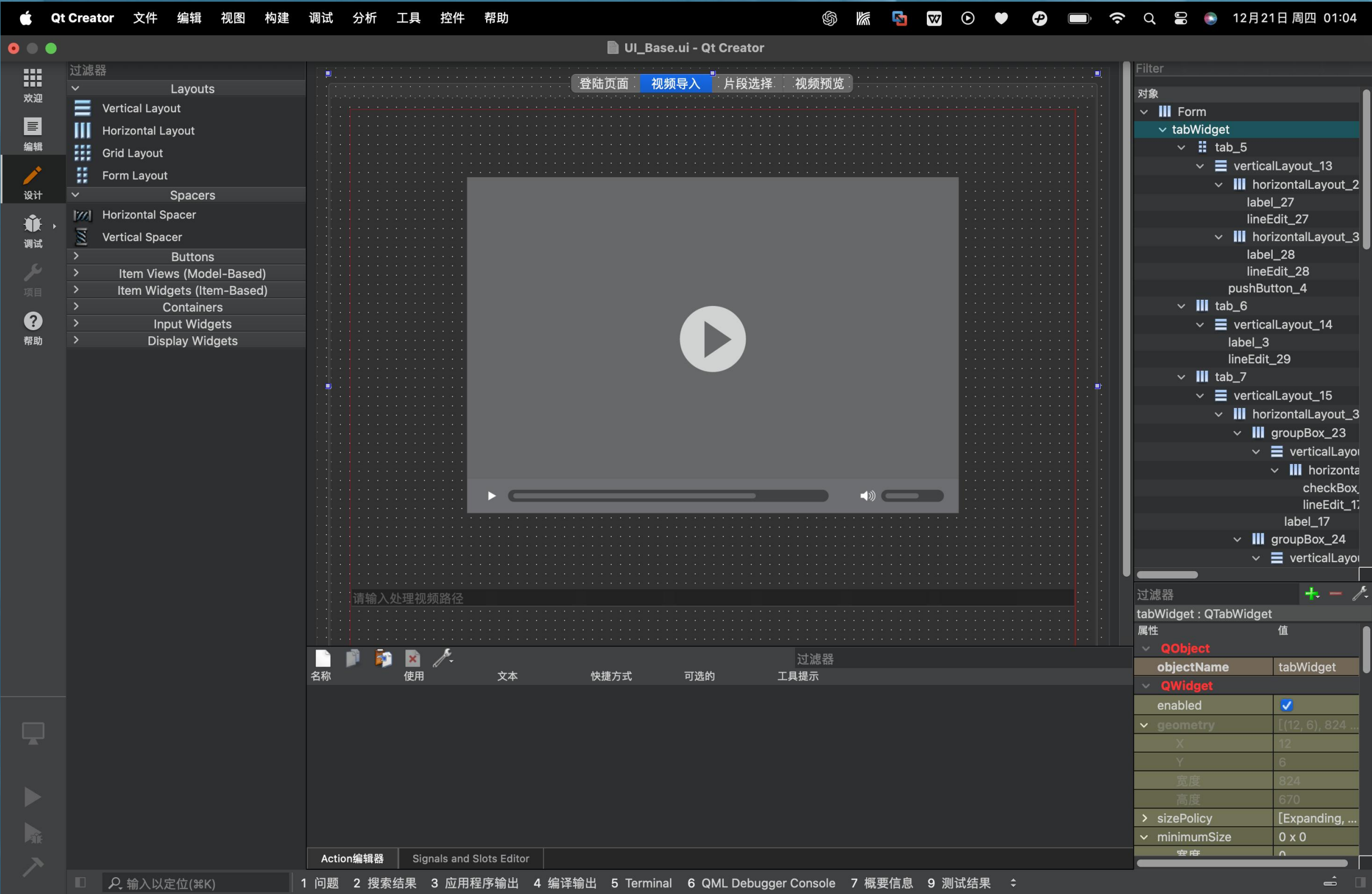
二. 优化视频资料在
医学教学与科研
中的检索及回放
等应用。



Part 1 UI 界面设计

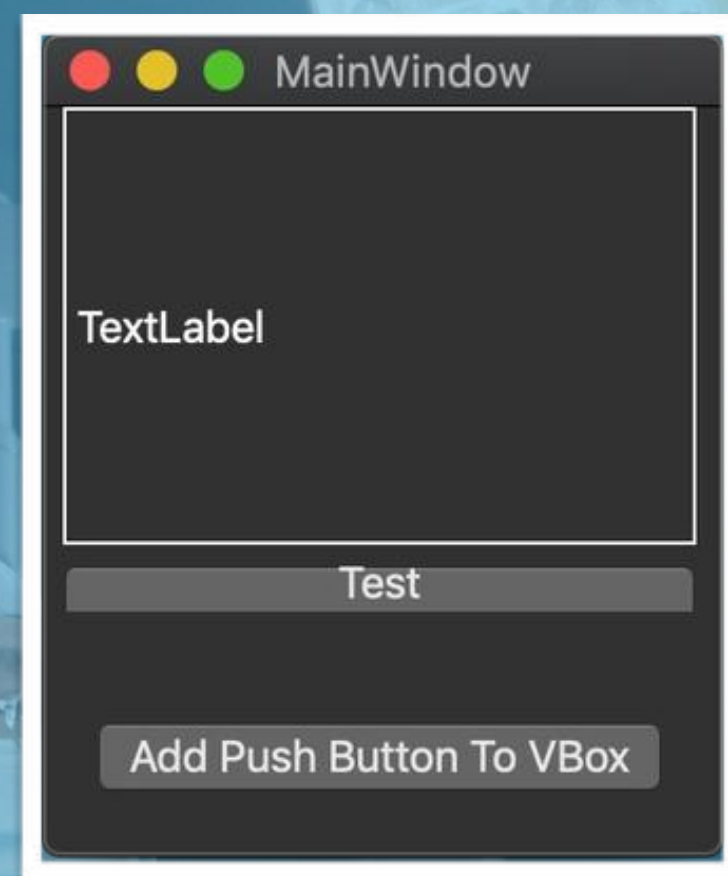
PySide6 + Qt

用户界面



主要运用模块

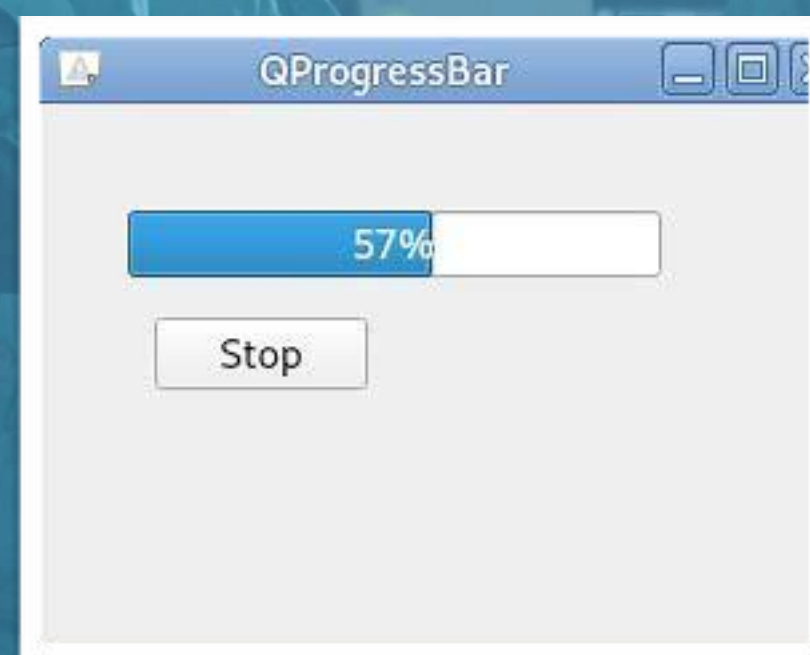
QPushButton
用于添加按钮



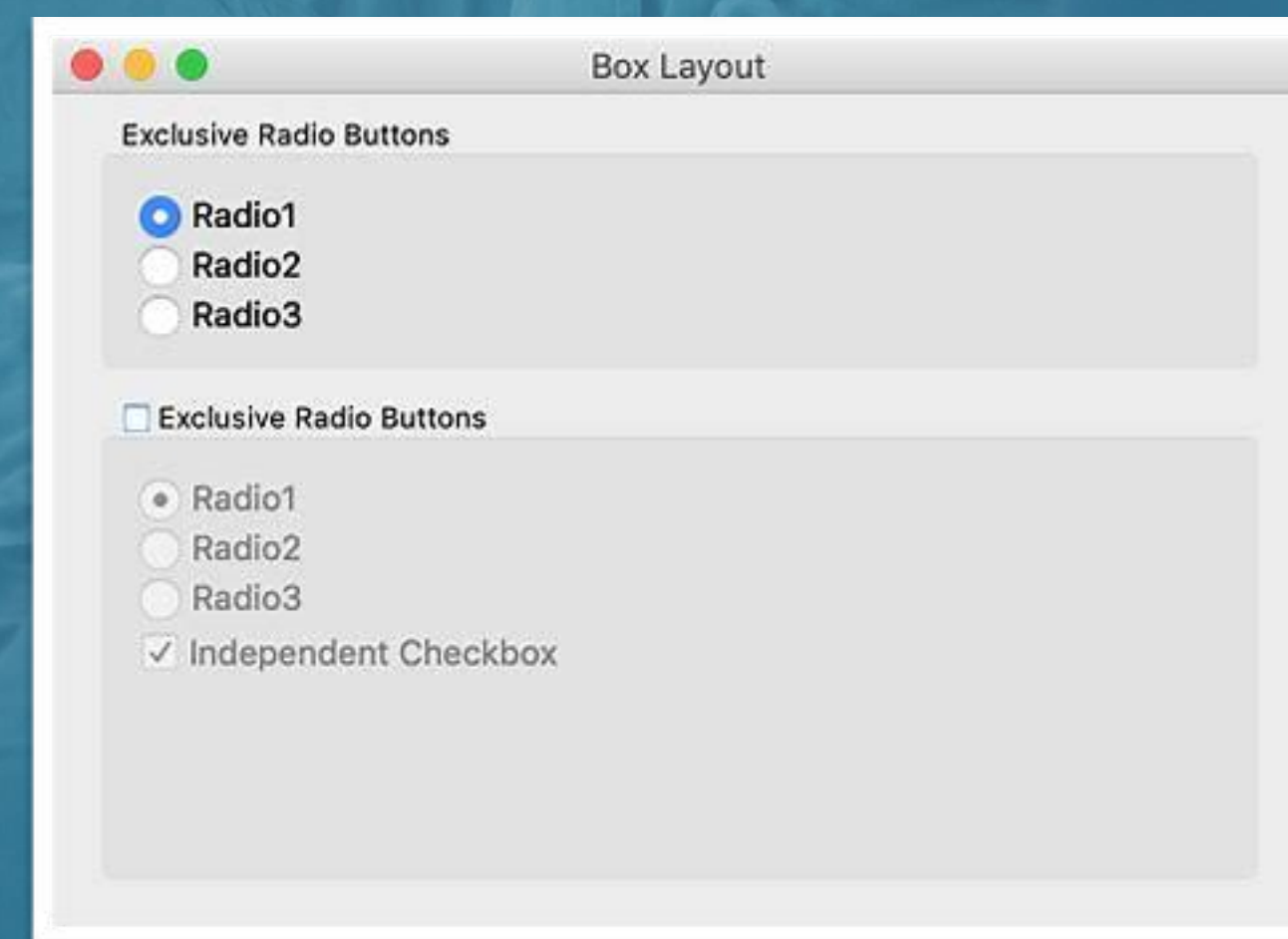
QCheckBox
用于添加选择按钮



QProgressBar
用于添加进度条



QGroupBox
用于将通用模块添加为组



目标实现

登录界面

视频导入

片段选取

视频预览

Part 2

视频分割拼接



视频切割主要目的是从一个较长的视频文件中自动检测和提取独立的场景，并将每个场景保存为单独的视频文件
主要运用库：**moviepy**、**scenedetect**、**os**

切割关键代码

```
15 def find_scenes(video_path):
16     video_manager = VideoManager([video_path])           # 创建视频管理器
17     stats_manager = StatsManager()                       # 创建统计管理器
18     scene_manager = SceneManager(stats_manager)          # 创建场景管理器
19     scene_manager.add_detector(ContentDetector(threshold=30)) # 添加内容检测器
20
21     try:
22
23         video_manager.set_downscale_factor()             # 设置下采样因子
24         video_manager.start()                             # 开始读取视频
25         scene_manager.detect_scenes(frame_source=video_manager) # 检测场景
26
27         scene_list = scene_manager.get_scene_list()       # 获取场景列表
28         print('List of scenes obtained:')
29         for i, scene in enumerate(scene_list):
30             output_file = f'video_{i + 1}.mp4'           # 输出文件名
31             print(f'video_{i + 1}.mp4')
32
33             start_time = scene[0].get_frames() / video_manager.get_framerate() # 计算起始和结束时间（以秒为单位）
34             end_time = scene[1].get_frames() / video_manager.get_framerate()
35
36             output_file = f'video_{i + 1}.mp4' # 输出文件名
37
38             ffmpeg_extract_subclip(video_path, start_time, end_time, targetname=output_file) # 使用ffmpeg提取子剪辑
39
40     finally:
41
42         video_manager.release()                           # 释放视频资源
43
44 if __name__ == '__main__':
45     find_scenes('operator.mp4')
46
```

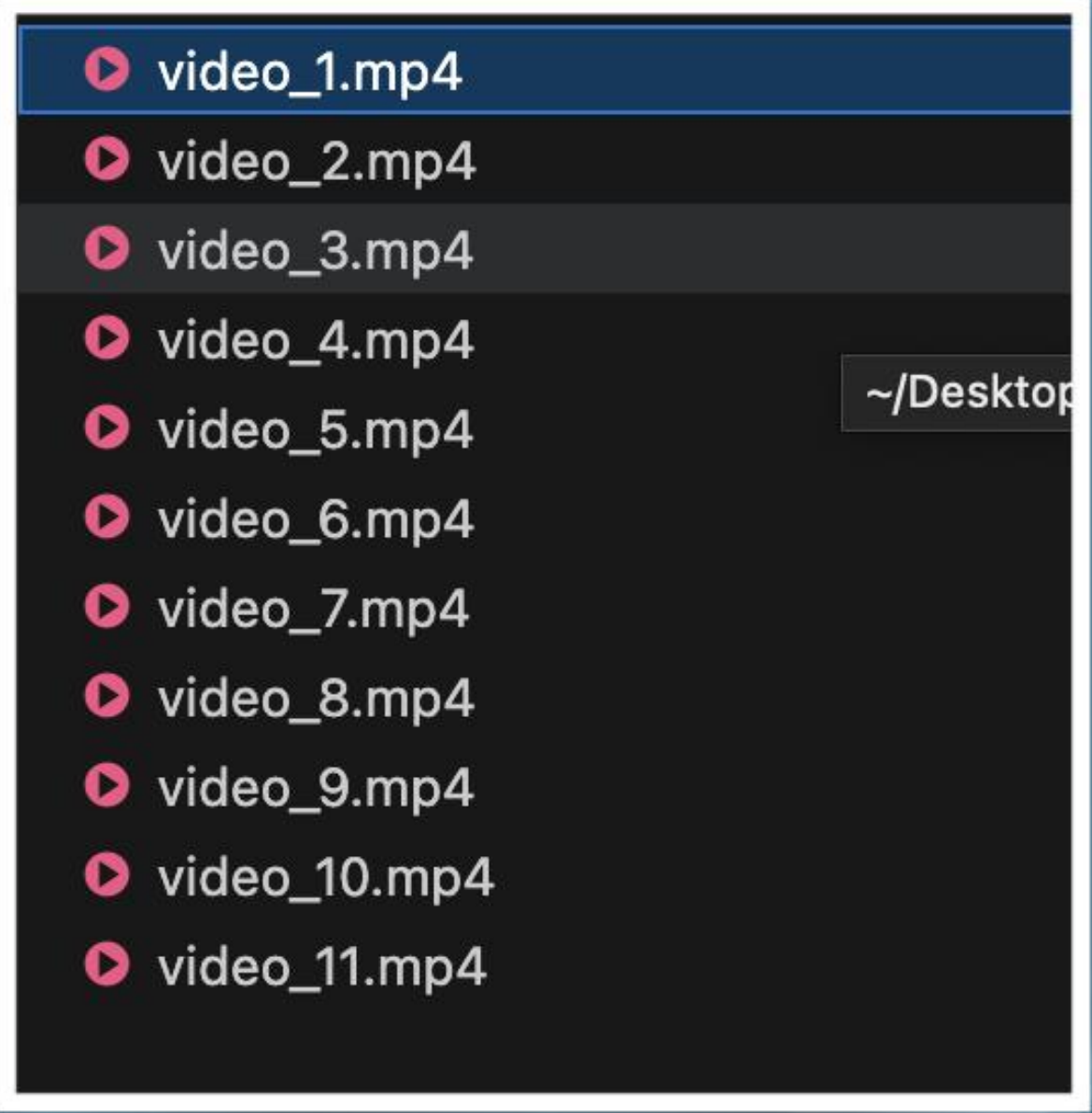
- 核心函数 `find_scenes`。这个函数首先实例化视频管理器、统计管理器和场景管理器。接着，我们添加了内容检测器，这个检测器的阈值设置为30，用于定义场景变换的灵敏度。
- 当我们运行脚本，它会输出每个检测到的场景，并将它们保存为以 `video_` 开头，后跟场景编号的文件。例如，`video_1.mp4`, `video_2.mp4`, 等等。
- 在脚本的最后，我们确保了所有的视频资源都被正确地释放，这是为了防止资源泄漏。

- 这段代码通过循环遍历视频的每一帧，使用场景检测器检测场景，并将每个新场景的时间范围添加到场景列表中

```
video_manager.start()
scene_manager = scenedetect.SceneManager(scene_detector, video_manager)

while True:
    frame = video_manager.get_frame()
    if frame is None:
        break
    scene_manager.detect_frame(frame)
    if scene_manager.is_new_scene():
        scene_list.append(scene_manager.get_scene_list()[-1])
```

- 将视频切割为文件名为video_x.mp4的格式



- ▶ video_1.mp4
- ▶ video_2.mp4
- ▶ video_3.mp4
- ▶ video_4.mp4
- ▶ video_5.mp4
- ▶ video_6.mp4
- ▶ video_7.mp4
- ▶ video_8.mp4
- ▶ video_9.mp4
- ▶ video_10.mp4
- ▶ video_11.mp4



视频拼接部分可以从多个视频片段中选择指定的部分，并将它们合并成一个单一的视频文件
主要运用库：**scenedetect**、**os**

拼接关键代码

```
def merge_selected_scenes(selected_scenes):  
  
    video_clips = []  
    file_names = sorted(os.listdir())  
  
    for file_name in file_names:  
        if file_name.endswith('.mp4') and file_name.startswith('video'):  
            scene_number = int(file_name.split('_')[1].split('.')[0])  
            if scene_number in selected_scenes:  
                video_clip = VideoFileClip(file_name)  
                video_clips.append(video_clip)  
  
    final_video = concatenate_videoclips(video_clips)  
  
    final_video.write_videofile('merged_scenes.mp4', codec='libx264')
```

用于存储视频片段的列表
获取当前目录下的所有文件名

遍历文件名列表
如果文件名以'.mp4'结尾且以'video'开头
获取场景编号
如果场景编号在选中的场景列表中
创建视频片段对象
将视频片段添加到列表中

合并视频片段

将合并后的视频写入文件

- 这个函数接受文件夹路径、选定的场景列表和输出路径作为输入参数。它遍历选定的场景列表，将每个场景的视频文件加载为 VideoFileClip 对象，并将它们添加到 video_clips 列表中。然后，使用 concatenate_videoclips 将所有视频片段合并为一个视频，并将合并后的视频保存到输出路径。


```
for scene in selected_scenes:
    scene_path = os.path.join(folder_path, scene)
    video_clip = VideoFileClip(scene_path)
    video_clips.append(video_clip)
```

```
final_clip = concatenate_videoclips(video_clips)
final_clip.write_videofile(output_path)
```

这段代码通过将文件夹路径和场景文件名拼接起来，加载每个场景的视频文件为 **VideoFileClip** 对象，并将它们添加

这段代码使用 **concatenate_videoclips** 将所有视频片段合并为一个视频，并使用 **write_videofile** 将合并后的视频保存到输出路径。



谢谢观看