远程桌面控制系统使用手册

本文档提供三个不同版本远程桌面控制系统的使用说明。

通用准备

1. **环境配置**:
    - 确保您的计算机上已安装 Python 3.6 或更高版本。
    - 打开命令行工具 (如 Windows上的 CMD 或 PowerShell，Linux/macOS 上的终端)。
    - 进入项目根目录。
    - 安装必要的 Python 库:

      ```
      pip install opencv-python numpy pyautogui pillow pyaudio
      ```

      (注意: `pyaudio` 主要用于 `network_audio_version`，但提前安装亦可。)
2. **网络设置**:
    - 为了获得最佳体验，建议服务端和客户端在**同一局域网 (LAN)** 内。
    - 确保防火墙允许 Python 程序进行网络通信。如果遇到连接问题，可能需要为 `python.exe` 或特定脚本添加入站和出站防火墙规则，允许其使用相应的TCP和UDP端口。
3. **关闭现有远程桌面或冲突程序**:
    - 确保没有其他远程桌面程序 (如Windows远程桌面、TeamViewer等) 正在运行，它们可能会占用必要的端口。

实验一：simple_version 使用手册

`simple_version` 实现了基础的屏幕共享和远程鼠标控制。

文件结构:

- `simple_server.py`: 服务端程序，在被控计算机上运行。
- `simple_client.py`: 客户端程序，在控制计算机上运行。
- `simple_start.bat` (Windows): 一个批处理文件，用于在一台机器上同时启动服务端和客户端进行本地测试。
- `simple_client.bat` (Windows): 一个批处理文件，用于单独启动客户端。

运行步骤:

1. **启动服务端 (在被控计算机上)**:

- 打开命令行工具。
- 导航到 `simple_version` 目录: `cd path/to/your_project/simple_version`
- 运行服务端脚本:

```
python simple_server.py
```

- 服务端启动后，会显示监听的TCP端口 (默认为 `8485` 用于屏幕数据) 和UDP端口 (默认为 `8486` 用于控制指令)。例如:

```
TCP服务器已启动，监听 0.0.0.0:8485
UDP服务器已启动，监听 0.0.0.0:8486
```

2. **启动客户端 (在控制计算机上)**:
   - 打开一个新的命令行工具。
   - 导航到 `simple_version` 目录: `cd path/to/your_project/simple_version`
   - 运行客户端脚本。你需要指定服务端的IP地址。
     - 如果服务端与客户端在同一台机器上测试，IP地址为 `localhost` 或 `127.0.0.1`。
     - 如果服务端在局域网内的另一台机器上，你需要找到该机器的局域网IP地址 (例如，通过在服务端机器上运行 `ipconfig` (Windows) 或 `ifconfig` / `ip addr` (Linux/macOS))。

```
python simple_client.py --host SERVER_IP_ADDRESS
```

     (将 `SERVER_IP_ADDRESS` 替换为实际的服务端IP地址。如果不提供 `--host` 参数，默认连接 `localhost`。)
   - 客户端将弹出一个GUI窗口，标题为 "简单屏幕共享客户端 - 支持鼠标控制"。
3. **进行远程控制**:
   - 客户端窗口成功连接后，会开始显示服务端的屏幕内容。
   - 在客户端窗口的视频区域内:
     - **移动鼠标**: 服务端屏幕上的鼠标指针会相应移动。
     - **单击鼠标左键**: 在服务端屏幕上执行左键单击。
     - **双击鼠标左键**: 在服务端屏幕上执行左键双击。
     - **单击鼠标右键**: 在服务端屏幕上执行右键单击。
     - **拖动鼠标左键**: 在服务端屏幕上执行拖动操作。
   - 客户端界面上有一个 "启用鼠标控制" 的复选框，可以用来临时禁用或启用控制指令的发送。

4. **结束程序**:
    - 关闭客户端的GUI窗口。
    - 在服务端和客户端的命令行窗口中按 `Ctrl+C` 来停止脚本运行。

本地测试 (使用批处理文件 - 仅Windows):

- 双击 `simple_start.bat` 会在同一台机器上启动服务端和客户端，客户端自动连接 `localhost`。这主要用于快速功能验证。

实验二：ros_version 使用手册

`ros_version` 模拟了ROS的节点化设计，将屏幕捕获和远程查看功能分离。

文件结构:

- `screen_capture_node.py`: 屏幕捕获节点 (服务端角色)，在被控计算机上运行。
- `remote_viewer_node.py`: 远程查看器节点 (客户端角色)，在控制计算机上运行。
- `start_ros.bat` (Windows): 一个批处理文件，用于在一台机器上同时启动两个节点进行本地测试。
- `实验手册_模拟ROS使用TCP_IP实现远程桌面的网络传输实验.md`: 此版本的详细实验文档。

运行步骤:

1. **启动屏幕捕获节点 (在被控计算机上)**:
    - 打开命令行工具。
    - 导航到 `ros_version` 目录: `cd path/to/your_project/ros_version`
    - 运行屏幕捕获节点脚本:

      ```
      python screen_capture_node.py
      ```

    - 该节点启动后会监听TCP端口 (默认为 `8485`) 等待查看器节点的连接。控制台会输出日志信息，如:

      ```
      INFO:root:TCP服务器监听端口: 8485
      ```

2. **启动远程查看器节点 (在控制计算机上)**:
    - 打开一个新的命令行工具。
    - 导航到 `ros_version` 目录: `cd path/to/your_project/ros_version`
    - 运行远程查看器节点脚本。你需要指定屏幕捕获节点的IP地址。

      ```
      python remote_viewer_node.py --host SERVER_IP_ADDRESS
      ```

(将 `SERVER_IP_ADDRESS` 替换为屏幕捕获节点所在机器的IP地址。如果不提供 `--host` 参数，默认连接 `localhost`。)
    - 远程查看器节点将弹出一个GUI窗口，标题为 "远程屏幕查看器"，并开始显示捕获节点的屏幕。
    - **注意**: 此版本 `remote_viewer_node.py` 的原始代码主要集中在屏幕查看。远程控制功能（如鼠标点击）可能未在此版本中直接集成或需要参照 `simple_version` 的控制逻辑自行添加或确认。使用手册基于其核心的查看功能。
3. **查看远程屏幕**:
    - 查看器窗口成功连接后，会显示捕获节点的屏幕内容。
4. **结束程序**:
    - 关闭远程查看器节点的GUI窗口 (这将自动停止该节点)。
    - 在屏幕捕获节点的命令行窗口中按 `Ctrl+C` 来停止脚本运行。

本地测试 (使用批处理文件 - 仅Windows):

- 双击 `start_ros.bat` 会在同一台机器上启动屏幕捕获节点和远程查看器节点，查看器自动连接 `localhost`。

实验三：network_audio_version 使用手册

`network_audio_version` 在远程桌面控制的基础上集成了双向音频通话功能。

文件结构:

- `remote_desktop.py`: 主程序，根据命令行参数启动为服务端或客户端。
- `start_server.bat` / `start_server.sh`: 启动服务端的脚本 (Windows / Linux & macOS)。
- `start_client.bat` / `start_client.sh`: 启动客户端的脚本 (Windows / Linux & macOS)。
- `requirements.txt`: 列出了此版本所需的Python库 (主要是添加了 `PyAudio`)。

运行步骤:

确保两台设备在同一个局域网内，或者使用内网穿透，确保两台电脑可以进行网络通信。

1. **安装 `PyAudio` (如果尚未安装)**:
    - `PyAudio` 的安装有时会比较复杂，因为它依赖系统的 PortAudio 库。
    - **Windows**:
        - 可以尝试 `pip install pyaudio`。

        - 如果失败，可能需要先从 Unofficial Windows Binaries for Python Extension Packages 下载对应Python版本和系统架构 (32位/64位) 的 `PyAudio` .whl 文件，然后使用 `pip install PyAudio-0.2.11-cp3x-cp3xm-win_amd64.whl` (文件名请替换为下载的实际文件名) 进行安装。
    - **Linux**:
        - 通常需要先安装 PortAudio 开发库: `sudo apt-get install portaudio19-dev python3-pyaudio` (Debian/Ubuntu) 或类似命令。
        - 然后 `pip install pyaudio`。
    - **macOS**:
        - 可以使用 Homebrew: `brew install portaudio` 然后 `pip install pyaudio`。
2. **启动服务端 (在被控计算机上)**:
    - 打开命令行工具。
    - 导航到 `network_audio_version` 目录: `cd path/to/your_project/network_audio_version`
    - 运行服务端:

      ```
      python remote_desktop.py --mode server
      ```

      或者使用提供的启动脚本:
        - Windows: 双击 `start_server.bat`
        - Linux/macOS: `./start_server.sh` (可能需要先 `chmod +x start_server.sh`)
    - 服务端会启动并监听三个端口:
        - 屏幕传输端口 (默认为 `8485`, TCP)
        - 控制命令端口 (默认为 `8486`, UDP)
        - 音频传输端口 (默认为 `8487`, TCP)
    - 控制台会显示类似信息:

      ```
      屏幕传输服务启动，监听 0.0.0.0:8485
      控制命令服务启动，监听 0.0.0.0:8486
      音频传输服务启动，监听 0.0.0.0:8487
      ```

3. **启动客户端 (在控制计算机上)**:
    - 打开一个新的命令行工具。
    - 导航到 `network_audio_version` 目录: `cd path/to/your_project/network_audio_version`

- 运行客户端，并指定服务端的IP地址:

```
python remote_desktop.py --mode client --host SERVER_IP_ADDRESS
```

(将 `SERVER_IP_ADDRESS` 替换为实际的服务端IP地址。)
或者使用提供的启动脚本 (通常需要编辑脚本内的IP地址或通过参数传入):

  - Windows: `start_client.bat SERVER_IP_ADDRESS` (或者编辑 `start_client.bat` 内的 `TARGET_HOST` 变量)
  - Linux/macOS: `./start_client.sh SERVER_IP_ADDRESS` (或者编辑 `start_client.sh` 内的 `TARGET_HOST` 变量)

- 客户端将弹出一个GUI窗口，并尝试连接服务端的所有三个端口。

4. **进行远程控制与语音通话**:
   - 连接成功后，客户端窗口会显示服务端的屏幕。
   - **屏幕控制**: 与 `simple_version` 类似，可以在视频区域进行鼠标操作。
   - **语音通话**:
     - 确保服务端和客户端的麦克风和扬声器都已正确连接并启用。
     - 双方即可开始语音通话。客户端的声音会传到服务端播放，服务端的声音会传到客户端播放。
     - 客户端界面可能包含 "静音麦克风" 的复选框，用于控制本地麦克风的开启/关闭。
   - 客户端界面上的 "鼠标控制" 复选框可以用来临时禁用或启用控制指令的发送。
5. **结束程序**:
   - 关闭客户端的GUI窗口。
   - 在服务端和客户端的命令行窗口中按 `Ctrl+C` 来停止脚本运行 (如果未使用批处理文件且脚本在前台运行)。如果使用了批处理文件，直接关闭对应的控制台窗口。

通用故障排除提示

- **端口被占用**: `[WinError 10048] 通常每个套接字地址(协议/网络地址/端口)只允许使用一次`。
  - 确保没有其他版本的远程桌面或本项目的其他实例正在运行。
  - 可以使用 `netstat -ano | findstr "PORT_NUMBER"` (Windows) 或 `sudo lsof -i :PORT_NUMBER` (Linux/macOS) 来查找占用特定端口的进程。
  - 在Windows上，可以使用 `taskkill /f /im python.exe` 强制关闭所有Python进程，但请谨慎使用。

- **PyAutoGUI 安全机制**: `PyAutoGUI fail-safe triggered from mouse moving to a corner of the screen.`
    - 代码中通常已设置 `pyautogui.FAILSAFE = False` 来禁用此机制。如果仍然出现，请检查对应代码。
- **音频问题 (network_audio_version)**:
    - **无声或杂音**: 检查麦克风和扬声器是否被系统正确识别和选择为默认设备。检查音量设置。
    - `PyAudio` **错误**: 确保 `PyAudio` 及其依赖 (PortAudio) 已正确安装。
- **连接超时/失败**:
    - 确认服务端IP地址正确无误。
    - 检查网络连接是否通畅 (例如，使用 `ping SERVER_IP_ADDRESS`)。
    - 检查防火墙设置。