AI Lab Scribe

智能电镜实验记录系统

技术设计文档 v0.1

AtomSTEM / Yale University

2026年1月

1. 项目概述

1.1 核心痛点

电镜实验过程中存在两个核心问题：

- **实时记录困境：**操作电镜时双手在调参，眼睛盯着屏幕，没有能力同时记录实验细节。尤其在独自做实验或安静的夜晚，很多关键参数和决策过程会遗漏。
- **培训位置盲区：**培训时听到的是概念（如调high tension），但实际操作需要空间记忆（菜单在哪里）。新学员自己操作时往往找不到对应的UI元素。

与Nion实验室合作时发现，即使是顶级实验室也需要专门配备一个记录员——这说明问题普遍存在且缺乏好的解决方案。

1.2 解决方案

构建一个后台运行的AI系统，同时捕获三个数据流：屏幕内容、键鼠操作、语音对话，然后通过AI处理生成结构化的实验日志。

1.3 三个产品形态

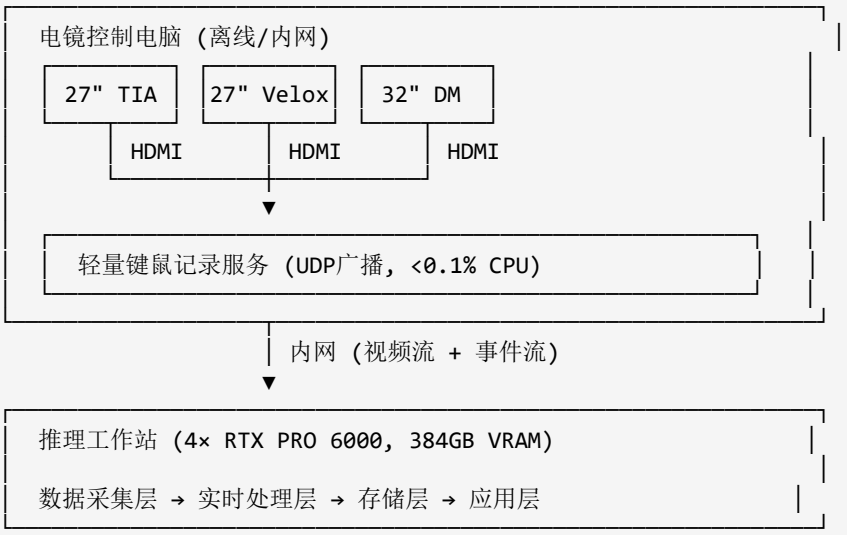
产品	用户	核心价值
Lab Scribe	做实验的人	无需手动记录，实验完自动生成日志
Microscope Academy	新学员	可搜索的操作指南，带真实截图和路径
AtomSTEM Dataset	AI研发团队	人类专家操作电镜的高质量demonstration数据

前两个产品可完全开源以扩大影响力，第三个（RL数据集格式和训练pipeline）作为核心护城河。

2. 系统架构

2.1 整体架构

系统采用推流架构，不在电镜控制电脑上做任何重计算，保证稳定性和数据安全。



2.2 数据采集

数据源	采集方式	说明
屏幕画面	3× HDMI采集卡	零软件入侵，通过视频流传输
键鼠事件	轻量软件 + 图像识别	双管齐下，互为冗余
语音	OWL Labs / 麦克风阵列	支持多人speaker识别

2.3 键鼠事件双管齐下方案

为确保数据完整性，采用软件记录 + 图像识别双重方案：

- **软件记录：**在控制电脑上运行轻量服务，通过UDP广播键鼠事件，获取精确坐标和点击类型
- **图像识别：**通过模板匹配识别屏幕上的光标位置，作为冗余验证和掉线恢复手段

图像识别光标的优势：光标形状本身携带语义信息（箭头=选择，手指=可点击，十字=画ROI，等待=处理中）。

2.4 UI状态机

预先标注电镜软件（TIA/Velox/DM）的UI元素，构建状态机：

```
UI_MAP = {
    "TIA": {
```

```
        "main_window": {
            "imaging_menu": {"bbox": [120, 45, 180, 65], "type": "menu"},
            "stigmator_panel": {"bbox": [20, 200, 150, 400], ...},
        },
        "transitions": {
            ("main_window", "click", "imaging_menu") → "imaging_menu_expanded",
        }
    }
}
```

实时推理时查表即可将原始坐标转换为语义action，如 *click(TIA, Imaging > Stigmator)*。这个action space对后续RL训练至关重要。

3. 硬件配置

3.1 MVP开发环境

在本地4090电脑上开发和测试：

组件	配置	用途
GPU	NVIDIA RTX 4090 (24GB)	本地推理测试
麦克风	Fifine K669B 或 Blue Yeti	语音采集
测试软件	Nion Swift (开源)	模拟电镜操作环境

3.2 生产环境工作站

基于Puget Systems配置，预估价格约$52,700：

组件	配置	说明
平台	Puget Rackstation X141-5U	5U机架式，散热能力强
GPU	4× RTX PRO 6000 Blackwell Max-Q	96GB×4 = 384GB VRAM
CPU	Intel Xeon w7-3565X	24核，工作站级稳定性
RAM	512GB DDR5 ECC	支持长时间运行
存储	4TB NVMe (建议扩至16TB+)	8小时实验约300-400GB
网络	Dual 10G	视频流采集绑绑有余

RTX PRO 6000 Blackwell Max-Q规格

指标	单卡	4卡总计
VRAM	96GB ECC GDDR7	384GB
CUDA Cores	24,064	96,256
Tensor Cores	752	3,008
带宽	1,792 GB/s	—
TDP	300W	1,200W

选择Max-Q版本（300W）而非Workstation版本（600W）的原因：更适合4卡散热，性能仅慢5-14%，但功耗减半。

3.3 VRAM分配策略

```
GPU 0 (96GB) - 实时管线，常驻
├── Whisper large-v3            4GB
├── 实时VLM (Qwen2.5-VL-72B)    ~70GB
```

```
└── 预留                          22GB

GPU 1-2 (192GB) - Agent主模型
├── Agent决策模型                ~150GB
└── 预留                          42GB

GPU 3 (96GB) - 备用/多Agent分析
├── 第二意见模型                  ~60GB
└── 预留                          36GB

总预留: ~100GB (26%) → 确保稳定性，不会OOM
```

3.4 注意事项

- **存储扩展**：建议扩至16-32TB，或配置NAS用于长期数据存储
- **噪音处理**：5U机架式系统噪音较大，建议放置在隔壁房间或机房，通过10G网线连接
- **UPS配置**：总功耗约1800-2000W，建议配置3000VA以上UPS

4. 软件技术栈

4.1 数据采集层

功能	工具	说明
屏幕录制	OBS / FFmpeg	支持多路HDMI采集
键鼠记录	pynput + pygetwindow	获取坐标、事件、活动窗口
语音录制	PyAudio / sounddevice	与视频同步录制

4.2 实时处理层

功能	模型/工具	资源占用
语音转录	faster-whisper (large-v3)	~4GB VRAM
Speaker识别	pyannote-audio	~2GB VRAM
UI识别	Qwen2.5-VL-7B/72B	16-70GB VRAM
OCR	PaddleOCR	~1GB VRAM
光标检测	OpenCV模板匹配	CPU

4.3 中英文混合语音识别

电镜操作场景的语音特点是中文句子中嵌入英文专业术语：

「我们现在要调一下stigmator，你看这个EELS spectrum这里有个artifact」

Whisper large-v3对这种模式处理得较好。可通过prompt提示（这是中英混合的科学讨论）进一步提升准确率，或使用后处理LLM修正。

4.4 多Agent分析架构

利用384GB VRAM，可同时运行多个模型协作分析：

```
实时层 (常驻):
├── Qwen2.5-VL-7B    快速UI识别 + 裁剪决策
└── Whisper Large    语音转录

分析层 (周期性触发，每5-10分钟):
├── Agent A (操作专家)   总结操作序列
├── Agent B (科学专家)   解读实验意图
├── Agent C (质量审核)   检查准确性
└── Synthesizer          整合输出最终日志
```

多Agent协作的优势：交叉验证减少幻觉、不同模型专注不同方面、可解释性更强。

5. 数据结构设计

5.1 增量处理Segment

采用流式增量处理——边录边处理，每5分钟生成一个segment：

```
{
    "segment_id": "2024-01-15_14-30-00",
    "time_range": ["14:30:00", "14:35:00"],

    "transcript": [
        {"time": "14:30:12", "speaker": "A", "text": "我们现在要调stigmator"},
        {"time": "14:31:45", "speaker": "B", "text": "Y方向再调一点"}
    ],

    "screen_events": [
        {"time": "14:30:15", "monitor": "TIA", "event": "menu_open",
         "path": "Imaging > Stigmator"},
        {"time": "14:30:18", "monitor": "TIA", "event": "param_change",
         "param": "Stig Y", "old": "-2.3", "new": "+1.1"}
    ],

    "keyframes": ["frames/14-30-15.jpg", "frames/14-30-18.jpg"],

    "ai_summary": "调整了stigmator Y从-2.3到+1.1，Speaker B确认效果改善"
}
```

5.2 智能裁剪策略

不是简单地以鼠标为中心裁剪固定区域，而是根据操作类型决定裁剪范围：

操作类型	裁剪策略
点击菜单项	包含整个菜单 + 标题栏
拖动滑块调参	包含参数名 + 数值显示 + 滑块
在图像上选ROI	包含足够的图像上下文
切换软件窗口	整个窗口截图

这样存储量可降低10-20倍，同时信息密度更高。

5.3 存储估算

数据类型	参数	8小时大小
2×27"显示器	1080p, 10fps, H.265	~8GB × 2
1×32"显示器	1440p, 10fps, H.265	~6GB
音频	48kHz stereo	~0.3GB
结构化日志	JSON + 裁剪截图	~2GB

总计	—	~25GB (优化后)

注：降到10fps（电镜UI变化不快）可大幅减少存储需求。原始全帧率录制约340GB/8小时。

6. MVP开发计划

6.1 MVP目标

在本地4090电脑上，使用Nion Swift（开源电镜软件）作为测试环境，验证核心pipeline。

6.2 开发阶段

Phase 1: 能录能看 (1-2周)

- 配置OBS/FFmpeg屏幕录制
- 配置麦克风音频录制
- 实现时间戳同步
- **交付物**：可回放的屏幕视频 + 同步音频

Phase 2: 能听能记 (2-3周)

- 部署faster-whisper实时转录
- 测试中英文混合识别质量
- 实现pynput键鼠事件记录
- **交付物**：带时间戳的转录文本 + 键鼠事件日志

Phase 3: 能看能懂 (3-4周)

- 部署Qwen2.5-VL进行UI识别
- 标注Nion Swift基础UI元素
- 实现智能裁剪和参数变化检测
- **交付物**：带语义标注的截图序列

Phase 4: 能写能用 (3-4周)

- 整合所有数据流生成结构化日志
- 实现Markdown日志生成器
- 完成端到端demo session
- **交付物**：自动生成的完整实验日志

7. 代码示例

7.1 轻量键鼠记录服务

运行在电镜控制电脑上，通过UDP广播事件：

```
# mouse_broadcast.py
from pynput import mouse, keyboard
import socket, json, time
import win32gui

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
BROADCAST_ADDR = ("255.255.255.255", 9999)

def get_active_window():
    return win32gui.GetWindowText(win32gui.GetForegroundWindow())

def send_event(event_type, **data):
    event = {
        "ts": time.time(),
        "type": event_type,
        "window": get_active_window(),
        **data
    }
    sock.sendto(json.dumps(event).encode(), BROADCAST_ADDR)

def on_click(x, y, button, pressed):
    send_event("click", x=x, y=y, button=str(button), pressed=pressed)

def on_move(x, y):
    send_event("move", x=x, y=y)

def on_key(key, pressed):
    send_event("key", key=str(key), pressed=pressed)

mouse.Listener(on_click=on_click, on_move=on_move).start()
keyboard.Listener(
    on_press=lambda k: on_key(k, True),
    on_release=lambda k: on_key(k, False)
).start()

while True:
    time.sleep(1)
```

资源占用：<0.1% CPU, <10MB内存，不写本地磁盘。

7.2 光标图像识别

```
# cursor_detect.py
import cv2

cursor_templates = {
    "arrow": cv2.imread("cursors/arrow.png"),
```

```
    "hand": cv2.imread("cursors/hand.png"),
    "crosshair": cv2.imread("cursors/crosshair.png"),
    "wait": cv2.imread("cursors/wait.png"),
}

def find_cursor(frame):
    for cursor_type, template in cursor_templates.items():
        result = cv2.matchTemplate(frame, template, cv2.TM_CCOEFF_NORMED)
        _, confidence, _, location = cv2.minMaxLoc(result)
        if confidence > 0.8:
            return {
                "type": cursor_type,
                "x": location[0],
                "y": location[1],
                "confidence": confidence
            }
    return None

# 光标类型语义:
# arrow = 普通选择
# hand = 可点击元素
# crosshair = 在图像上选点/画ROI
# wait = 系统处理中
```

7.3 事件接收与融合

```
# event_receiver.py
import socket, json

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind(("0.0.0.0", 9999))

def receive_events():
    while True:
        data, addr = sock.recvfrom(4096)
        event = json.loads(data.decode())

        # 融合软件事件和图像识别结果
        if event["type"] == "click":
            cursor_from_image = find_cursor(current_frame)

            if cursor_from_image:
                # 验证坐标一致性
                dx = abs(event["x"] - cursor_from_image["x"])
                dy = abs(event["y"] - cursor_from_image["y"])

                event["confidence"] = "high" if dx < 10 and dy < 10 else "medium"
                event["cursor_type"] = cursor_from_image["type"]

        yield event
```

8. 未来扩展

8.1 与AI Agent的闭环

记录系统最终将成为AI Agent训练的数据飞轮：

```
记录系统观察人类操作
        |
        ▼
积累"人类怎么操作电镜软件"的知识
        |
        ▼
训练/指导AI Agent理解UI
        |
        ▼
Agent能够自主操作
        |
        ▼
Agent的操作也被记录系统捕获
        |
        ▼
形成完整的实验日志（人+AI混合操作）
```

8.2 数据集价值

积累的数据可用于：

- **Imitation Learning**：从人类专家操作中学习
- **Reinforcement Learning**：基于实验结果的奖励信号
- **UI理解模型微调**：针对电镜软件特化的VLM

8.3 跨设备扩展

当前设计针对赛默飞系统（TIA/Velox/DM），但架构可扩展到：

- Nion电镜系统
- JEOL电镜系统
- 其他科学仪器（SEM、AFM等）

核心工作是针对每个软件构建UI Map和状态机，数据采集和处理架构可复用。

附录A：采购清单

MVP阶段

物品	型号建议	预估价格
麦克风	Fifine K669B / Blue Yeti	$30-130
总计	-	$30-130

生产环境

物品	型号建议	预估价格
推理工作站	Puget X141-5U (4x RTX PRO 6000)	$52,700
存储扩展	额外12TB NVMe	$1,500
UPS	3000VA以上	$800
HDMI采集卡	3x 4K采集卡	$300
麦克风阵列	OWL Labs Meeting Owl	$1,000
总计	-	约$56,300

附录B：软件依赖

```
# Python环境
pip install pynput pygetwindow pyaudio faster-whisper
pip install torch torchvision torchaudio
pip install transformers accelerate
pip install opencv-python paddleocr
pip install pyannote-audio

# 系统工具
# OBS Studio (屏幕录制)
# FFmpeg (视频处理)

# 模型下载
# faster-whisper: large-v3
# Qwen2.5-VL: 7B或72B根据硬件选择
```

— 文档结束 —

AI Lab Scribe v0.1
AtomSTEM × Yale University