Software Engineering and Applications 软件工程与应用, 2025, 14(2), 245-260
Published Online April 2025 in Hans. https://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2025.142023

Hans汉斯

利用Python批量处理Krpano全景漫游文件的方法

邓　鹏[1]，陈敏铭[2]，周　叶[2]

[1]中国南方电网云南电网有限责任公司，云南　昆明
[2]江苏金沙地理信息股份有限公司，江苏　常州

收稿日期：2025年1月15日；录用日期：2025年4月3日；发布日期：2025年4月14日

摘　要

针对Krpano全景漫游软件制作软件的生产文件，本文利用Python开发预处理insta文件、VR处理完整版、制作VR内部指向、制作层VR (from标准层)和合并层为幢VR五个模块，对全景图切片、tour.xml、vtourskin.xml标记语言文件进行自动化批量场景和片段修正，制作了以项目和建筑幢为单位的全景漫游。文中对场景切片文件夹、场景内容收集、标准层文件复制等难点列出了脚本内容。

关键词

Krpano，Python，全景漫游，自动化批处理

Method for Batch Processing Krpano Panorama Tour Files Using Python

Peng Deng[1], Minming Chen[2], Ye Zhou[2]

[1]Yunnan Power Grid Co., Ltd., China Southern Power Grid, Kunming Yunnan
[2]Jiangsu Jingsha Geographic Information Co., Ltd., Changzhou Jiangsu

Received: Jan. 15th, 2025; accepted: Apr. 3rd, 2025; published: Apr. 14th, 2025

Abstract

This paper discusses the production files generated by the Krpano panorama tour software. It utilizes Python to develop five modules: preprocessing of insta files, processing complete VR versions, creating internal directional VR, generating layer VR (from standard layers), and merging layers into building VRs. The method automates the batch correction of scenes and segments in panorama image slices, tour.xml, and vtourskin.xml markup language files, resulting in panorama tours

文章引用: 邓鹏, 陈敏铭, 周叶. 利用 Python 批量处理 Krpano 全景漫游文件的方法[J]. 软件工程与应用, 2025, 14(2): 245-260. DOI: 10.12677/sea.2025.142023

organized by project and building units. The paper also includes script content addressing challenges such as managing scene slice folders, collecting scene content, and duplicating standard layer files.

Keywords

Krpano, Python, Panorama Tours, Automated Batch Processing

1. 引言

1.1. Krpano 软件结构

Krpano 由奥地利公司 krpano Gesellschaft mbH 开发，是一款全景漫游制作软件和工具[1]，它兼容 HTML5 和 Flash，支持 WebGL 下的 WebVR 展示。使用 XML 标记语言编写全景漫游可以开发出高度定制化的项目，也可以利用 krpano 工具开发在线全景制作及展示平台[2]。

Krpano 在 Windows 操作系统下有制作漫游批处理可执行文件 MAKE VTOUR Droplet.exe；全景格式转换可执行文件 Convert SPHERE CUBE Droplet.exe；krpano 测试服务器 krpano Testing Server.exe；可视化操作界面 krpano Tools.exe 和命令行操作界面 krpanotools.exe。

全景漫游制作结束后生成一个 vtour 文件夹[3]；panos 文件夹存放上传全景切片图；plugins 文件夹存放插件；skin 文件夹存放与皮肤有关的文件和图片，包括底部操作栏；tour.html 是全景入口文件；tour.jskrpano 是 js 库；tour.xml 是全景漫游文件。

在 html 文件里，调用了 embedpano 方法，并且传入主 XML 和目标 id 等作为参数[4]，embedpano 是 tour.js 里的一个方法，通过 XML 文件里的标签生成全景，目标 id 元素来作为全景展示的容器。

1.2. XML 文件结构

全景漫游制作结束后在 vtour 文件里生成一个 tour.xml，在 vtour 的 skin 子文件夹生成一个 vtourskin.xml。

tour.xml 文件用于定义虚拟旅游或全景视图项目中的场景、交互和设置。tour.xml 文件主要内容[5]：

基本信息：项目的名称、对旅游项目的简要描述、创建项目的个人或团队、文件的版本号以及创建或修改的日期。

场景定义：多个场景的定义，包括每个场景的标识符、名称和类型；指向场景图像或视频文件的路径；定义场景的初始视角，如方位角、俯仰角等；场景参数，如缩放级别、场景持续时间等。

交互元素：在场景中定义可交互区域，通常用于跳转到其他场景或触发信息框等；热点在场景中的坐标或位置；指向其他场景或外部资源的链接；热点的描述或提示信息。

媒体内容：指向音频和视频文件的路径，以提供多媒体丰富的体验；附加图像文件的引用，用于增强体验。

过渡效果：定义在场景之间切换时使用的视觉效果；用户与热点或界面元素互动时可能发生的动画。

自定义脚本：定义用户与场景或元素交互时触发的行为或脚本；自定义功能的定义，用于扩展项目的交互性。

导航信息：定义可供用户使用的菜单或工具条，方便在不同场景之间导航；提供一个全局视图，帮助用户了解虚拟旅游的结构。

设置和配置：定义用户可以调整的设置，如音量、语言等；定义整个旅游项目的一些全局属性。

vtourskin.xml 文件则包括：

基本信息：项目的标题、作者或创建者信息；版本号和文件的创建日期。

皮肤和界面设置：定义在虚拟旅游中使用的图形用户界面(GUI)元素，如按钮、菜单、信息框等；颜色、字体、大小和其他样式设置。

场景配置：指向不同全景图像的路径或文件引用；场景之间的切换方式和交互方式，如，点击热点；场景的初始视角和视图设置。

交互元素：热点定义，用户在场景中可以点击的区域，通常与信息或其他场景链接相关；动画和过渡效果，场景切换或元素交互时的动画效果。

多媒体内容：可能包含音频或视频元素的引用，用于增强用户体验；文本描述或标签，提供有关特定场景或对象的信息。

自定义脚本：支持用户自定义的脚本或函数，用于更复杂的交互或功能。

1.3. 存在的问题

Krpano 提供了一系列软件工具加工全景图像和全景漫游，例如对全景图像拼接用了"%~dp0\krpanotools.exe" makepano "%~dp0\templates\vtour-normal.config" %*为核心内容的批处理，从而生成了 vtour 及其子文件夹 skin 相配套的单场景配置文件 vtour.xml 和全局配置文件 vtourskin.xml。

对有一定工程量的项目而言，全景图制作、场景配置文件的修改全部人工操作，存在内业技术人员要求多、手工处理部分的工作量大、文件修改的内容容易出错，对项目的推进存在效益低下的问题。利用 Python 编写脚本，通过编制修改要素、场景等 Excel 对照表，对全景图、配置文件进行批量生成和修改，有利于工程的顺利推进。

2. Krpano 制作全景漫游

2.1. 全景图像

2.1.1. 获取全景图像

使用 Insta360、GoPro 等全景相机拍摄全景图像，如图 1。

Figure 1. Panorama image captured by Insta360
图 1. Insta360 拍摄的全景图像

使用 Insta360 Stitcher 拼接软件，将包含 1 个工程文件、1 个陀螺仪数据文件、6 个独立镜头拍摄的原文件和 1 个预览照片文件拼接成一张全景图，如图 2。

Figure 2. Panorama stitched from multiple photos
图 2. 由多张照片拼接的全景图

2.1.2. 生成多分辨率图像

使用 KRpano 工具生成不同分辨率的快照，以提高加载速度和性能。

使用命令行工具生成级别图像：krpano_tools.exe makepano your_image.jpg，不同分辨度的快照图如图 3。

Figure 3. Snapshots at various resolutions
图 3. 不同分辨率的快照

2.2. 创建 XML 配置文件

创建一个 XML 文件来配置全景场景、视角和其他设置。

2.2.1. XML 文件基本配置

KRpano XML 文件基本配置内容可以设置为：

```
<krpano version="1.20">
    <scene name="panorama1">
        <image>
            <cube url="pano_%s.jpg" />   <!--  立方体全景图  -->
        </image>
        <view>
            <fov>100</fov> <!--  视场  -->
            <hlookat>0</hlookat> <!--  水平位置  -->
            <vlookat>0</vlookat> <!--  垂直位置  -->
```

```
        </view>
    </scene>
</krpano>
```

2.2.2. 添加多个场景

对于有多个全景图像，可以在同一文件中添加多个场景：

```
<krpano version="1.20">
    <scene name="scene1">
        <image>
            <cube url="scene1_%s.jpg" />
        </image>
    </scene>
    <scene name="scene2">
        <image>
            <cube url="scene2_%s.jpg" />
        </image>
    </scene>
    <!-- 更多场景 -->
</krpano>
```

2.3. 修改 XML 文件

2.3.1. 修改 tour.xml 文件

修改的主要内容有：

将 title 默认的“Virtual Tour”更改为工程项目名称。

将 webvr="true"改为 webvr="false"。

将 loadingtext="loading..."改为 loadingtext="加载中..."。

对热点文字显示、自动旋转、场景设置等进行配置。在文件的最后</scene>标签与</krpano>标签之间插入：

```
    <!--热点显示文字 -->
    <action name="add_all_the_time_tooltip">
        txtadd(tooltipname, 'tooltip_', get(name));
        addplugin(get(tooltipname));
        txtadd(plugin[get(tooltipname)].parent, 'hotspot[', get(name), ']');
……
        set(plugin[get(tooltipname)].enabled,false);
    </action>
    <!--自动旋转 -->
    <autorotate enabled="true" waittime="0.5" accel="1.0" speed="5"
        horizon="0.0" tofov="120" />
    <!-- 移除加载中文字 (覆盖 tour.xml 的设置) -->
    <skin_settings loadingtext="" />
```

```
    <!-- 旋转一圈后加载下一个场景 -->
<events onautorotateoneround="blend_to_next_scene_during_autorotation()" />
<!-- 修改场景载入参数增加 KEEPMOVING 实现旋转过程中的无缝融合 -->
<action name="blend_to_next_scene_during_autorotation">
    <!-- 保存当前的 loadscene 设置 -->
    push(skin_settings.loadscene_flags);
    push(skin_settings.loadscene_blend_next);
    <!-- 为无缝融合设置新的设置 -->
    set(skin_settings.loadscene_flags, MERGE|KEEPVIEW|KEEPMOVING|NOPREVIEW);
    set(skin_settings.loadscene_blend_next, 'SLIDEBLEND(1.5, 0, 0.75, linear)');
    <!-- 载入下一个场景 -->
    skin_nextscene_loop(+1);
    <!-- 恢复之前的设置 -->
    pop(skin_settings.loadscene_blend_next);
    pop(skin_settings.loadscene_flags);
</action>
```

2.3.2. 修改 Vtourskin.xml 文件

修改的主要内容有：

将 onloaded="skin_hotspotstyle_setup();"修改为 onloaded="add_all_the_time_tooltip();"

对"skin_addthumbs"方法进行替换，替换前：

```
<action name="skin_addthumbs" scope="local">
        if(skin_settings.thumbs == false,
            set(layer[skin_btn_thumbs].visible,false);
……
            if(scene.count == 1,
                set(layer[skin_thumbs].align, 'lefttop');
              );
            );
    </action>
```

替换后：

```
<action name="skin_addthumbs">
        if(skin_settings.thumbs == false,
            set(layer[skin_btn_thumbs].visible,false);
……
set(layer[get(thumbtitlname)].css,"text-align:center; color:#ffffff; font-family: 宋体; font-weight: normal; font-size:11px; "); set(layer[get(thumbtitlname)].textshadow, 2);
            <!-- END:缩略图标题 -->
              );
            );
        </action>
```

2.3.3. 添加互动元素

利用 krpano Tools.exe 载入 tour.xml 即可浏览全景漫游，点击 Add hotspots 可以添加一个热点让用户在不同场景之间跳转，图 4 点击箭头配置场景热点，实现场景的跳转，也可以编辑热点文字。

Figure 4. Scene hot spot configuration diagram
图 4. 场景热点配置图

配置完成后在 tour.xml 文件里的表达为：<hotspot name="hotspot1" style="hotspot_style" ath="30" atv="10" onclick="skin_loadscene('scene2', 花谷奇缘.jpg);" />。

2.4. 嵌入全景漫游到网页

创建一个 HTML 文件，将全景漫游嵌入到网页中，全景漫游如图 4。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>KRpano Panorama</title>
    <script src="krpano.js"></script> <!-- 引入 KRpano 的 JavaScript 文件 -->
</head>
<body>
    <div id="pano"></div> <!-- 全景图的容器 -->
    <script>
        embedpano({
            swf: "krpano.swf", // SWF 文件路径
            xml: "your_pano.xml", // XML 配置文件路径
            target: "pano",
            html5: "prefer",
            passQueryParameters: true
```

```
        });
    </script>
</body>
</html>
```

3. 利用 Python 对文件批处理

通过上述分析，krpano 制作全景漫游自身有一套完整的全景图像和漫游制作工具，制作过程中，技术人员利用相关工具，逐个环节进行图片导入、复制、文件修改可以很顺利制作出全景漫游成果。

本文的技术关键是针对所有人工环节，编制要素对照表，利用 Python 编写脚本对图像、文件进行批量处理和修改。

3.1. 技术路线

利用 insta 软件批量自动导出各场景拼接后全景图像，编制文件名与实际场景拍摄点全称的“对照表.xlsx”文件。Python 脚本利用“对照表.xlsx”自动建立与文件名同名的文件夹，将全景图像文件拷贝到对应文件夹里。

脚本自动调用 Krpanostools 执行文件，利用模板对全景图像进行切片。对 tour.xml 和 vtourshin.xml 文件所涉内容进行批量修改。

对项目为单位的经修改后全景图像的相应 xml 文件的热点文字、跳转等，编制“内部指向.xlsx”，利用相关前后场景图像名称、角度等对相关 xml 文件所在位置进行批量修改，技术路线如图 5。

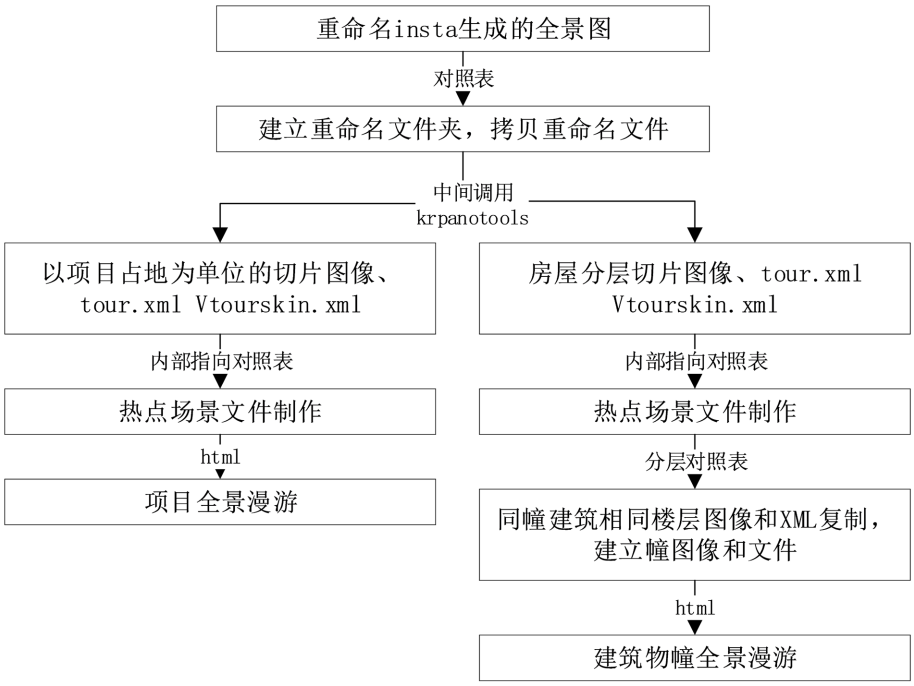

Figure 5. Technical route for batch processing Krpano panorama tour files using Python scripts
图 5. Python 脚本批量处理 Krpano 全景漫游文件技术路线

根据技术路线，利用 Python 开发预处理 insta 文件、VR 处理完整版、制作 VR 内部指向、制作层 VR (from 标准层)和合并层为幢 VR 五个模块，功能模块如图 6。

Figure 6. Python functional modules for creating panorama tours
图 6. 制作全景漫游 Python 功能模块

3.2. 预处理全景图

利用预处理 insta 文件模块，选择存放拼接后的自动编号全景 jpg 文件所在文件夹，生成 ALLfolds.xlsx 文件，文件格式包括项目名称、拍摄顺序号、文件夹名、全景图名，将文件夹内所有全景 jpg 图像文件进行文件名和扩展名分离处理，写入 ALLfolds.xlsx 里。利用文件夹名批量生成同名文件夹，同时，将同名 jpg 文件拷贝到同名文件夹里。

预处理结束后，对 ALLfolds.xlsx 文件进行编辑，增加图片顺序、现文件名(场景名称)，完善对应场景的信息，另存为对照表.xlsx，信息如表 1。

Table 1. Information content of the panorama JPG image file comparison table
表 1. 全景 jpg 图像文件对照表信息内容

项目名称	拍摄顺序	文件夹名称	原文件名	图片顺序	现文件名
庆康加速产业园	1	PIC 20241017142413	PIC 20241017 142413.jpg	A1	A1 产业园 2 幢东进出口
庆康加速产业园	2	PIC 20241017 142516	PIC 20241017 142516.jpg	A2	A2 产业园 2 幢东进出口
庆康加速产业园	3	PIC 20241017142621	PIC 20241017 142621.jPg	A3	A3_2 幢东南角一楼进出口
庆康加速产业园	4	PIC 20241017 143018	PIC 20241017143018.jpg	A4	A4_2 幢三楼东南角电梯口
庆康加速产业园	5	PIC 20241017 143109	PIC 20241017 143109.jPg	A5	A5 2 幢三楼
庆康加速产业园	6	PIC 20241017 143211	PIC 20241017 143211.ipg	A6	A6 2 幢三楼

3.3. 全景图切片和 XML 处理

利用 VR 处理完整版模块和对照表里的场景名称信息，对全景图进行切片、重命名，对 tour.xml 和 Vtourskin.xml 内容进行修改。在工程实施过程中，制作全景漫游一般有二种类型，一类称为项目漫游，是对地理空间基本在一个平面的工程项目周边和内部进行多场景拍摄；二类称为整幢建筑物漫游，是由分层组成的建筑物幢，在此情况下，对楼层相同的室内只拍摄一次标准层，在制作整幢建筑物漫游时，对标准层进行复制合并。

3.3.1. 全景图切片

VR 处理完整版模块在处理全景图前，在选择存放有全景图的一级文件夹里建立“全景 jVR”的文件夹，将所有的重命名全景图拷贝到该文件夹里。调用 krpanotools.exe 及其模板，对全景图进行切片，切片后的文件存放在全景 jVR\vtour\panos\场景名的文件夹里。

Python 调用 krpanotools.exe 的方法首先获取当前脚本目录和工具路径，script_path = os.path.dirname(os.path.abspath(__file__)，config_file_path = os.path.join(script_path, 'templates', 'vtour-normal.config')。

运行 krpanotools 处理每个文件的方法为 command = [krpanotools_path, 'makepano', config_file_path,

image_path]。Jpg 文件路径则为 image_path = os.path.join(image_folder, image_file)。

Python 调用 krpanotools.exe 对每个场景图片进行切片时，都会在 vtour 文件夹里生成一个 tour.xml 文件，而且，后者会覆盖前者。脚本设计时，将每个场景生成的 tour. xml 存放到对应的文件夹里，重命名为 tour.txt。

3.3.2. 处理 tour.txt 文件

处理 tour.txt 时，将第一个场景子文件夹里的 tour. txt 拷贝到 vtour 文件夹里，同时，建立一个 collect_scene_contents(base_folder)函数，用于收集标签<scene name= ".*?">.*?</scene>的所有内容，然后对 tour.txt 标签内容进行替换，最终形成以项目为单位的 tour.txt。

利用对照表第一列项目名称替换 Virtual Tour。对热点文字显示、自动旋转、场景设置等则在文件的最后</scene>标签与</krpano>标签之间插入：

```
<!--热点显示文字 -->
<action name="add_all_the_time_tooltip">
    txtadd(tooltipname, 'tooltip_', get(name));
……
</action>
```

利用制作 VR 内部指向模块，依据 VR 内部指向编制表.xlsx，第一列场景名称和第四列热点指向场景名称、场景方位角、俯仰角等的合集如表 2，对 tour.txt 相关要素进行修改。

Table 2. Content of the VR internal directional compilation table
表 2. VR 内部指向编制表内容

场景名称	方位角	俯仰角	热点要素
A1_产业园 2 幢东进出口	123.509	16.458	<hotspot name="spot1" style="skin_hotspotstyle" ath="123.509" atv="16.458" linkedscene="scene_A2_____2_____" />
A2_产业园 2 幢东进出口	168.433、117.717、77.183、77.183、−70.365	5.662、7.076、11.595、16.854	<hotspot name="spot1" style="skin_hotspotstyle" ath="-168.433" atv="5.662" linkedscene="scene_B10_2_____" /> <hotspot name="spot2" style="skin_hotspotstyle" ath="117.717" atv="7.076" linkedscene="scene_A3_2_____" /> <hotspot name="spot3" style="skin_hotspotstyle" ath="77.183" atv="11.595" linkedscene="scene_B21_2_____" /> <hotspot name="spot4" style="skin_hotspotstyle" ath="-70.365" atv="16.854" linkedscene="scene_A1_____2_____" />

Python 脚本设计重点是对场景的读取与替换，利用<scene> 标签进行缓存处理，内容片段：

```
if '<scene name' in line:
    scene_buffer = [line]  # 重置缓存，开始新场景
elif '</scene>' in line and scene_buffer:  # 结束场景
    scene_buffer.append(line)  # 加入结束标签
    # 获取场景 name 和 title 之间的内容
    scene_title = ''.join(scene_buffer).split('title=')[1].split('"')[1]
                # 匹配当前场景标题
    for first_col_value, fourth_col_value in data:
```

```
                    if first_col_value == scene_title:
                        new_lines.insert(-1, f"\t\t{fourth_col_value}\n")  # 在 </scene> 前插入
                        break  # 只需要找到第一个匹配，退出循环
                # 重建场景缓存
                scene_buffer = []  # 清空场景缓冲区
```

处理后的 tour.txt 场景内容另存到 tour_modified.xml 文件里，内容片段：

```
<scene name="scene_A1__________2_________________" title="A1_产业园 2 幢东进出口" onstart="" thumburl="panos/A1__________2_________________.tiles/thumb.jpg" lat="" lng="" heading="">
        <view hlookat="0.0" vlookat="0.0" fovtype="MFOV" fov="120" maxpixelzoom="2.0" fovmin="70" fovmax="140" limitview="auto" />
        <preview url="panos/A1__________2_________________.tiles/preview.jpg" />
        <image>
            <cube url="panos/A1__________2_________________.tiles/pano_%s.jpg" />
        </image>
        <hotspot name="spot1" style="skin_hotspotstyle" ath="123.509" atv="16.458" linkedscene="scene_A2__________2_________________" />
    </scene>
<scene name="scene_A2__________2_________________" title="A2_产业园 2 幢东进出口" onstart="" thumburl="panos/A2__________2_________________.tiles/thumb.jpg" lat="" lng="" heading="">
        <view hlookat="0.0" vlookat="0.0" fovtype="MFOV" fov="120" maxpixelzoom="2.0" fovmin="70" fovmax="140" limitview="auto" />
        <preview url="panos/A2__________2_________________.tiles/preview.jpg" />
        <image>
            <cube url="panos/A2__________2_________________.tiles/pano_%s.jpg" />
        </image>
        <hotspot name="spot1" style="skin_hotspotstyle" ath="-168.433" atv="5.662" linkedscene="scene_B10_2_____________________________" />
<hotspot name="spot2" style="skin_hotspotstyle" ath="117.717" atv="7.076" linkedscene="scene_A3_2_____________________________" />
<hotspot name="spot3" style="skin_hotspotstyle" ath="77.183" atv="11.595" linkedscene="scene_B21_2______" />
<hotspot name="spot4" style="skin_hotspotstyle" ath="-70.365" atv="16.854" linkedscene="scene_A1__________2_________________" />
</scene>
```

3.3.3. 制作建筑层漫游

如上述方法对建筑物楼层(标准层)全景漫游制作完成后，用制作层 VR (from 标准层)模块来制作一幢建筑物楼层结构相同分层漫游。

制作层 VR (from 标准层)模块利用分层编码与对应楼层关系(分层对照表.xlsx)，如表 3，对已经制作完成的标准层漫游文件进行批量复制和 XML 文件修正。

Table 3. Layer coding and corresponding floor relationship table
表 3. 分层编码与对应楼层关系表

分层编码	楼层
T3204020001100002003 (标准层)	三楼
T3204020001100002004	四楼
T3204020001100002005	五楼
T3204020001100002006	六楼
T3204020001100002007	七层

标准层文件结构以该层编码(标准层)文件夹组织，全景切片文件存放在\vtour\panos\场景名称.tiles 里。

利用分层对照表.xlsx 里的分层编码内容，建立除标准层以外的分层文件夹，将标准层文件夹里的所有子文件夹和文件拷贝到对应的分层文件夹里，利用对应楼层名称建立映射关系，对切片文件重命名。将\vtour\tour.xml 改名为 tour.txt，利用标准层文件夹结构和名称建立字典，建立 def replace_text_in_file 和 def process_directory 处理各分层 tour.txt 需要修改内容。

```
def replace_text_in_file(file_path, old_text, new_text):
# 替换文件内容
    try:
        with open(file_path, 'r', encoding='utf-8') as file:
            content = file.read()
        content = content.replace(old_text, new_text)
        with open(file_path, 'w', encoding='utf-8') as file:
            file.write(content)
    except Exception as e:
        print(f"Error processing {file_path}: {e}")
def process_directory(base_folder, df):
# 获取含“标准层”的第二列值
    standard_layers = df[df.iloc[:, 0].str.contains("标准层")]
    if standard_layers.empty:
        print("没有找到包含标准层的二级文件夹")
        return
    # 创建一个字典用于存储二级文件夹名称与对应的楼层名称
    replacement_dict = {row.iloc[0]: row.iloc[1] for index, row in df.iterrows() if "标准层" not in row.iloc[0]}
        for root, dirs, files in os.walk(base_folder):
        for dir_name in dirs:
            # 检查当前二级文件夹是否在 replacement_dict 中
            if dir_name in replacement_dict:
                old_text = standard_layers.iloc[0, 1]  # 获取第一个“标准层”的对应第二列的值
                new_text = replacement_dict[dir_name]      # 获取当前二级文件夹对应楼层名称
                # 查找并替换 tour.txt 内容
```

```
                tour_file_path = os.path.join(root, dir_name, 'tour.txt')
                if os.path.exists(tour_file_path):
                    print(f"Replacing '{old_text}' with '{new_text}' in {tour_file_path}")
                    replace_text_in_file(tour_file_path, old_text, new_text)
```

模块执行后，会将标准层拷贝到映射文件夹里的 tour.txt 相关内容进行批量修改。

标准层的内容片段：

```
<scene name="scene_03" title="2 号楼三楼电梯进出口" onstart="" thumburl="panos/03.tiles/thumb.jpg" lat="" lng="" heading="">
        <view hlookat="0.0" vlookat="0.0" fovtype="MFOV" fov="120" maxpixelzoom="2.0" fovmin="70" fovmax="140" limitview="auto" />
        <preview url="panos/03.tiles/preview.jpg" />
        <image>
            <cube url="panos/03.tiles/pano_%s.jpg" />
        </image>
        <hotspot name="spot1" style="skin_hotspotstyle" ath="-27.589" atv="24.617" linkedscene="scene_04" />

    </scene>
```

映射关系(四楼)的内容片段：

```
<scene name="scene_四楼 03" title="2 号楼四楼电梯进出口" onstart="" thumburl="panos/四楼 03.tiles/thumb.jpg" lat="" lng="" heading="">
        <view hlookat="0.0" vlookat="0.0" fovtype="MFOV" fov="120" maxpixelzoom="2.0" fovmin="70" fovmax="140" limitview="auto" />
        <preview url="panos/四楼 03.tiles/preview.jpg" />
        <image>
            <cube url="panos/四楼 03.tiles/pano_%s.jpg" />
        </image>
        <hotspot name="spot1" style="skin_hotspotstyle" ath="-27.589" atv="24.617" linkedscene="scene_四楼 04" />
    </scene>
```

3.3.4. 制作建筑幢漫游

利用合并层为幢 VR 模块，对分层全景漫游文件进行处理，在选择存放分层漫游文件的一级文件夹里建立“全部 VR”文件夹，将处理完成的分层文件夹及文件拷贝到全部 VR 文件里，将标准标准层 tour.xml 文件拷贝到全部 VR\vtour 里，重命名为 tour.txt。对各分层 vtour\tour.xml 改名为 tour.txt，对其场景内容进行提取合并，对全部 VR\里的 tour.txt 进行替换。

合并其他 tour.txt 的内容，并替换“全部 VR\vtour\tour.txt”中指定位置的内容。

```
def extract_content(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        content = file.read()
```

```
                start_index = content.find("</action>") + len("</action>")
                end_index = content.find("<!--热点显示文字  -->")
                if start_index != -1 and end_index != -1:
                    return content[start_index:end_index]
            return ""
        # 合并内容
        merged_content = ""
        for sub_folder in sub_folders:
            if sub_folder == "全部 VR":
                continue
            tour_file_path = os.path.join(base_folder, sub_folder, "vtour", "tour.txt")
            if os.path.exists(tour_file_path):
                merged_content += extract_content(tour_file_path)
        # 替换“全部 VR/vtour/tour.txt”中的指定内容
        vr_tour_file = os.path.join(vtour_folder, "tour.txt")
        if os.path.exists(vr_tour_file):
            with open(vr_tour_file, 'r', encoding='utf-8') as file:
                content = file.read()
            # 找到替换位置
            start_index = content.find("</action>") + len("</action>")
            end_index = content.find("<!--热点显示文字  -->")
            if start_index != -1 and end_index != -1:
                # 替换指定范围内的内容
                modified_content = content[:start_index] + merged_content + content[end_index:]
                # 写回文件
                with open(vr_tour_file, 'w', encoding='utf-8') as file:
                    file.write(modified_content)
```

处理前的标准层 tour.txt 场景内容片段：

```
        <scene name="scene_03" title="2 号楼三楼电梯进出口" onstart="" thumburl="panos/03.tiles/thumb.jpg"
lat="" lng="" heading="">
                <view hlookat="0.0" vlookat="0.0" fovtype="MFOV" fov="120" maxpixelzoom="2.0"
fovmin="70" fovmax="140" limitview="auto" />
                <preview url="panos/03.tiles/preview.jpg" />
                <image>
                    <cube url="panos/03.tiles/pano_%s.jpg" />
                </image>
                <hotspot name="spot1" style="skin_hotspotstyle" ath="-27.589" atv="24.617"
linkedscene="scene_04" />
            </scene>
```

处理后幢合并后的 tour.txt 场景内容片段：

```
<scene name="scene_07" title="2 号楼三楼会议室" onstart="" thumburl="panos/07.tiles/thumb.jpg" lat="" lng="" heading="">
        <view hlookat="0.0" vlookat="0.0" fovtype="MFOV" fov="120" maxpixelzoom="2.0" fovmin="70" fovmax="140" limitview="auto" />
        <preview url="panos/07.tiles/preview.jpg" />
        <image>
            <cube url="panos/07.tiles/pano_%s.jpg" />
        </image>
        <hotspot name="spot1" style="skin_hotspotstyle" ath="-47.328" atv="26.910" linkedscene="scene_04" />
    </scene>
    <scene name="scene_08" title="2 号楼三楼大厅" onstart="" thumburl="panos/08.tiles/thumb.jpg" lat="" lng="" heading="">
        <view hlookat="0.0" vlookat="0.0" fovtype="MFOV" fov="120" maxpixelzoom="2.0" fovmin="70" fovmax="140" limitview="auto" />
        <preview url="panos/08.tiles/preview.jpg" />
        <image>
            <cube url="panos/08.tiles/pano_%s.jpg" />
        </image>
        <hotspot name="spot1" style="skin_hotspotstyle" ath="-101.931" atv="15.165" linkedscene="scene_04" />
        <hotspot name="spot2" style="skin_hotspotstyle" ath="176.417" atv="26.840" linkedscene="scene_09" />
        <hotspot name="spot3" style="skin_hotspotstyle" ath="-14.703" atv="25.255" linkedscene="scene_10" />
    </scene>
```

4. 结语

在日常工程中，利用专业软件制作全景漫游，要考察其工作量和制作对象的空间结构。对一定工作量的全景漫游，采用开发软件自动化批量生产全景漫游是首选。

全景漫游基本可以分为二类，一类称为项目漫游，是对地理空间基本在一个平面的工程项目周边和内部进行多场景拍摄；二类称为整幢建筑物漫游，是由分层组成的建筑物幢，在此情况下，对楼层相同的室内只拍摄一次标准层，在制作整幢建筑物漫游时，对标准层进行复制合并。

上述二类漫游制作过程中，其重点一是对场景切片文件的复制，二是对相应 XML 文件的场景、热点、文字、跳转等设置的自动化批量处理。本文通过制作工艺的分析，利用 Python 开发相对应的五个模块，在工程中得到很好的应用，提高了工作效率，提升了产品质量，有一定的实用价值。

参考文献

[1] 龚萍, 王晓龄, 成美娇, 等. 昆明动物博物馆全景漫游及 AR 交互设计研究[J]. 电脑编程技巧与维护, 2024(9): 152-154.

[2] 权西瑞, 王凯, 王小飞, 等. 基于 Three.js 的全景漫游产品设计与实现[J]. 地理空间信息, 2022, 20(7): 71-73, 119.
[3] 王家骐, 于海霞. 虚拟现实视角下三维全景漫游系统之虚拟交互实现[J]. 科技创新与应用, 2023, 13(34): 28-31.
[4] 贾亚娟, 卢春光. 基于图像的虚拟现实全景漫游技术[J]. 科学技术创新, 2022(10): 85-88.
[5] 车森, 杨辉, 葛磊, 等. 全景漫游制作方法及关键技术研究[J]. 测绘工程, 2021, 30(5): 38-42.