



# 展域智绘

## 户外广告物料与 效果图生成

OUTDOOR ADVERTISING MATERIALS & RENDERING GENERATION

# 目录 content



01

## 选题背景与意义

RESEARCH BACKGROUND AND VALUE

02

## 研究框架

RESEARCH STRUCTURE

03

## 产品展示

RESEARCH CONTENT

04

## 总结与思考

SUMMARY AND THINKING

01

# 选题背景与意义

RESEARCH BACKGROUND AND VALUE





## 数据资源的爆炸性增长

- 全球数据量正以惊人的速度增长：根据IDC的预测，全球数据量将在未来几年内达到数百ZB，相当于数万亿GB。这些数据来源于社交媒体、电子商务、物联网设备等多个领域。
- 数据在类型上更加多样化：文本、图像、视频、音频等多媒体数据的大量产生，为AIGC提供了丰富的素材库，使得生成的内容更加生动和真实。



## AI驱动广告变革：创意、制作到分发的重塑

- 当今时代，AI已经成为许多行业的变革力量。AIGC作为一种新兴的内容生产模式，相比于PGC和UGC，有更高的产出效率、更为稳定的内容质量、更低的产出成本，其内容的可拓展性也更强。
- 在广告领域，AI技术正在重塑广告创意、制作和分发的全过程。其中AIGC作为人工智能技术的一个重要分支，已经在国内外图频广告行业中得到了广泛的应用。

## 传统的制作展示流程繁琐低效

传统的户外广告制作普遍依赖人工设计与后期处理。设计师完成海报设计后，还需通过P图、样机等手段将广告内容“植入”实际投放场景（如公交站牌、地铁灯箱），以模拟真实展示效果。这一过程涉及大量手动操作，修改频繁、周期长、成本高，对广告投放的效率与灵活性有所制约。



## 户外广告时效性与个性化需求升级

随着市场竞争的日益激烈，企业对户外广告的时效性、个性化和精准度提出了更高的要求，传统设计模式已难以满足企业快速迭代、多样化的宣传需求。不同应用场景下的广告展示效果差异巨大，如何快速且精准地呈现物料在实际户外场景中的视觉效果，成为行业亟待解决的难题。

3

# 户外广告物料与效果图生成系统

一款专注于**户外广告物料与效果图生成**的 AI agent  
为用户提供便捷、高效、专业的户外广告设计服务

?是什么

展域智绘

?怎么用



有户外广告设计宣传需求的  
**设计师/品牌/代理商**

?给谁用

- 输入**产品名称、物料描述、物料尺寸和产品展示图**，快速生成**产品的宣传物料**
- 结合用户指定的应用场景和产品物料海报，生成**户外广告展示效果图**，还原广告在实际户外环境中的呈现效果。

整合Coze、ComfyUI等AI工具构建工作流，实现“图像分析-创意生成-效果优化”的应用闭环。通过多模型协同，高效联动不同效能的AI产品，实践广告生产从“人工主导”向“AI协同”的转型。

## 实现AI产品的有机融合实践

通过基于AIGC的户外广告物料生成实践，挖掘其在广告设计领域的创新潜力。项目覆盖从产品图分析到海报生成的全流程，探索可复用的技术路径。

### AIGC在广告领域的细分应用



通过AI技术实现了广告物料与效果图的快速生成，提升设计效率并降低企业的宣传成本。同时帮助企业快速预览广告在不同户外场景下的展示效果，响应户外广告行业向智能化、高效化方向发展的趋势。

### 降本增效并提升广告效果

02

## 研究框架

RESEARCH CONSTRUCTURE



1

# 研究框架

## 物料生成

### 文生图

图像理解+用户创意



方案→Prompt优化



大模型：素材生成



### 图生图

上传产品图



FLUX图像融合



物料成品图



## 效果图生成

上传  
实景图



深度图  
边缘检测



坐标检测  
透视矩阵

物料  
成品图



应用透视  
合成遮罩



图像融合  
光影渲染

↓

→

## 扣子

工作流整合  
前端复杂交互



## liblib | ngrok

云端部署  
内网穿透



## ComfyUI

强大的生图功能  
丰富的插件生态

## Python

自定义节点  
强化ComfyUI

# 2 产品物料图生成：文生图生成海报物料

拆解步骤，通过coze工作流初步生成海报图

产品素材理解 → 物料设计方案形成 → 中文提示词优化 → 海报物料生成



产品特色：后与新会柑皮邂逅诞生青柑普洱，  
道光三十年普洱茶经茶乌道出滇，0脂肪  
产品图案：一只彩色仙鹤，旁边有红色的果实

画面主体应该突出产品，可能需要展示瓶身，上面的仙鹤和红色果实是关键图案，要清晰呈现。主色调选择清新的绿色和橙色，象征柑和普洱，背景可以用茶园或柑园的自然场景，加上云雾效果，营造清纯感。构图上，产品放在中央，文案置于下方或侧面，保持平衡。

2

## 产品物料图生成：图生图智能替换产品图

产品图不一致



+



想要保持背景和文字不变  
并替换产品为真实产品



- coze常见图像生成插件，再次经过图生图后，**文字混乱**
- 既参考真实产品图，又参考背景图的**多图参考**难以实现

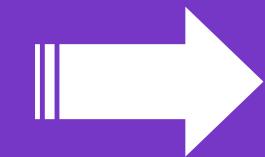


ComfyUI工作流  
FLUX-Kontext模型

2

# 产品物料图生成：图生图智能替换产品图

识别图像 生成提示词



liblib云端在线ComfyUI工作流 图生图

**豆包·视觉理解·Pro·1.5**

**识别图像 生成提示词**

**输出**

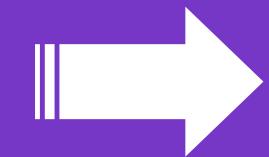
描述两张图中的产品  
生成替换产品的提示词



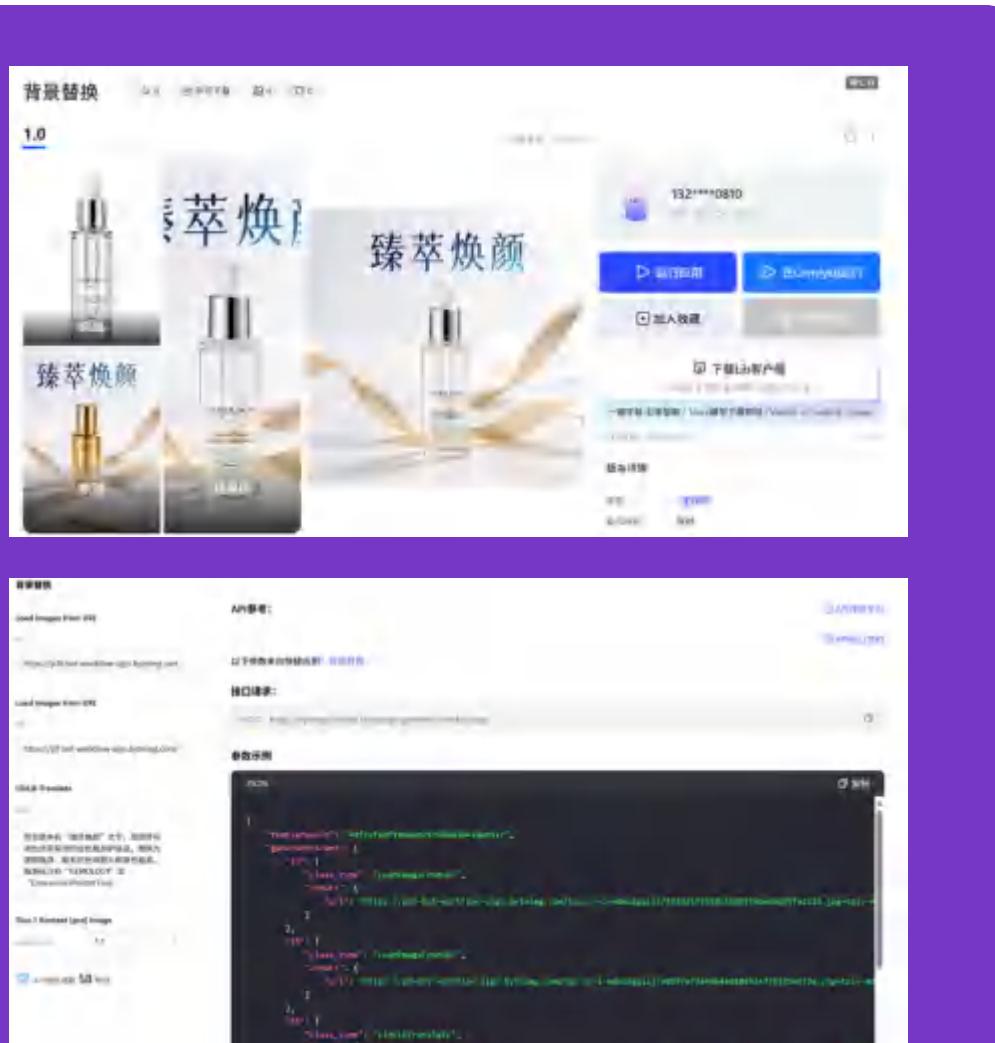
2

## 产品物料图生成：图生图智能替换产品图

liblib发布应用 获取API接口

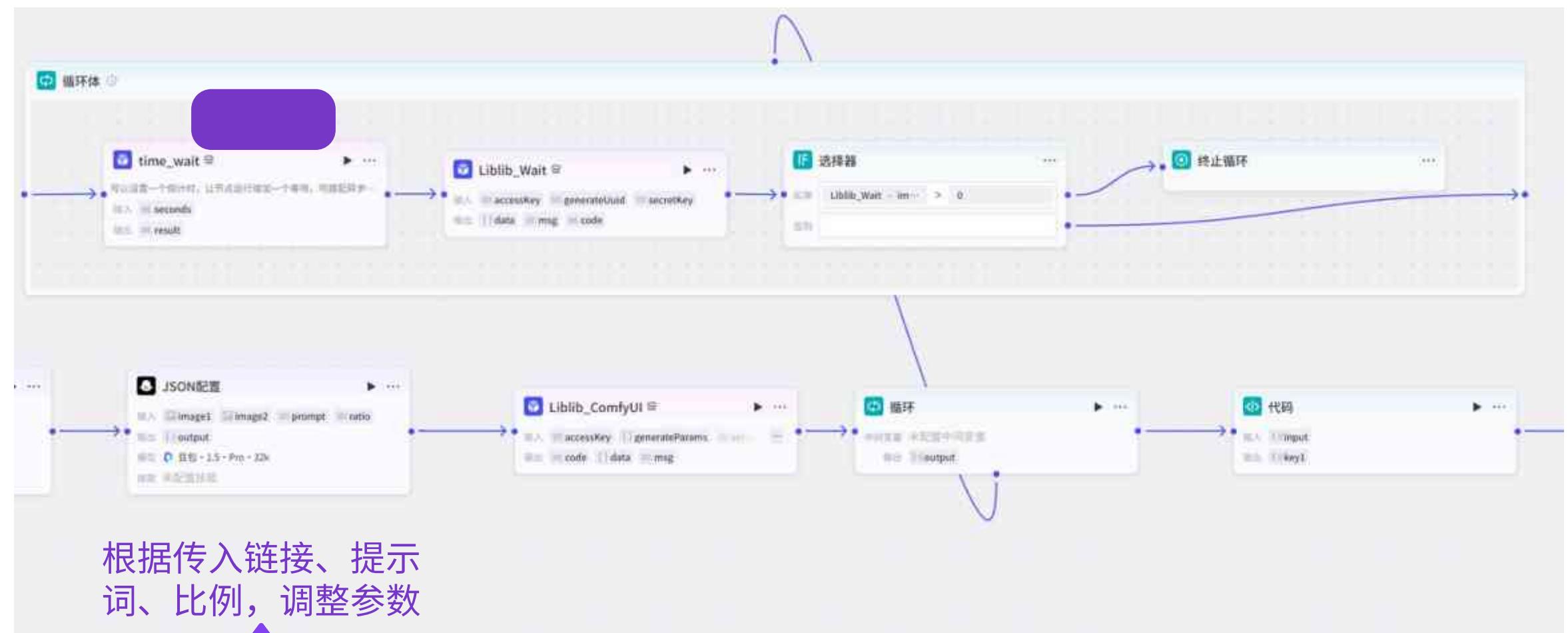


coze工作流 使用liblib插件调用API



设定图片链接、提示词、比例为可更改部分

liblib是异步执行，通过不断循环调用，直到liblib返回结果



2

# 产品物料图生成：工作流整合

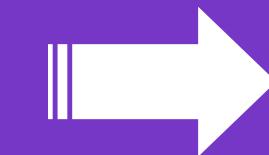
用户传入比例参数  
选择器判定支路    → 生成特定比例的  
待替换海报物料    → 变量聚合获取最终运行  
支路出参的图片URL    → 运行liblib调用comfyui  
工作流替换产品



2

## 产品物料图生成：物料效果提升结果展示

文生图-初步物料



图生图-替换产品



2

## 产品物料图生成：物料效果提升结果展示

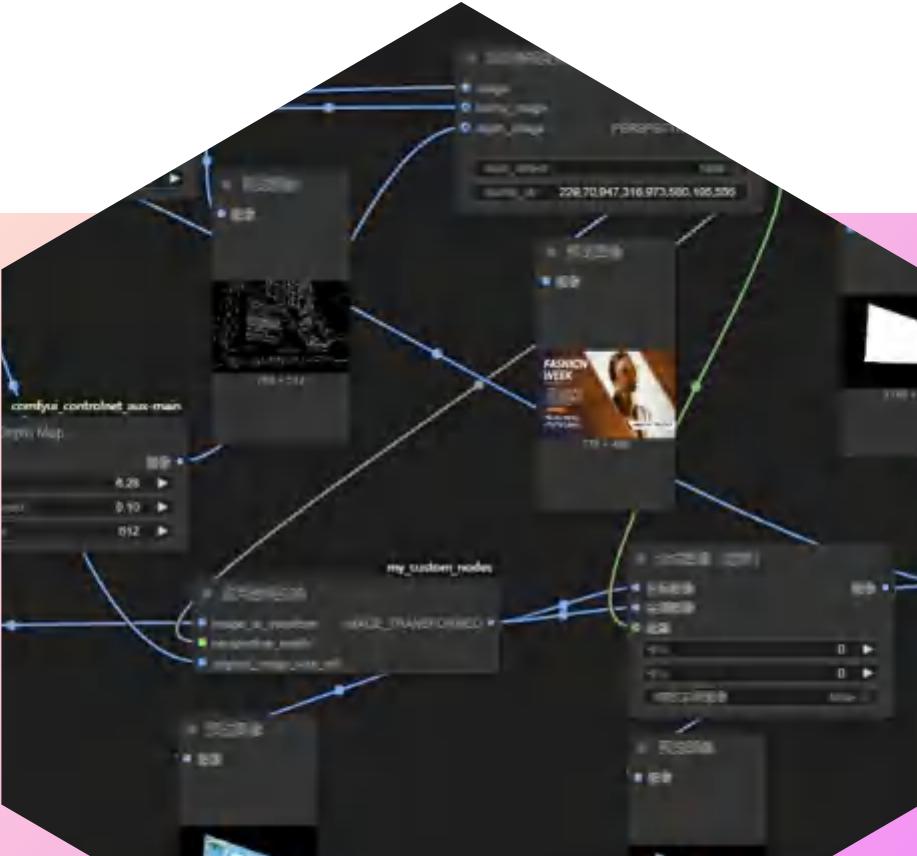
如果在提示词中表明户外场景

可以直接生成户外效果



# 3

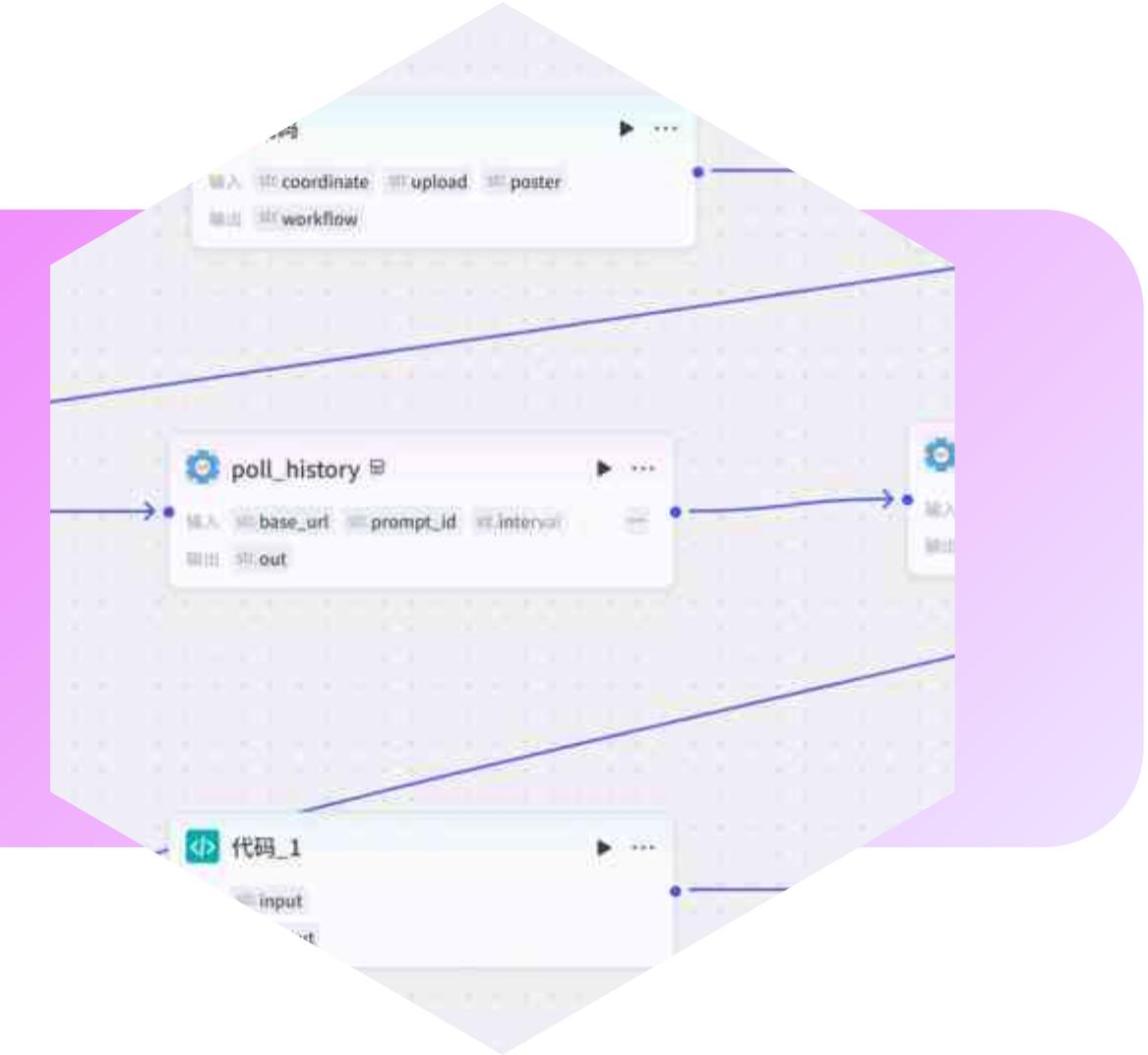
# 户外效果图生成



ComfyUI



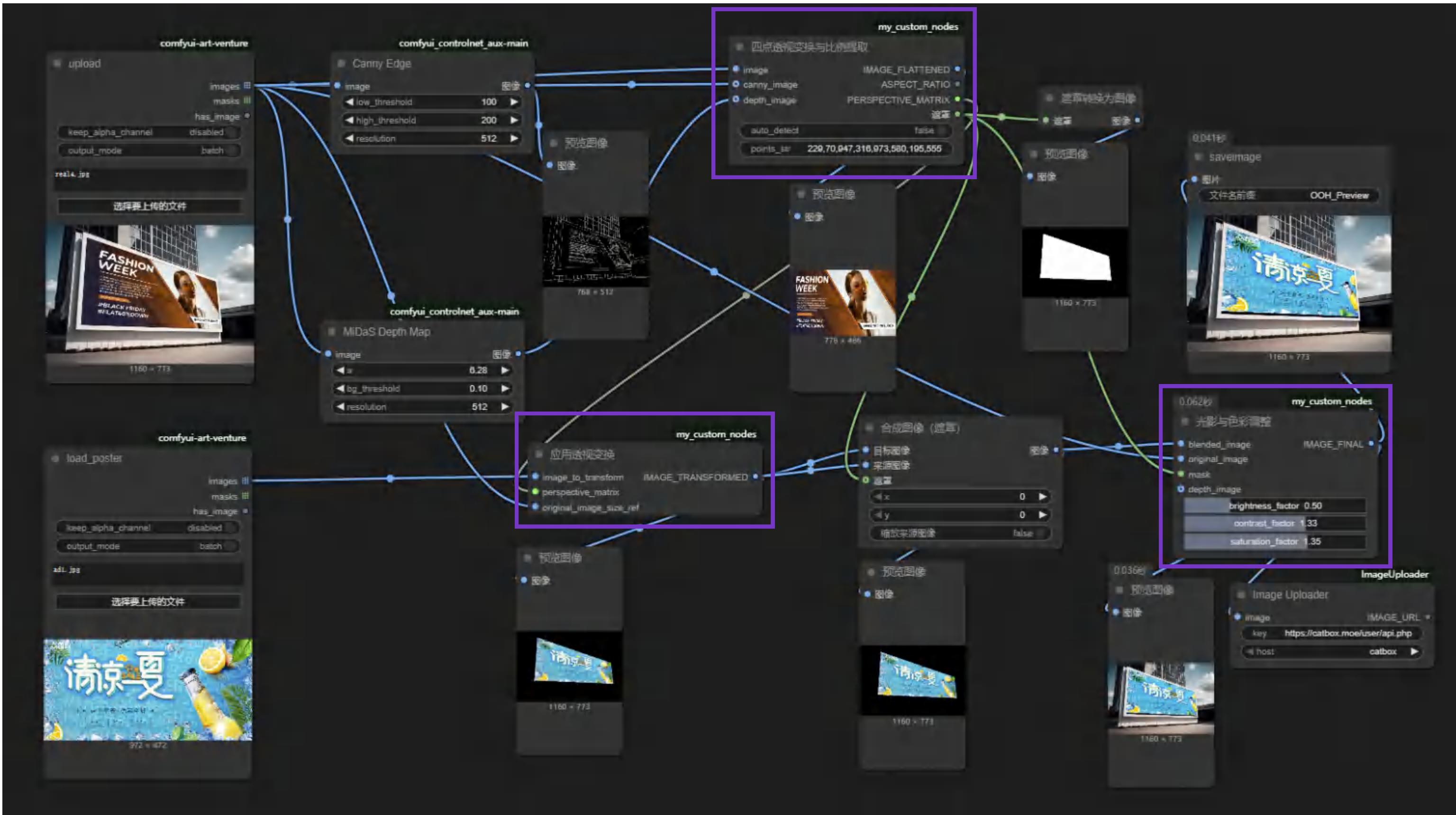
ngrok



COZE

## 3

# 户外效果图生成 - 底层逻辑



缺少可用插件

广告屏难以检测

如何在线调用

## 3

# 户外效果图生成 - 问题解决

## 自定义节点

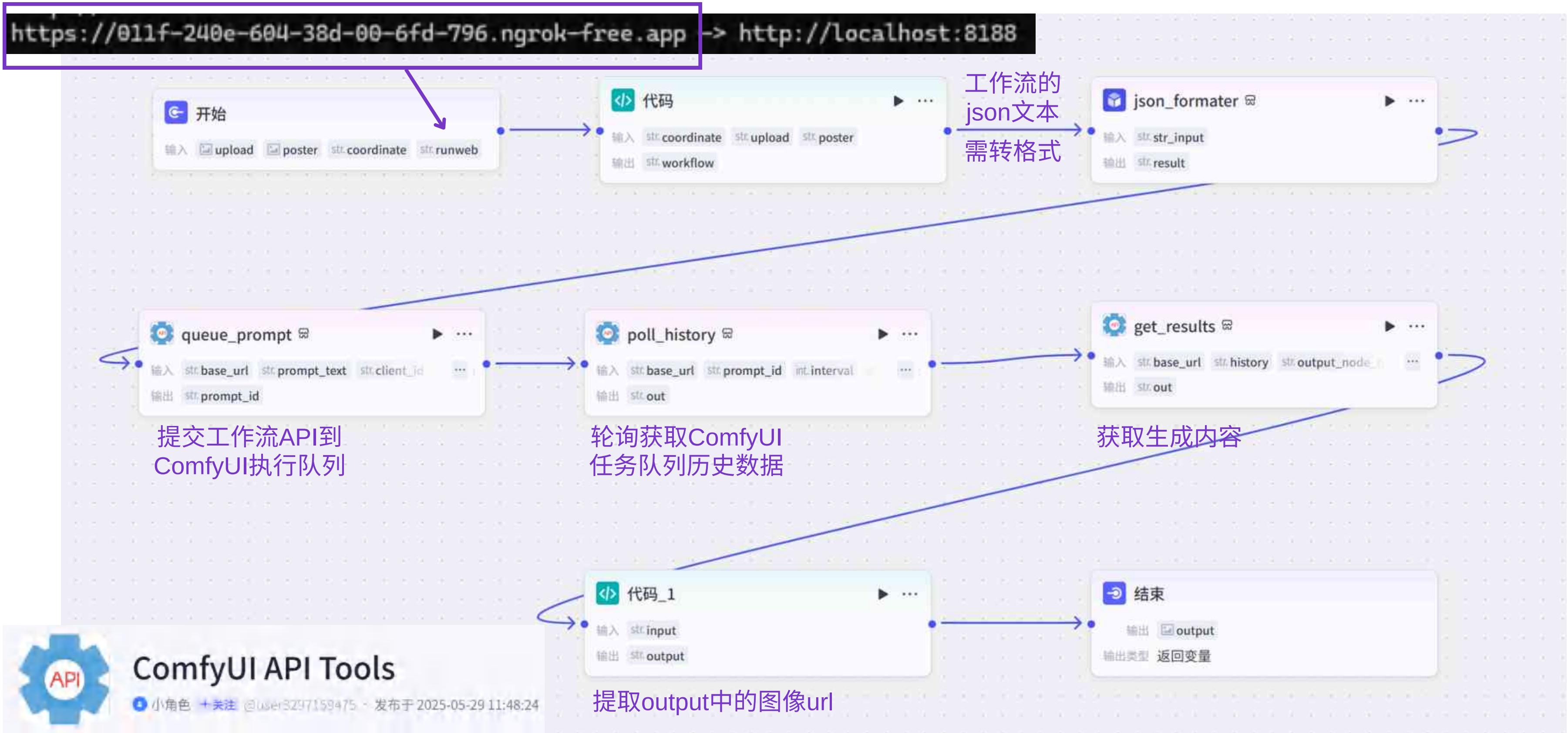
```

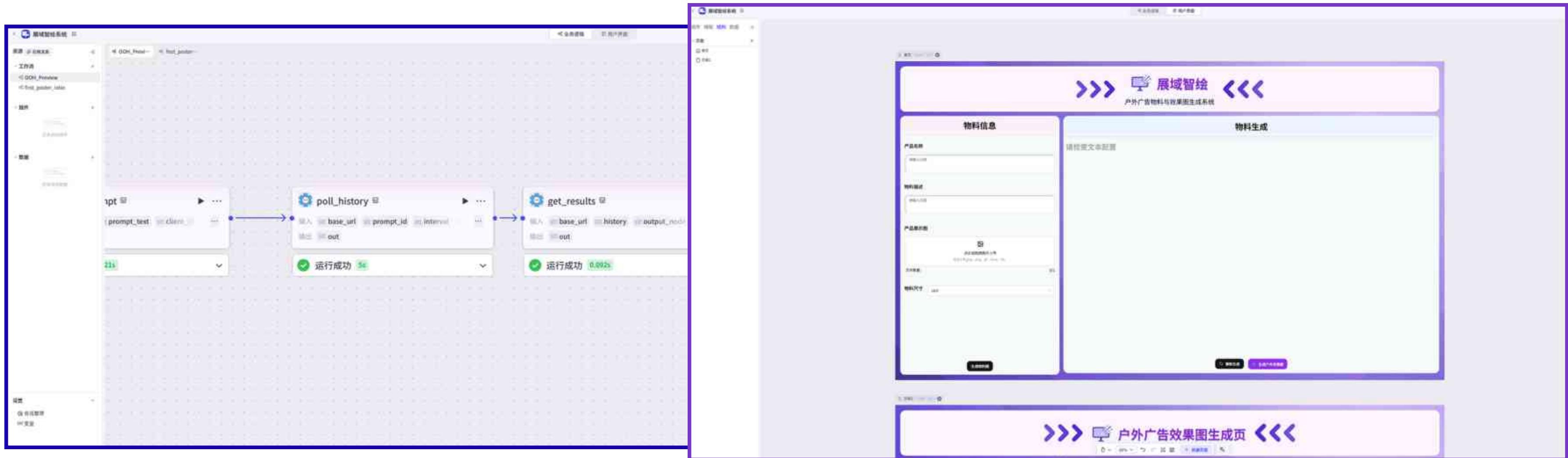
1 // 定义节点类
2 class CustomNode_FourPointPerspectiveTransform:
3     # 重写 __init__ 方法
4     def __init__(self):
5         self.INPUT_TYPES = ["image"]
6         self.RETURNS = ["image"]
7         self.FUNCTION = "apply_transform"
8         self.CATEGORY = "Image Processing"
9
10    # 定义方法
11    def apply_transform(self, image):
12        # 读取图像
13        img_np = np.array(image)
14
15        # 调用自定义函数
16        result_np = self._apply_transform(img_np)
17
18        # 将结果转换为 PIL 图像
19        result_image = Image.fromarray(result_np)
20
21        return result_image
22
23    # 定义私有方法
24    def _apply_transform(self, img_np):
25        # 读取图像
26        original_w, original_h = img_np.size
27
28        # 定义四个角点
29        points_str = "0,0 100,0 100,100 0,100"
30
31        # 将字符串转换为浮点数列表
32        points_flat = [float(p) for p in points_str.split(",")]
33
34        # 检查点数是否正确
35        if len(points_flat) != 8:
36            raise ValueError("点字符串必须包含8个逗号分隔的坐标 (x1,y1,x2,y2,x3,y3,x4,y4)。")
37
38        # 将点数组重新排列
39        src_pts = np.array(points_flat).reshape(4, 2).astype(np.float32)
40
41        # 尝试自动检测广告牌
42        if self.INPUT_TYPES == ["image"]:
43            # 使用YOLO模型检测广告牌
44            src_pts = detect_billboard_corners(img_np)
45
46            # 如果检测失败，手动输入
47            if src_pts is None:
48                raise RuntimeError("AI未检测到广告牌，请切换为手动输入模式。")
49
50            print(f"AI检测到的点: {src_pts.tolist()}")
51
52        # 计算透视变换矩阵
53        M_inv = cv2.getPerspectiveTransform(src_pts, np.array([[0, 0], [100, 0], [100, 100], [0, 100]]))
54
55        # 应用透视变换
56        transformed_poster_np = cv2.warpPerspective(img_np, M_inv, (original_w, original_h), flags=cv2.INTER_LINEAR)
57
58        # 将结果转换为 PIL 图像
59        transformed_image = Image.fromarray(transformed_poster_np)
60
61        return transformed_image
62
63    # 定义私有方法
64    def _adjust_size(self, image, target_size):
65        # 调整大小
66        adjusted_image = image.resize(target_size)
67
68        return adjusted_image
69
70    # 定义私有方法
71    def _convert_to_cmyk(self, image):
72        # 转换为 CMYK
73        cmyk_image = image.convert("CMYK")
74
75        return cmyk_image
76
77    # 定义私有方法
78    def _convert_to_l(self, image):
79        # 转换为 L
80        l_image = image.convert("L")
81
82        return l_image
83
84    # 定义私有方法
85    def _convert_to_pil(self, image):
86        # 转换为 PIL
87        pil_image = Image.fromarray(np.array(image))
88
89        return pil_image
90
91    # 定义私有方法
92    def _convert_to_np(self, image):
93        # 转换为 np
94        np_image = np.array(image)
95
96        return np_image
97
98    # 定义私有方法
99    def _convert_to_tiff(self, image):
100       # 转换为 TIFF
101      tiff_image = image.save("output.tiff")
102
103      return tiff_image
104
105    # 定义私有方法
106    def _convert_to_jpg(self, image):
107       # 转换为 JPG
108      jpg_image = image.save("output.jpg")
109
110      return jpg_image
111
112    # 定义私有方法
113    def _convert_to_png(self, image):
114       # 转换为 PNG
115      png_image = image.save("output.png")
116
117      return png_image
118
119    # 定义私有方法
120    def _convert_to_gif(self, image):
121       # 转换为 GIF
122      gif_image = image.save("output.gif")
123
124      return gif_image
125
126    # 定义私有方法
127    def _convert_to_tiff(self, image):
128       # 转换为 TIFF
129      tiff_image = image.save("output.tiff")
130
131      return tiff_image
132
133    # 定义私有方法
134    def _convert_to_jpg(self, image):
135       # 转换为 JPG
136      jpg_image = image.save("output.jpg")
137
138      return jpg_image
139
140    # 定义私有方法
141    def _convert_to_png(self, image):
142       # 转换为 PNG
143      png_image = image.save("output.png")
144
145      return png_image
146
147    # 定义私有方法
148    def _convert_to_gif(self, image):
149       # 转换为 GIF
150      gif_image = image.save("output.gif")
151
152      return gif_image
153
154    # 定义私有方法
155    def _convert_to_tiff(self, image):
156       # 转换为 TIFF
157      tiff_image = image.save("output.tiff")
158
159      return tiff_image
160
161    # 定义私有方法
162    def _convert_to_jpg(self, image):
163       # 转换为 JPG
164      jpg_image = image.save("output.jpg")
165
166      return jpg_image
167
168    # 定义私有方法
169    def _convert_to_png(self, image):
170       # 转换为 PNG
171      png_image = image.save("output.png")
172
173      return png_image
174
175    # 定义私有方法
176    def _convert_to_gif(self, image):
177       # 转换为 GIF
178      gif_image = image.save("output.gif")
179
180      return gif_image
181
182    # 定义私有方法
183    def _convert_to_tiff(self, image):
184       # 转换为 TIFF
185      tiff_image = image.save("output.tiff")
186
187      return tiff_image
188
189    # 定义私有方法
190    def _convert_to_jpg(self, image):
191       # 转换为 JPG
192      jpg_image = image.save("output.jpg")
193
194      return jpg_image
195
196    # 定义私有方法
197    def _convert_to_png(self, image):
198       # 转换为 PNG
199      png_image = image.save("output.png")
200
201      return png_image
202
203    # 定义私有方法
204    def _convert_to_gif(self, image):
205       # 转换为 GIF
206      gif_image = image.save("output.gif")
207
208      return gif_image
209
210    # 定义私有方法
211    def _convert_to_tiff(self, image):
212       # 转换为 TIFF
213      tiff_image = image.save("output.tiff")
214
215      return tiff_image
216
217    # 定义私有方法
218    def _convert_to_jpg(self, image):
219       # 转换为 JPG
220      jpg_image = image.save("output.jpg")
221
222      return jpg_image
223
224    # 定义私有方法
225    def _convert_to_png(self, image):
226       # 转换为 PNG
227      png_image = image.save("output.png")
228
229      return png_image
230
231    # 定义私有方法
232    def _convert_to_gif(self, image):
233       # 转换为 GIF
234      gif_image = image.save("output.gif")
235
236      return gif_image
237
238    # 定义私有方法
239    def _convert_to_tiff(self, image):
240       # 转换为 TIFF
241      tiff_image = image.save("output.tiff")
242
243      return tiff_image
244
245    # 定义私有方法
246    def _convert_to_jpg(self, image):
247       # 转换为 JPG
248      jpg_image = image.save("output.jpg")
249
250      return jpg_image
251
252    # 定义私有方法
253    def _convert_to_png(self, image):
254       # 转换为 PNG
255      png_image = image.save("output.png")
256
257      return png_image
258
259    # 定义私有方法
260    def _convert_to_gif(self, image):
261       # 转换为 GIF
262      gif_image = image.save("output.gif")
263
264      return gif_image
265
266    # 定义私有方法
267    def _convert_to_tiff(self, image):
268       # 转换为 TIFF
269      tiff_image = image.save("output.tiff")
270
271      return tiff_image
272
273    # 定义私有方法
274    def _convert_to_jpg(self, image):
275       # 转换为 JPG
276      jpg_image = image.save("output.jpg")
277
278      return jpg_image
279
280    # 定义私有方法
281    def _convert_to_png(self, image):
282       # 转换为 PNG
283      png_image = image.save("output.png")
284
285      return png_image
286
287    # 定义私有方法
288    def _convert_to_gif(self, image):
289       # 转换为 GIF
290      gif_image = image.save("output.gif")
291
292      return gif_image
293
294    # 定义私有方法
295    def _convert_to_tiff(self, image):
296       # 转换为 TIFF
297      tiff_image = image.save("output.tiff")
298
299      return tiff_image
300
301    # 定义私有方法
302    def _convert_to_jpg(self, image):
303       # 转换为 JPG
304      jpg_image = image.save("output.jpg")
305
306      return jpg_image
307
308    # 定义私有方法
309    def _convert_to_png(self, image):
310       # 转换为 PNG
311      png_image = image.save("output.png")
312
313      return png_image
314
315    # 定义私有方法
316    def _convert_to_gif(self, image):
317       # 转换为 GIF
318      gif_image = image.save("output.gif")
319
320      return gif_image
321
322    # 定义私有方法
323    def _convert_to_tiff(self, image):
324       # 转换为 TIFF
325      tiff_image = image.save("output.tiff")
326
327      return tiff_image
328
329    # 定义私有方法
330    def _convert_to_jpg(self, image):
331       # 转换为 JPG
332      jpg_image = image.save("output.jpg")
333
334      return jpg_image
335
336    # 定义私有方法
337    def _convert_to_png(self, image):
338       # 转换为 PNG
339      png_image = image.save("output.png")
340
341      return png_image
342
343    # 定义私有方法
344    def _convert_to_gif(self, image):
345       # 转换为 GIF
346      gif_image = image.save("output.gif")
347
348      return gif_image
349
350    # 定义私有方法
351    def _convert_to_tiff(self, image):
352       # 转换为 TIFF
353      tiff_image = image.save("output.tiff")
354
355      return tiff_image
356
357    # 定义私有方法
358    def _convert_to_jpg(self, image):
359       # 转换为 JPG
360      jpg_image = image.save("output.jpg")
361
362      return jpg_image
363
364    # 定义私有方法
365    def _convert_to_png(self, image):
366       # 转换为 PNG
367      png_image = image.save("output.png")
368
369      return png_image
370
371    # 定义私有方法
372    def _convert_to_gif(self, image):
373       # 转换为 GIF
374      gif_image = image.save("output.gif")
375
376      return gif_image
377
378    # 定义私有方法
379    def _convert_to_tiff(self, image):
380       # 转换为 TIFF
381      tiff_image = image.save("output.tiff")
382
383      return tiff_image
384
385    # 定义私有方法
386    def _convert_to_jpg(self, image):
387       # 转换为 JPG
388      jpg_image = image.save("output.jpg")
389
390      return jpg_image
391
392    # 定义私有方法
393    def _convert_to_png(self, image):
394       # 转换为 PNG
395      png_image = image.save("output.png")
396
397      return png_image
398
399    # 定义私有方法
400    def _convert_to_gif(self, image):
401       # 转换为 GIF
402      gif_image = image.save("output.gif")
403
404      return gif_image
405
406    # 定义私有方法
407    def _convert_to_tiff(self, image):
408       # 转换为 TIFF
409      tiff_image = image.save("output.tiff")
410
411      return tiff_image
412
413    # 定义私有方法
414    def _convert_to_jpg(self, image):
415       # 转换为 JPG
416      jpg_image = image.save("output.jpg")
417
418      return jpg_image
419
420    # 定义私有方法
421    def _convert_to_png(self, image):
422       # 转换为 PNG
423      png_image = image.save("output.png")
424
425      return png_image
426
427    # 定义私有方法
428    def _convert_to_gif(self, image):
429       # 转换为 GIF
430      gif_image = image.save("output.gif")
431
432      return gif_image
433
434    # 定义私有方法
435    def _convert_to_tiff(self, image):
436       # 转换为 TIFF
437      tiff_image = image.save("output.tiff")
438
439      return tiff_image
440
441    # 定义私有方法
442    def _convert_to_jpg(self, image):
443       # 转换为 JPG
444      jpg_image = image.save("output.jpg")
445
446      return jpg_image
447
448    # 定义私有方法
449    def _convert_to_png(self, image):
450       # 转换为 PNG
451      png_image = image.save("output.png")
452
453      return png_image
454
455    # 定义私有方法
456    def _convert_to_gif(self, image):
457       # 转换为 GIF
458      gif_image = image.save("output.gif")
459
460      return gif_image
461
462    # 定义私有方法
463    def _convert_to_tiff(self, image):
464       # 转换为 TIFF
465      tiff_image = image.save("output.tiff")
466
467      return tiff_image
468
469    # 定义私有方法
470    def _convert_to_jpg(self, image):
471       # 转换为 JPG
472      jpg_image = image.save("output.jpg")
473
474      return jpg_image
475
476    # 定义私有方法
477    def _convert_to_png(self, image):
478       # 转换为 PNG
479      png_image = image.save("output.png")
480
481      return png_image
482
483    # 定义私有方法
484    def _convert_to_gif(self, image):
485       # 转换为 GIF
486      gif_image = image.save("output.gif")
487
488      return gif_image
489
490    # 定义私有方法
491    def _convert_to_tiff(self, image):
492       # 转换为 TIFF
493      tiff_image = image.save("output.tiff")
494
495      return tiff_image
496
497    # 定义私有方法
498    def _convert_to_jpg(self, image):
499       # 转换为 JPG
500      jpg_image = image.save("output.jpg")
501
502      return jpg_image
503
504    # 定义私有方法
505    def _convert_to_png(self, image):
506       # 转换为 PNG
507      png_image = image.save("output.png")
508
509      return png_image
510
511    # 定义私有方法
512    def _convert_to_gif(self, image):
513       # 转换为 GIF
514      gif_image = image.save("output.gif")
515
516      return gif_image
517
518    # 定义私有方法
519    def _convert_to_tiff(self, image):
520       # 转换为 TIFF
521      tiff_image = image.save("output.tiff")
522
523      return tiff_image
524
525    # 定义私有方法
526    def _convert_to_jpg(self, image):
527       # 转换为 JPG
528      jpg_image = image.save("output.jpg")
529
530      return jpg_image
531
532    # 定义私有方法
533    def _convert_to_png(self, image):
534       # 转换为 PNG
535      png_image = image.save("output.png")
536
537      return png_image
538
539    # 定义私有方法
540    def _convert_to_gif(self, image):
541       # 转换为 GIF
542      gif_image = image.save("output.gif")
543
544      return gif_image
545
546    # 定义私有方法
547    def _convert_to_tiff(self, image):
548       # 转换为 TIFF
549      tiff_image = image.save("output.tiff")
550
551      return tiff_image
552
553    # 定义私有方法
554    def _convert_to_jpg(self, image):
555       # 转换为 JPG
556      jpg_image = image.save("output.jpg")
557
558      return jpg_image
559
560    # 定义私有方法
561    def _convert_to_png(self, image):
562       # 转换为 PNG
563      png_image = image.save("output.png")
564
565      return png_image
566
567    # 定义私有方法
568    def _convert_to_gif(self, image):
569       # 转换为 GIF
570      gif_image = image.save("output.gif")
571
572      return gif_image
573
574    # 定义私有方法
575    def _convert_to_tiff(self, image):
576       # 转换为 TIFF
577      tiff_image = image.save("output.tiff")
578
579      return tiff_image
580
581    # 定义私有方法
582    def _convert_to_jpg(self, image):
583       # 转换为 JPG
584      jpg_image = image.save("output.jpg")
585
586      return jpg_image
587
588    # 定义私有方法
589    def _convert_to_png(self, image):
590       # 转换为 PNG
591      png_image = image.save("output.png")
592
593      return png_image
594
595    # 定义私有方法
596    def _convert_to_gif(self, image):
597       # 转换为 GIF
598      gif_image = image.save("output.gif")
599
600      return gif_image
601
602    # 定义私有方法
603    def _convert_to_tiff(self, image):
604       # 转换为 TIFF
605      tiff_image = image.save("output.tiff")
606
607      return tiff_image
608
609    # 定义私有方法
610    def _convert_to_jpg(self, image):
611       # 转换为 JPG
612      jpg_image = image.save("output.jpg")
613
614      return jpg_image
615
616    # 定义私有方法
617    def _convert_to_png(self, image):
618       # 转换为 PNG
619      png_image = image.save("output.png")
620
621      return png_image
622
623    # 定义私有方法
624    def _convert_to_gif(self, image):
625       # 转换为 GIF
626      gif_image = image.save("output.gif")
627
628      return gif_image
629
630    # 定义私有方法
631    def _convert_to_tiff(self, image):
632       # 转换为 TIFF
633      tiff_image = image.save("output.tiff")
634
635      return tiff_image
636
637    # 定义私有方法
638    def _convert_to_jpg(self, image):
639       # 转换为 JPG
640      jpg_image = image.save("output.jpg")
641
642      return jpg_image
643
644    # 定义私有方法
645    def _convert_to_png(self, image):
646       # 转换为 PNG
647      png_image = image.save("output.png")
648
649      return png_image
650
651    # 定义私有方法
652    def _convert_to_gif(self, image):
653       # 转换为 GIF
654      gif_image = image.save("output.gif")
655
656      return gif_image
657
658    # 定义私有方法
659    def _convert_to_tiff(self, image):
660       # 转换为 TIFF
661      tiff_image = image.save("output.tiff")
662
663      return tiff_image
664
665    # 定义私有方法
666    def _convert_to_jpg(self, image):
667       # 转换为 JPG
668      jpg_image = image.save("output.jpg")
669
670      return jpg_image
671
672    # 定义私有方法
673    def _convert_to_png(self, image):
674       # 转换为 PNG
675      png_image = image.save("output.png")
676
677      return png_image
678
679    # 定义私有方法
680    def _convert_to_gif(self, image):
681       # 转换为 GIF
682      gif_image = image.save("output.gif")
683
684      return gif_image
685
686    # 定义私有方法
687    def _convert_to_tiff(self, image):
688       # 转换为 TIFF
689      tiff_image = image.save("output.tiff")
690
691      return tiff_image
692
693    # 定义私有方法
694    def _convert_to_jpg(self, image):
695       # 转换为 JPG
696      jpg_image = image.save("output.jpg")
697
698      return jpg_image
699
700    # 定义私有方法
701    def _convert_to_png(self, image):
702       # 转换为 PNG
703      png_image = image.save("output.png")
704
705      return png_image
706
707    # 定义私有方法
708    def _convert_to_gif(self, image):
709       # 转换为 GIF
710      gif_image = image.save("output.gif")
711
712      return gif_image
713
714    # 定义私有方法
715    def _convert_to_tiff(self, image):
716       # 转换为 TIFF
717      tiff_image = image.save("output.tiff")
718
719      return tiff_image
720
721    # 定义私有方法
722    def _convert_to_jpg(self, image):
723       # 转换为 JPG
724      jpg_image = image.save("output.jpg")
725
726      return jpg_image
727
728    # 定义私有方法
729    def _convert_to_png(self, image):
730       # 转换为 PNG
731      png_image = image.save("output.png")
732
733      return png_image
734
735    # 定义私有方法
736    def _convert_to_gif(self, image):
737       # 转换为 GIF
738      gif_image = image.save("output.gif")
739
740      return gif_image
741
742    # 定义私有方法
743    def _convert_to_tiff(self, image):
744       # 转换为 TIFF
745      tiff_image = image.save("output.tiff")
746
747      return tiff_image
748
749    # 定义私有方法
750    def _convert_to_jpg(self, image):
751       # 转换为 JPG
752      jpg_image = image.save("output.jpg")
753
754      return jpg_image
755
756    # 定义私有方法
757    def _convert_to_png(self, image):
758       # 转换为 PNG
759      png_image = image.save("output.png")
760
761      return png_image
762
763    # 定义私有方法
764    def _convert_to_gif(self, image):
765       # 转换为 GIF
766      gif_image = image.save("output.gif")
767
768      return gif_image
769
770    # 定义私有方法
771    def _convert_to_tiff(self, image):
772       # 转换为 TIFF
773      tiff_image = image.save("output.tiff")
774
775      return tiff_image
776
777    # 定义私有方法
778    def _convert_to_jpg(self, image):
779       # 转换为 JPG
780      jpg_image = image.save("output.jpg")
781
782      return jpg_image
783
784    # 定义私有方法
785    def _convert_to_png(self, image):

```

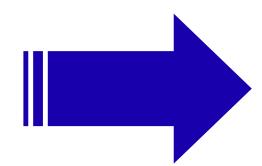
## 3

## 户外效果图生成 - 在线调用

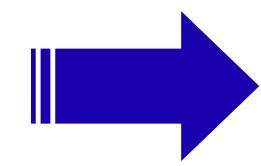




开发应用项目  
导入工作流



设计用户  
使用界面



前后端绑定  
全流程测试

03

## 产品展示

RESEARCH CONSTRUCTURE



1

# 运行展示

展域智绘 户外广告物料与效果图生成系统

物料信息

产品名称: HBN柔光水

物料描述: 化妆水

产品展示图

文件类型: HBN柔光水.jpg

物料尺寸: 100px

物料生成

https://liblibai-tmp-image.liblib.cloud/img/0f109d14a11a4cd68ac7d7fda1fd899a/8f7c124feb0c4a2e67a8931918a06773f52dc40178a03f452ce2c05888a9d157.png

生成生成 取消并返回

历史

- 2024-03-22 14:50:22 -> https://liblibai-tmp-image.liblib.cloud/img/0f109d14a11a4cd68ac7d7fda1fd899a/8f7c124feb0c4a2e67a8931918a06773f52dc40178a03f452ce2c05888a9d157.png



# 1

# 运行展示

>>> 📺 户外广告效果图生成页 <<<

广告位信息

广告位四角坐标  
H311f-240e-604-38d-00-6fd-796.ngrok-free.app

产品物料图  
HBN\_Honey-Mate\_Beauty-Naturalism\_00121\_.png

户外广告截图  
HBN\_Honey-Mate\_Beauty-Naturalism\_00121\_.png

ngrok (中国)  
https://011f-240e-604-38d-00-6fd-796.ngrok-free.app/

生成海报

效果图生成  
[https://011f-240e-604-38d-00-6fd-796.ngrok-free.app/view?filename=OOH\\_Preview\\_00121\\_.png&subfolder=&type=output](https://011f-240e-604-38d-00-6fd-796.ngrok-free.app/view?filename=OOH_Preview_00121_.png&subfolder=&type=output)

输出文件  
output undefined

输出格式  
image



## 2 成品展示

产品图与  
生成设置



物料信息

产品名称: 华为三折叠手机

物料描述: 高端大气, 面向高端商务人群

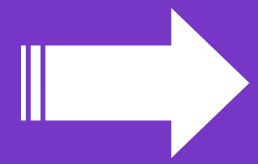
产品展示图:

文件类型: jpg, gif, pdf, bmp, docx

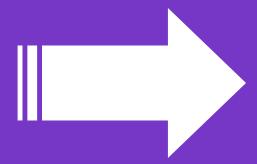
文件数量: 1/1

文件名: 华为三折叠手机.jpg

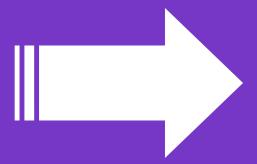
物料尺寸: 3:4



生成物料图



户外广告展示底图

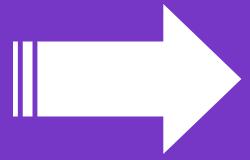


生成效果展示图

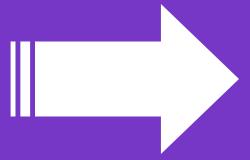


## 2 成品展示

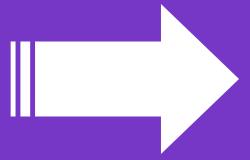
产品图与  
生成设置



生成物料图



户外广告展示底图



生成效果展示图



04

## 总结与思考

RESEARCH BACKGROUND AND VALUE



## 2 优化方向

### 改良图像生成模型

目前，海报物料的生成主要依赖Coze平台内置图像生成插件中的通用AI模型，该模型不支持参考图输入，导致生成的海报中产品主体与其实际外观有时存在较大偏差。这种不确定性会使得后续进行智能替换产品图时的难度也更高。

因此，下一步的优化方向将考虑跳出Coze平台，探索其他具备参考图控制能力的AI生成工具，在保证生成质量的同时，简化整体生成流程，提升可控性与效率。

### 训练广告屏幕检测模型

当前，户外广告效果生成中的广告屏检测主要依赖人工坐标键入，局限有二：

- ①需要依赖第三方工具获取坐标；
- ②虽然能够有效处理平面屏幕，但难以处理曲面屏和裸眼3D效果。

由于前端交互的局限性，下一步的优化方向将是训练轻量级的广告屏幕检测模型，在实现自动检测空间坐标的同时保证效率。

# 感谢 您的观看

THANK YOUR VIEWING

