

Web端实时 防挡弹幕

哔哩哔哩

刘俊·风痕

目录

1 原理解析

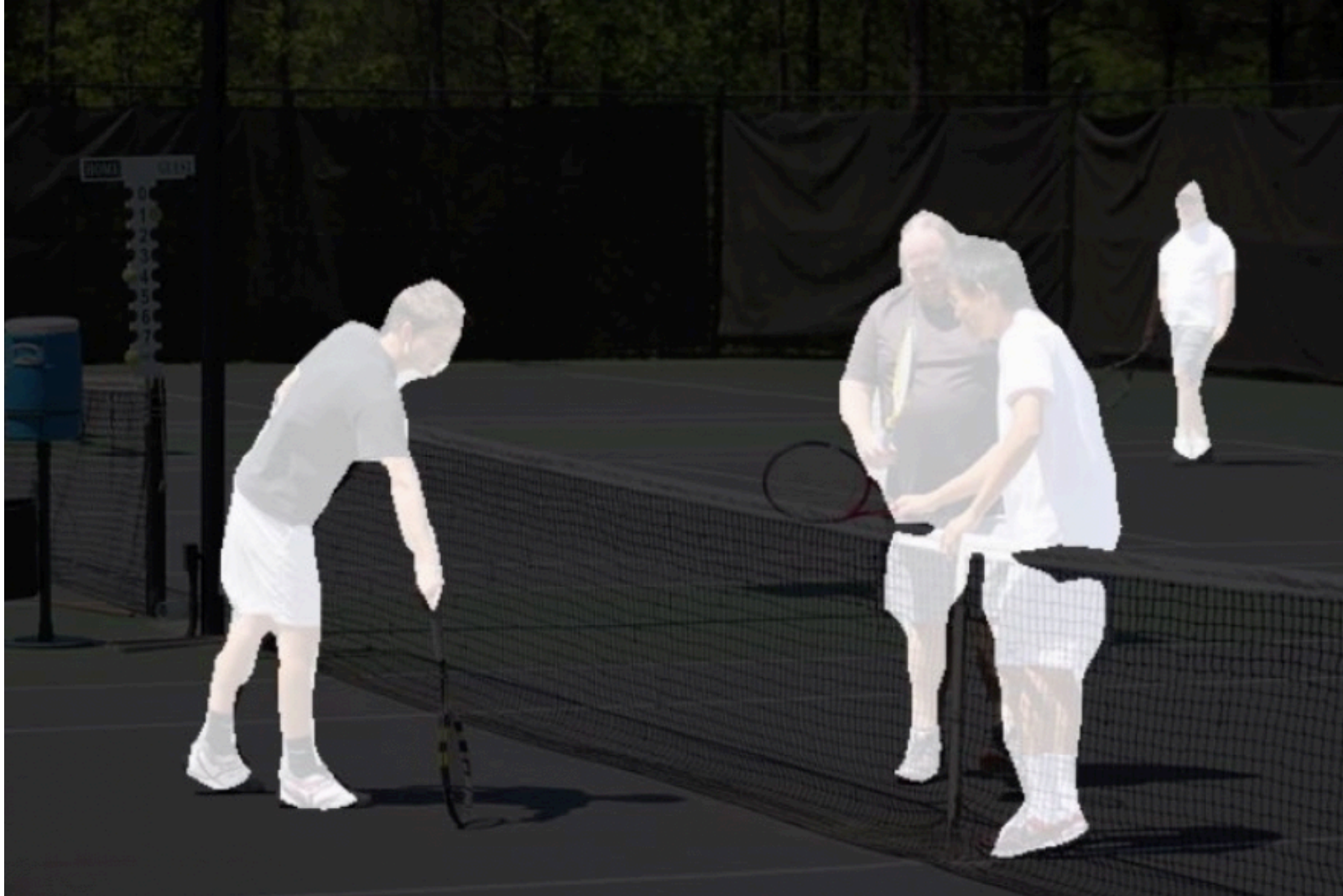
2 工程实践

3 经验总结

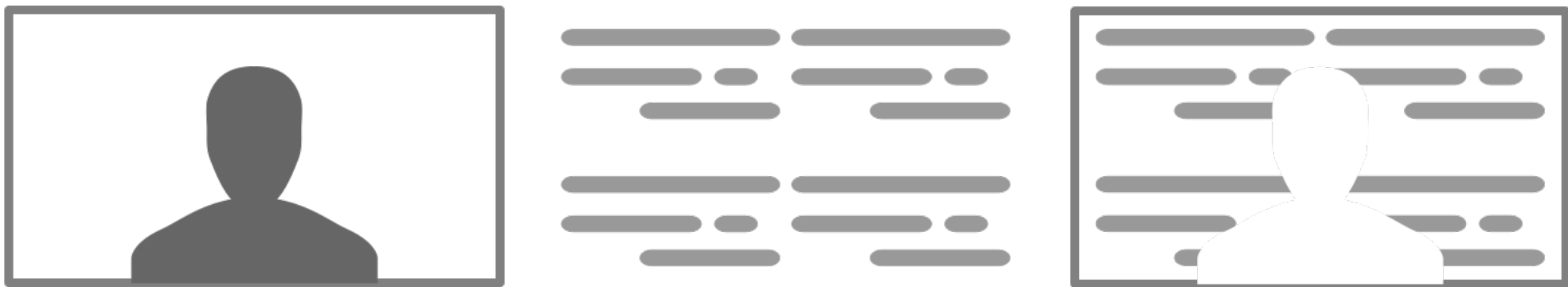
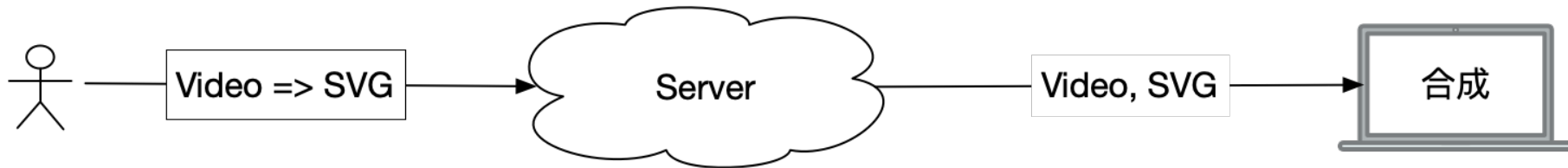
防挡弹幕介绍



Mask 生成




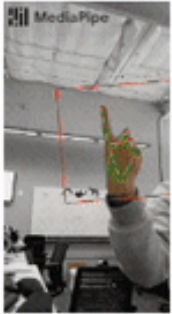




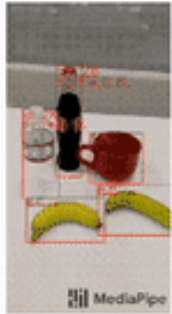





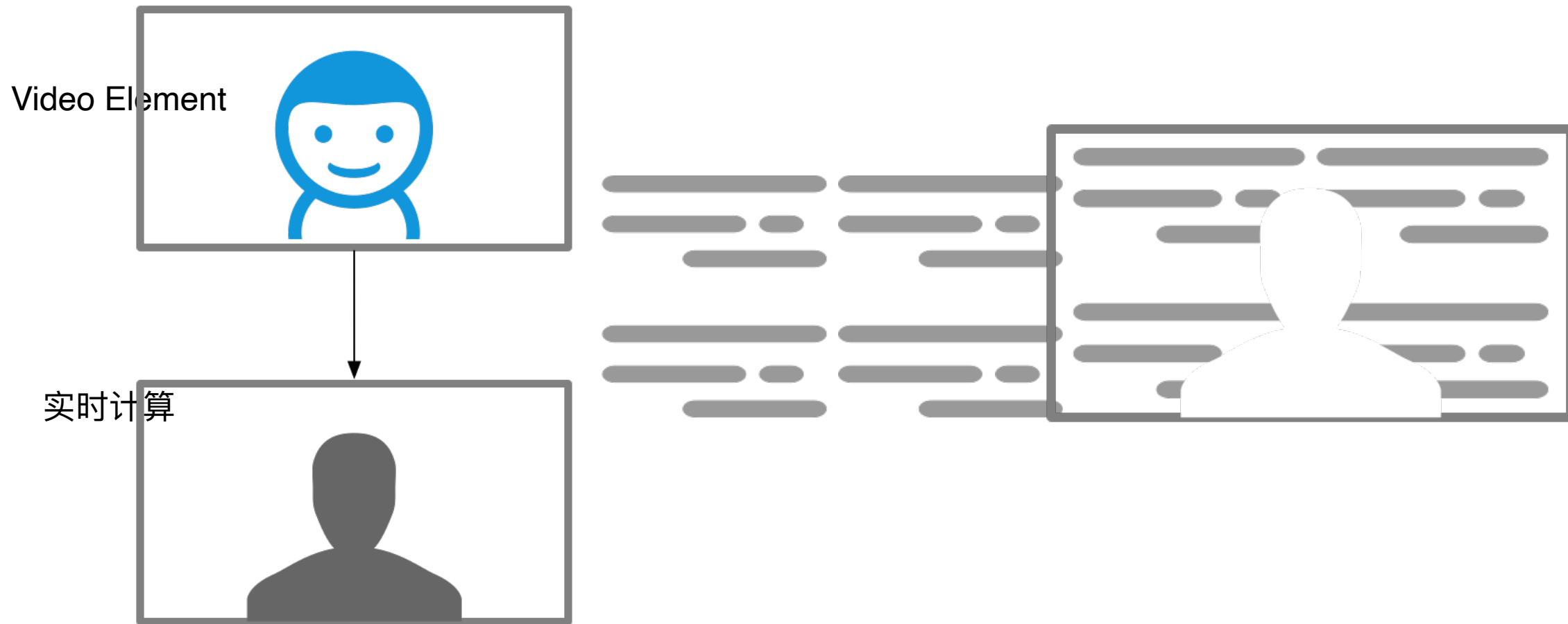
防挡弹幕原理



Mask (SVG) + 弹幕 => 防挡弹幕

ML solutions in MediaPipe

Face Detection	Face Mesh	Iris	Hands	Pose	Holistic
					
Hair Segmentation	Object Detection	Box Tracking	Instant Motion Tracking	Objectron	KNIFT
					



工程实践

选择模型, CPU 70%

```
1 [
2   {
3     score: 0.8,
4     keypoints: [
5       {x: 230, y: 220, score: 0.9, score: 0.99, name: "nose"},
6       {x: 212, y: 190, score: 0.8, score: 0.91, name: "left_eye"},
7       ...
8     ],
9     keypoints3D: [
10      {x: 0.65, y: 0.11, z: 0.05, score: 0.99, name: "nose"},
11      ...
12    ],
13    segmentation: {
14      maskValueToLabel: (maskValue: number) => { return 'person' },
15      mask: {
16        toCanvasImageSource(): ...
17        toImageData(): ...
18        toTensor(): ...
19        getUnderlyingType(): ...
20      }
21    }
22  }
23 ]
```

BlazePose



模型输出

```
1 {
2   maskValueToLabel: (maskValue: number) => { return 'person' },
3   mask: {
4     toCanvasImageSource(): ...
5     toImageData(): ...
6     toTensor(): ...
7     getUnderlyingType(): ...
8   }
9 }
```

SelfieSegmentation

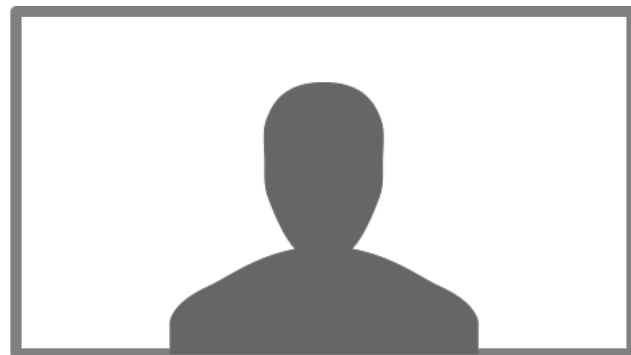
降低频率，CPU 50%



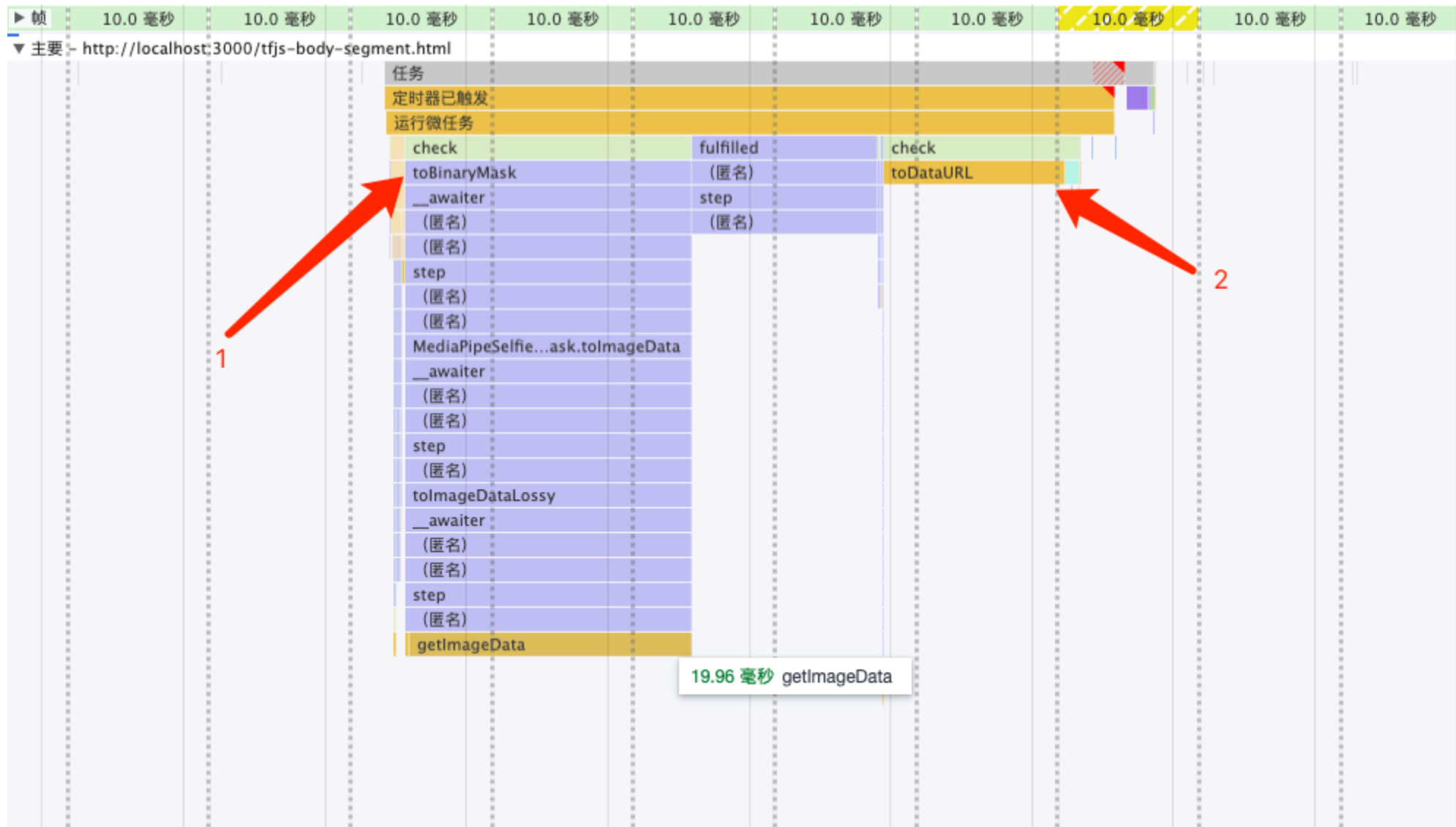
30 FPS



15 FPS



分析性能瓶颈



API 替代实现, CPU 33%



```
// 1. 将`ImageBitmap`绘制到 Canvas 上
context.drawImage(
  // 经验证 即使出现多人, 也只有一个 segmentation
  await segmentation[0].mask.toCanvasImageSource(),
  0, 0,
  canvas.width, canvas.height
)

// 2. 设置混合模式
context.globalCompositeOperation = 'source-out'

// 3. 反向填充黑色
context.fillRect(0, 0, canvas.width, canvas.height)

// 导出Mask图片, 需要的是轮廓, 图片质量设为最低
handler(canvas.toDataURL('image/png', 0))
```

API 替代实现, CPU 33%



模型输出

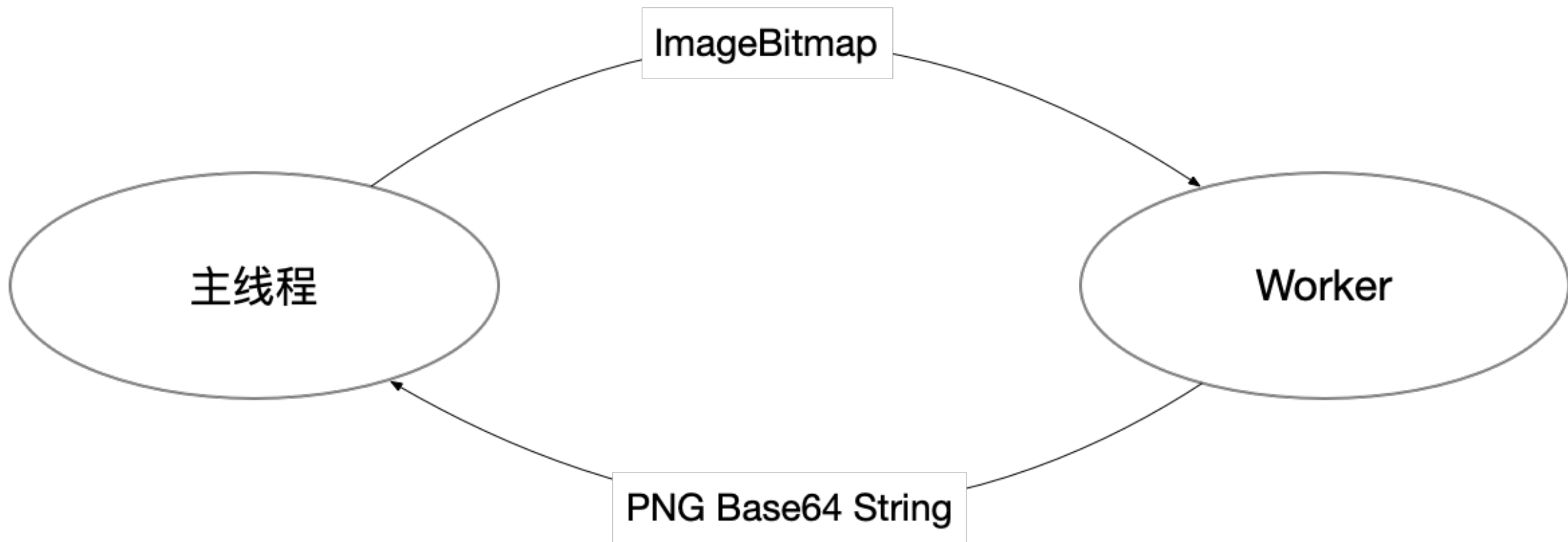


css mask-image 需要

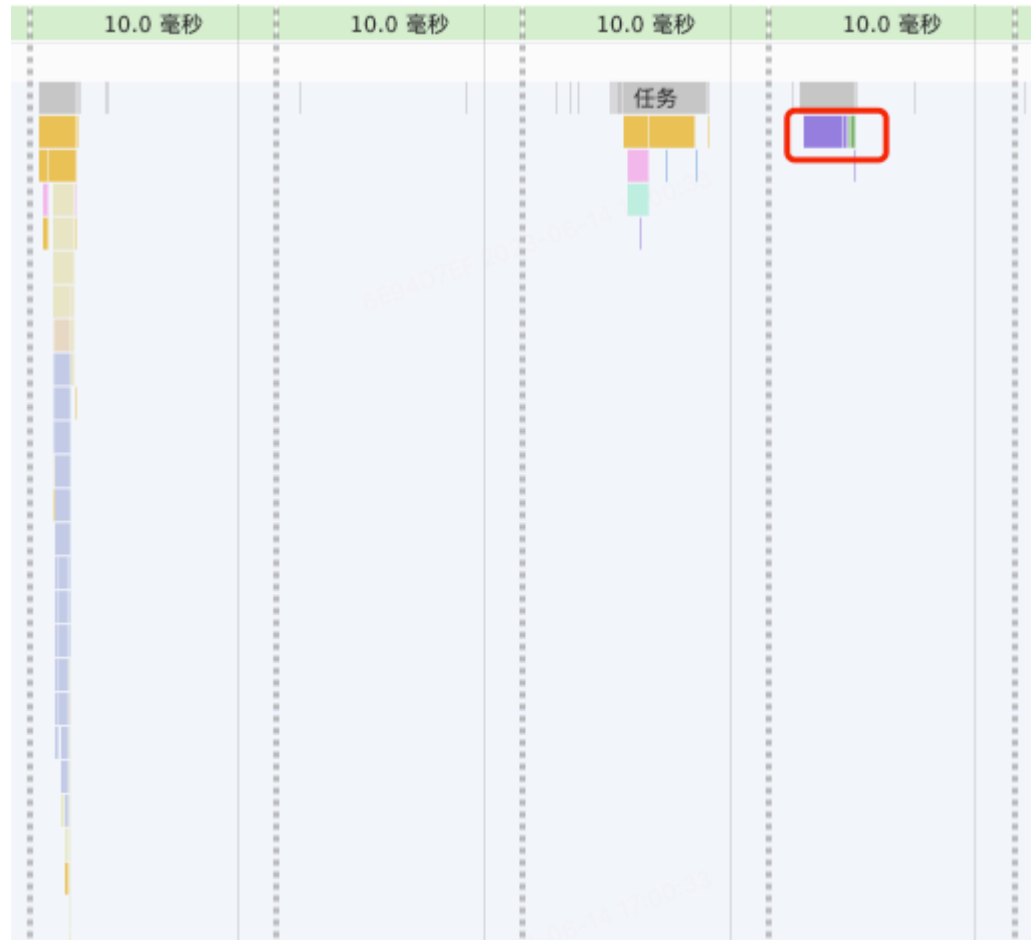
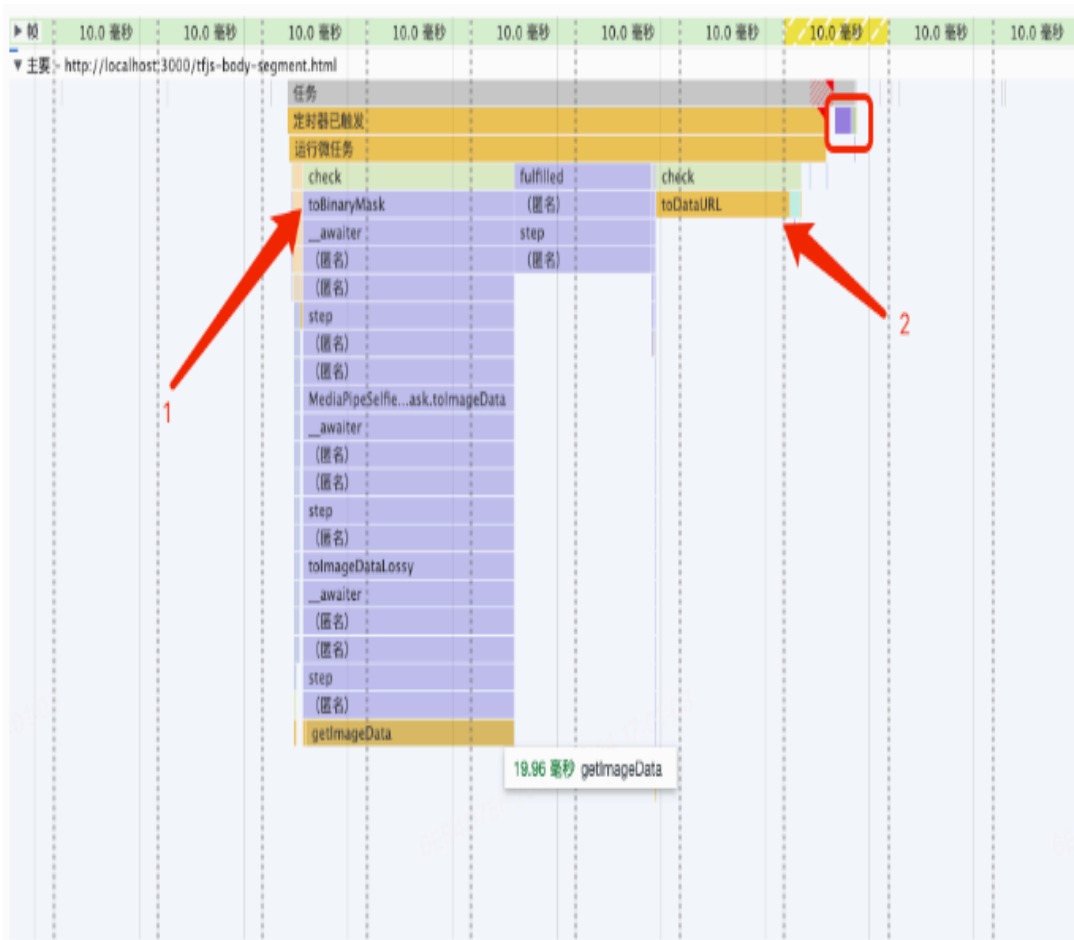


toDataURL

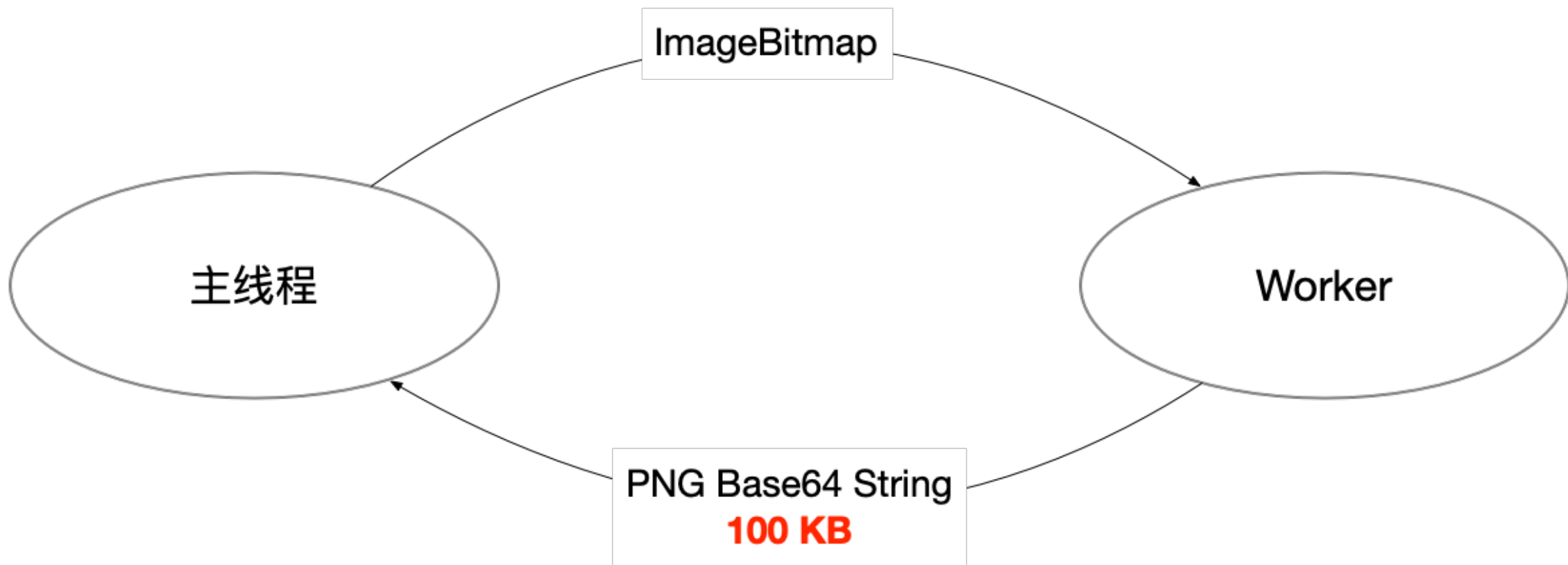
PNG
Base64 String



降低图片尺寸



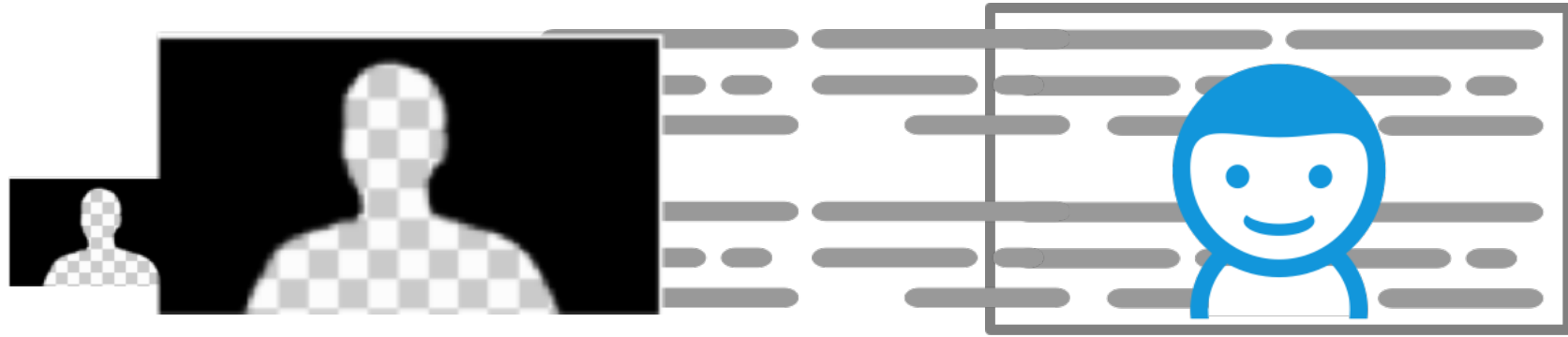
优化前后



```
danmakuContainer.style.maskImage = `url(${imgStr})`
```



缩小尺寸

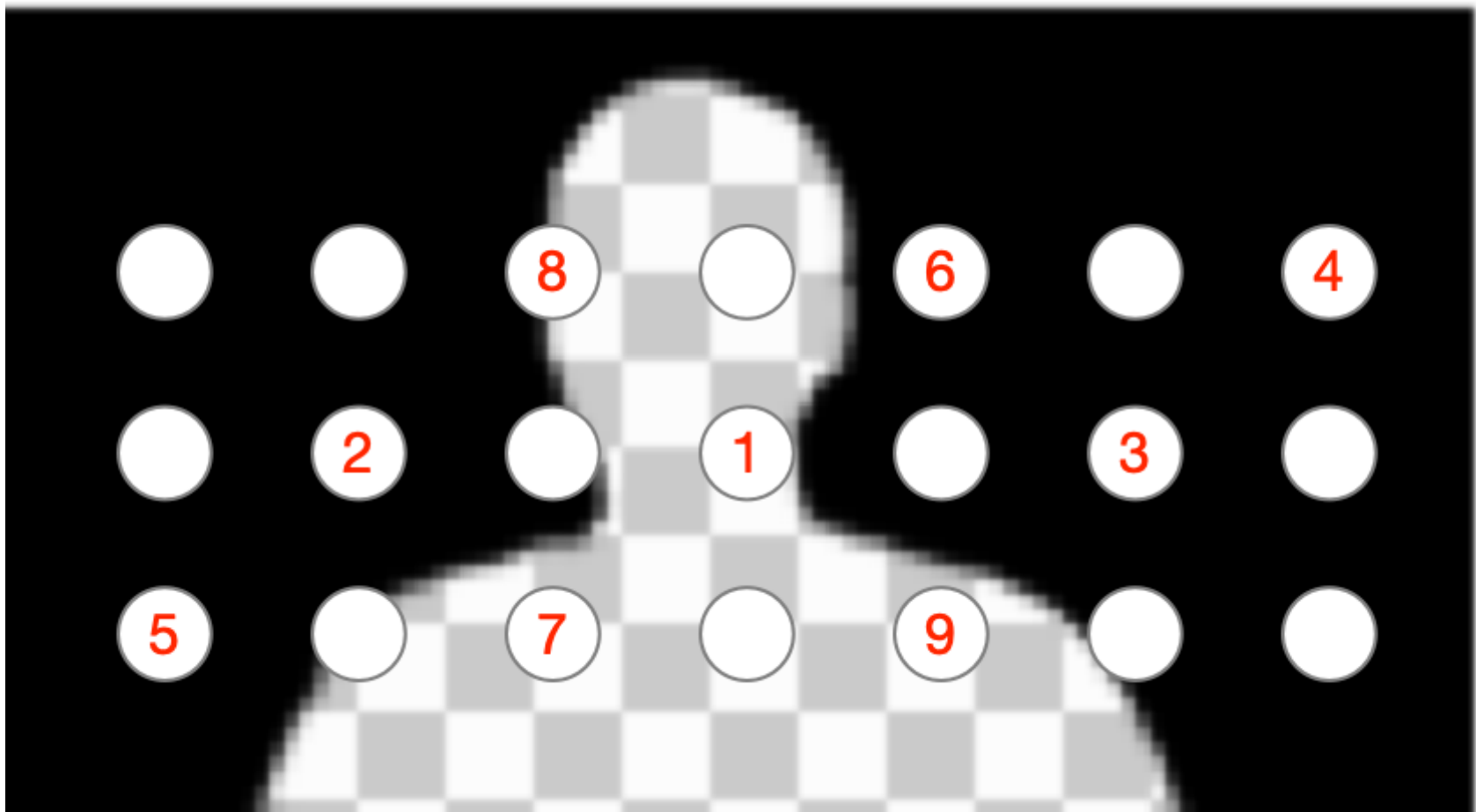


拉伸mask再合成

降低图片尺寸, CPU 5%

✓ CPU 使用情况	6.5%
✓ JS 堆大小	16.8 MB
✓ DOM 节点	365
✓ JS 事件监听器	344
✓ 文档	
✓ 文档框架	
✓ 布局个数/秒	0
✓ 样式重新计算次数/秒	

- 无（少）弹幕 暂停运行
- 画面中没人 逐渐降低计算频率



左右横跳检查像素值

- 选择高性能模型后，初始状态 CPU 70%
- 降低 Mask 提取频率，CPU 50%
- MediaPipe API 替代实现，CPU 33%
- 多线程优化，CPU 15%
- 缩小 Mask 尺寸，CPU 5%
- 判断画面是否有人，无人时 CPU 接近 0%

CPU 数值指主线程占
用

- 结合业务场景特征进行分析优化
- 优化后主要计算量在GPU
- 可拓展的应用场景
 - 背景替换, 自动马赛克, 抠图
 - 人体姿态检测、互动特效 (更换模型)
- 关注 WebNN、WebGPU 进展和应用

Q & A



哔哩哔哩技术

<https://github.com/hughfenghen/WebAV/>