**🚨 多智能体应急意图理解与协同决策系统**



**🌐 项目概要**

本项目旨在构建一个**应急场景下的多智能体智能决策系统**，  
通过「规划 → 执行 → 总结」三层架构实现从**用户意图理解到自动决策生成**的完整闭环。

系统核心目标：

将用户的多模态输入（文本、语音、图像）自动转化为可执行的应急任务，  
并通过多个AI模型协同完成，从而输出一份可直接用于指挥的综合决策报告。

**🧩 模块一：规划器（Planner）**

**📌 作用**

解析用户多模态输入，识别显式意图、隐式意图与环境信息，  
并生成**层次化任务流程（Hierarchical Task Flow）**。

**📥 输入**

用户输入（文本、语音、图像）：

{

"text": "请帮我规划南区火灾的救援路线。",

"image": "./inputs/fire\_scene.jpg",

"audio": "./inputs/report.wav"

}

**📤 输出**

任务与步骤结构：

{

"tasks": [

{

"task\_id": "T1",

"task\_name": "fire\_region\_analysis",

"steps": [

{"model": "SAM", "input": "fire\_scene.jpg"},

{"model": "GRIT", "input": "fire\_mask.png"}

]

},

{

"task\_id": "T2",

"task\_name": "path\_planning",

"steps": [

{"model": "A\*", "input": "safe\_zone.json"}

]

}

]

}

**🎯 输出意义**

为系统提供任务执行蓝图——定义“要做什么”和“用什么模型做”。

**⚙️ 模块二：调用器（Executor）**

**📌 作用**

根据规划器生成的任务结构，调度并执行对应的模型，  
在执行中通过\*\*共享意图变量（Shared Intent Variable）\*\*保证输出一致性。

**📥 输入**

规划器生成的任务流程：

{

"workflow": ["T1", "T2"],

"shared\_intent": {

"goal": "优化救援决策",

"location": "南区A栋"

}

}

**📤 输出**

各模型的执行结果：

{

"results": [

{"task\_id": "T1", "summary": "火源位于南区A栋东侧"},

{"task\_id": "T2", "summary": "推荐北门进入，西北角为安全区"}

]

}

**🎯 输出意义**

将任务流程转化为可运行的模型调用，产生结构化中间结果供总结使用。

**🧠 模块三：总结器（Aggregator）**

**📌 作用**

整合所有模型输出，结合用户需求与共享意图变量，  
通过LLM生成最终的**应急决策报告与态势总结**。

**📥 输入**

来自调用器的模型结果 + 用户初始需求：

{

"results": [...],

"user\_query": "请告诉我当前最优的救援方案。"

}

**📤 输出**

决策级报告：

{

"final\_summary": "火源集中东侧，建议北门救援并在西北角设医疗点。",

"visual\_outputs": ["./output/fire\_heatmap.png", "./output/route.json"]

}

**🎯 输出意义**

生成高层次的行动决策，形成面向指挥层的正式报告。

**🔄 系统整体流程**

用户输入（多模态）

↓

规划器（任务理解与分解）

↓

调用器（模型协同执行）

↓

总结器（汇总与决策生成）

↓

输出：图文结合的应急决策报告

**🧭 项目目标总结**

| **模块** | **核心功能** | **输入** | **输出** | **关键作用** |
| --- | --- | --- | --- | --- |
| 规划器 | 多模态意图理解与任务分解 | 文本、语音、图像 | 任务流程JSON | 定义要做什么 |
| 调用器 | 模型调度与执行 | 任务流程 | 模型结果JSON | 让任务落地执行 |
| 总结器 | 决策汇总与报告生成 | 模型结果 + 用户需求 | 决策报告 | 给出最终答案 |

✅ 最终系统形成了一个从理解 → 执行 → 决策的完整智能闭环，  
**以下是每个部分具体的实现细节**

**规划器**

**🧭 改进版规划器模块标准化技术文档**

（支持任务内多步骤结构）

**一、模块总览**

规划器（Planner）负责从用户的多模态输入中自动生成可执行的**层次化任务流程（Hierarchical Task Flow）**，供后续智能体协同执行。

整体流程：

多模态输入（文本/语音/图像）

↓

意图识别层 → 语义归一化层 → 任务结构生成层

↓

输出：分层任务流程（任务→步骤→模型执行）

**二、输入识别层（Intent Recognition Layer）**

**🎯 目标**

识别用户多模态输入中的显式意图、隐式意图与环境信息。

**🧠 模型与方法**

| **模块** | **模型/算法** | **说明** |
| --- | --- | --- |
| 文本意图识别 | ChatGLM3 / GPT-4 / Qwen-Max | 从语句中提取显式目标与任务对象 |
| 隐式意图推理 | Chain-of-Thought + Prompt Engineering | 根据上下文推理出隐含任务 |
| 图像环境识别 | SAM + GRIT + OWLv2 | 检测火灾区域、障碍物、人员分布 |
| 语音识别 | Whisper / FunASR | 语音转文本，提取地理和事件信息 |
| 融合层 | CLIP / BLIP2 特征对齐 | 将多模态描述映射到同一语义空间 |

**📥 输入格式**

{

"text": "请帮我规划南区火灾的救援路线，东侧通道已经堵塞。",

"image": "./inputs/fire\_scene.jpg",

"audio": "./inputs/report.wav"

}

**📤 输出格式**

{

"explicit\_intent": ["规划救援路线"],

"implicit\_intent": ["检测火源", "评估建筑稳定性"],

"environment": {

"location": "南区A栋",

"risk\_level": "high",

"obstacle": ["东侧通道堵塞"]

}

}

**三、语义归一化层（Semantic Normalization Layer）**

**🎯 目标**

融合多模态信息并将其抽象成可机读的**任务层结构**（Task Graph）。

**🧠 模型与方法**

| **功能** | **模型/算法** | **说明** |
| --- | --- | --- |
| 语义合并 | BERT / LLM结构化提示 | 对多模态内容进行统一表达 |
| 冲突消解 | 置信度加权投票 | 消除文本、图像、语音矛盾 |
| 任务关系推理 | 场景图生成（Scene Graph） | 构建任务节点和依赖关系 |
| 重要度评估 | 注意力权重分析 | 评估每个任务的优先级 |

**📥 输入格式**

上一步识别层输出的三类信息。

**📤 输出格式：任务图结构**

{

"tasks": [

{

"id": "T1",

"name": "fire\_region\_analysis",

"description": "识别火灾区域并计算热度图",

"depends\_on": [],

"priority": "high"

},

{

"id": "T2",

"name": "safe\_zone\_detection",

"description": "识别建筑内安全区域",

"depends\_on": ["T1"],

"priority": "medium"

},

{

"id": "T3",

"name": "path\_planning",

"description": "基于安全区生成救援路径",

"depends\_on": ["T2"],

"priority": "high"

}

]

}

**四、任务结构生成层（Task Structuring Layer）**

**🎯 目标**

在每个任务内生成详细步骤（Step），每个步骤对应一个可执行动作或模型调用。

任务内部结构化为：

Task → Step → Model

**🧠 模型与方法**

| **子模块** | **模型/算法** | **说明** |
| --- | --- | --- |
| 任务步骤生成 | LLM + JSON模板 | 将任务分解为若干步骤（Step） |
| 参数填充 | 规则模板 + 历史知识库 | 自动生成步骤参数 |
| 顺序规划 | 拓扑排序 + 层次控制 | 确定任务与步骤的执行顺序 |
| 格式验证 | JSON Schema Validation | 验证每个任务及步骤格式一致性 |

**📥 输入格式**

任务图结构（来自语义归一化层）

**📤 输出格式（层次化任务流程）**

{

"workflow": ["T1", "T2", "T3"],

"schema\_version": "v2.0",

"tasks": [

{

"task\_id": "T1",

"task\_name": "fire\_region\_analysis",

"description": "识别火灾区域并计算热度图",

"dependencies": [],

"steps": [

{

"step\_id": "T1\_S1",

"name": "image\_preprocessing",

"model": "OpenCV",

"input": {"image": "./inputs/fire\_scene.jpg"},

"output": {"path": "./temp/clean\_image.jpg"}

},

{

"step\_id": "T1\_S2",

"name": "region\_segmentation",

"model": "SAM",

"input": {"image": "./temp/clean\_image.jpg"},

"output": {"mask": "./output/fire\_mask.png"}

},

{

"step\_id": "T1\_S3",

"name": "heatmap\_generation",

"model": "GRIT",

"input": {"mask": "./output/fire\_mask.png"},

"output": {"heatmap": "./output/fire\_heatmap.png"}

}

]

},

{

"task\_id": "T2",

"task\_name": "safe\_zone\_detection",

"dependencies": ["T1"],

"steps": [

{

"step\_id": "T2\_S1",

"name": "obstacle\_mapping",

"model": "OWLv2",

"input": {"image": "./output/fire\_heatmap.png"},

"output": {"obstacle\_map": "./output/obstacle\_map.json"}

},

{

"step\_id": "T2\_S2",

"name": "safe\_zone\_estimation",

"model": "LLM-Reasoner",

"input": {"obstacle\_map": "./output/obstacle\_map.json"},

"output": {"safe\_zones": "./output/safe\_zone.json"}

}

]

},

{

"task\_id": "T3",

"task\_name": "path\_planning",

"dependencies": ["T2"],

"steps": [

{

"step\_id": "T3\_S1",

"name": "graph\_construction",

"model": "NetworkX",

"input": {"safe\_zones": "./output/safe\_zone.json"},

"output": {"graph": "./output/navigation\_graph.json"}

},

{

"step\_id": "T3\_S2",

"name": "path\_computation",

"model": "A\* Algorithm",

"input": {"graph": "./output/navigation\_graph.json"},

"output": {"path": "./output/route.json"}

},

{

"step\_id": "T3\_S3",

"name": "instruction\_generation",

"model": "GPT-4",

"input": {"path": "./output/route.json"},

"output": {"summary": "./output/instructions.txt"}

}

]

}

]

}

**五、格式验证标准（层次化Task Schema）**

{

"type": "object",

"properties": {

"task\_id": {"type": "string"},

"task\_name": {"type": "string"},

"description": {"type": "string"},

"dependencies": {"type": "array", "items": {"type": "string"}},

"steps": {

"type": "array",

"items": {

"type": "object",

"properties": {

"step\_id": {"type": "string"},

"name": {"type": "string"},

"model": {"type": "string"},

"input": {"type": "object"},

"output": {"type": "object"}

},

"required": ["step\_id", "model", "input", "output"]

}

}

},

"required": ["task\_id", "task\_name", "steps"]

}

**六、任务与步骤的关系说明**

| **层级** | **定义** | **示例** | **对应模型类型** |
| --- | --- | --- | --- |
| Task | 表示一个语义独立的功能单元 | 火源识别、路径规划 | LLM + 专用算法组合 |
| Step | 任务内部可执行动作 | 图像分割、路径计算 | SAM、A\*、GRIT等 |
| Model | 执行Step的底层工具 | PyTorch模块/API调用 | Vision/Language/Planning模型 |

**七、总结与优势**

| **特点** | **说明** |
| --- | --- |
| 层次清晰 | 每个任务包含可追踪的步骤 |
| 可扩展性强 | 新增模型只需定义新的Step模板 |
| 兼容性高 | 所有任务均遵循统一JSON格式 |
| 可追踪性好 | 系统可记录每个Step的输入/输出与耗时 |
| 易可视化 | 任务层→步骤层→模型层三维可展示 |

**八、最终执行流程概览**

多模态输入

↓

意图识别层 → 显式/隐式/环境提取

↓

语义归一化层 → 生成任务图

↓

任务结构生成层 → 各任务分解为步骤

↓

输出层次化任务流程（Task + Step）

↓

调度器按依赖执行 → 各智能体调用模型 → 汇总结果

**✅ 结论**

规划器输出的层次化任务流程（Hierarchical Task Flow）是整个系统的核心接口。  
每个任务拥有自己的步骤序列（Step List），每个步骤映射到具体模型执行单元。  
该标准不仅支持多智能体协同决策，还能实现任务级与步骤级的可追踪性与可扩展性。

**调用器**

**⚙️ 调用器（Executor）模块标准化技术文档**

（模型调用与共享意图一致性机制）

**一、模块定位与总体目标**

调用器（Executor）位于整个系统的执行层。  
它负责接收规划器输出的**层次化任务流程（Hierarchical Task Flow）**，  
并为每个任务及其内部步骤选择、调度、调用相应模型执行。

执行完成后，将所有模型输出统一汇入总结器（Aggregator），  
形成整体态势与决策建议。

**二、系统结构概览**

规划器输出（Task + Step）

↓

调用器（Executor）

├── 模型选择与调度（Model Dispatcher）

├── Prompt构建与共享意图注入（Prompt Builder）

├── 模型执行（Model Runner）

└── 输出管理与反馈（Output Manager）

↓

总结器（Aggregator）

**三、输入输出接口定义**

**📥 输入：来自规划器的任务结构**

{

"workflow": ["T1", "T2", "T3"],

"tasks": [

{

"task\_id": "T1",

"task\_name": "fire\_region\_analysis",

"steps": [

{

"step\_id": "T1\_S2",

"name": "region\_segmentation",

"model": "SAM",

"input": {"image": "./inputs/fire\_scene.jpg"},

"output": {"mask": "./output/fire\_mask.png"}

}

]

},

...

]

}

**📤 输出：统一结果集合（供总结器使用）**

{

"results": [

{

"task\_id": "T1",

"summary": "火源区域已定位于南区A栋，热度高风险区域半径约20米。",

"output\_files": ["./output/fire\_mask.png", "./output/fire\_heatmap.png"]

},

{

"task\_id": "T2",

"summary": "安全区在西北角，路径规划推荐北门出入口。",

"output\_files": ["./output/safe\_zone.json", "./output/route.json"]

}

],

"intent\_context": {

"goal": "优化救援路径与安全区域",

"shared\_variables": {

"location": "南区A栋",

"risk\_level": "high",

"priority": ["救援安全性", "时间效率"]

}

}

}

**四、模块结构与工作流程**

**1️⃣ 模型选择与调度（Model Dispatcher）**

**职责：**  
根据任务与步骤描述，从模型注册表中选取合适模型进行调用。

**实现机制：**

* 模型注册表（Model Registry）预定义任务-模型映射；
* 调度器根据任务类型自动匹配；
* 支持异步执行与批处理。

**示例映射表：**

{

"fire\_region\_analysis": ["SAM", "GRIT"],

"safe\_zone\_detection": ["OWLv2", "LLM-Reasoner"],

"path\_planning": ["A\*", "LLM-Planner"]

}

**2️⃣ Prompt 构建与共享意图注入（Prompt Builder）**

**职责：**  
在调用每个模型前，将上层的“共享意图变量（Shared Intent Variable）”注入Prompt中，  
以确保模型输出在目标、语义和上下文上一致。

**✅ 共享意图变量（Shared Intent Variable）**

这些变量由规划器或系统上下文生成，贯穿所有模型调用，确保：

* 输出统一语义目标；
* 模型间共享环境与任务背景；
* 避免冲突输出。

**示例：**

{

"shared\_intent": {

"global\_goal": "应急救援任务优化",

"location": "南区A栋",

"risk\_level": "high",

"task\_priority": ["安全性", "效率"],

"style": "正式汇报风格"

}

}

**Prompt 构建模板**

每次模型调用的Prompt格式：

[系统上下文]

你是一个应急智能体，负责执行 {task\_name} 任务。

[全局意图变量]

{shared\_intent}

[当前任务说明]

任务ID: {task\_id}

任务目标: {task\_description}

输入数据: {input\_info}

[输出要求]

请输出:

1. 模型分析结果

2. 结构化摘要（JSON格式）

3. 与全局意图一致的说明文字

示例调用（路径规划模型）：

你是应急调度智能体。任务: 生成救援路径。

全局意图：保证人员安全并最短时间抵达。

输入：安全区位置 safe\_zone.json。

输出：最佳路径坐标和理由说明。

**3️⃣ 模型执行（Model Runner）**

**职责：**  
执行具体的模型推理任务，并记录输出。

**执行方式：**

* 对于传统算法模型（如A\*、Dijkstra） → 直接运行Python函数；
* 对于深度模型（SAM、GRIT、OWLv2） → 载入checkpoint推理；
* 对于语言模型 → 调用LLM接口（如GPT-4 / Qwen / ChatGLM）。

**标准输出格式：**

{

"model\_name": "OWLv2",

"step\_id": "T2\_S1",

"status": "success",

"output": {

"file": "./output/obstacle\_map.json",

"summary": "识别出3个主要障碍区域"

}

}

**4️⃣ 输出管理与反馈（Output Manager）**

**职责：**

* 统一存储模型输出；
* 记录任务完成度与执行日志；
* 向总结器传递结构化结果。

**输出格式标准：**

{

"task\_id": "T1",

"completed\_steps": ["T1\_S1", "T1\_S2", "T1\_S3"],

"artifacts": [

"./output/fire\_mask.png",

"./output/fire\_heatmap.png"

],

"summary": "火势主要集中于南区A栋，建议立即疏散北侧通道。"

}

**五、总结器（Aggregator）接口规范**

**🎯 职责**

整合来自不同模型的结果，形成统一的语义报告与态势图输出。

**📥 输入**

调用器的结果集（含任务摘要与共享意图上下文）。

**📤 输出**

综合态势与行动建议：

{

"final\_summary": "南区A栋火势严重，推荐北门通行，需在西北角设置应急医疗点。",

"visual\_outputs": [

"./output/fire\_heatmap.png",

"./output/route.json"

],

"report": "./output/final\_briefing.txt"

}

**六、执行流程（全链路）**

1️⃣ 调用器读取层次化任务流程

↓

2️⃣ 为每个任务选择模型（Model Dispatcher）

↓

3️⃣ 构建Prompt并注入共享意图变量（Prompt Builder）

↓

4️⃣ 执行模型推理（Model Runner）

↓

5️⃣ 收集输出与摘要（Output Manager）

↓

6️⃣ 将所有任务结果交给总结器汇总

**七、共享意图变量在模型协同中的作用**

| **功能** | **说明** | **影响层级** |
| --- | --- | --- |
| 全局目标一致性 | 保证所有模型的输出目标一致 | 系统级 |
| 语义连贯性 | 模型在表述中共享背景信息（如位置、风险等级） | 模型级 |
| 可追踪性 | 通过共享变量追踪每个模型输出上下文 | 调度级 |
| Prompt上下文复用 | 减少重复描述，提高Prompt效率 | Prompt级 |

**八、模块接口结构总结**

| **子模块** | **输入** | **输出** | **核心算法** | **作用** |
| --- | --- | --- | --- | --- |
| 模型调度（Dispatcher） | 任务结构 | 模型调用计划 | 模型映射表 | 确定模型选择 |
| Prompt生成（Builder） | 任务描述 + 意图变量 | 统一Prompt文本 | 模板 + LLM填充 | 构建输入上下文 |
| 模型执行（Runner） | Prompt + 数据 | 模型输出 | 深度模型推理 / 算法执行 | 执行核心推理 |
| 输出管理（Manager） | 各模型输出 | 标准化结果JSON | 日志记录 + 路径管理 | 输出整合与追踪 |

**九、最终输出规范（统一接口）**

系统的所有结果汇总为：

{

"executor\_summary": {

"tasks\_completed": ["T1", "T2", "T3"],

"overall\_status": "success",

"shared\_intent\_context": {

"goal": "应急救援优化",

"location": "南区A栋",

"priority": ["安全", "效率"]

},

"aggregated\_report": "./output/final\_briefing.txt",

"visual\_assets": [

"./output/fire\_heatmap.png",

"./output/route.json"

]

}

}

**🔍 十、总结**

✅ 调用器是将“规划器输出的任务结构”转化为**实际模型执行与推理的核心模块**。  
✅ 每个模型在调用时通过**共享意图变量**保持任务目标一致，避免输出割裂。  
✅ 执行结果由输出管理器统一封装，最终交由总结器整合生成全局态势。  
✅ 整个系统形成了：

规划器（Plan） → 调用器（Execute） → 总结器（Summarize）

三层闭环式多智能体协同决策结构。

**总结器**

**🧠 总结器（Aggregator）模块标准化技术文档**

（基于任务结果的多模态汇总与决策生成）

**一、模块定位与总体目标**

总结器是整个系统的**决策与综合层（Decision Layer）**。  
其主要职责是：

1. 接收调用器（Executor）输出的各任务结果；
2. 根据用户原始需求（Intent）构建决策级Prompt；
3. 调用大型语言模型（LLM）综合多模态结果；
4. 输出结构化的决策建议与态势总结。

✅ 核心理念：**从“任务执行结果”生成“高层次决策洞见”**

**二、总体数据流结构**

用户输入（文本/图像/语音）

↓

规划器（Planner） → 任务结构

↓

调用器（Executor） → 各模型结果

↓

总结器（Aggregator）

↓

综合态势报告 / 决策建议 / 图文输出

**三、输入与输出接口**

**📥 输入：来自调用器的结果集**

{

"results": [

{

"task\_id": "T1",

"summary": "火源已定位于南区A栋东侧。",

"output\_files": ["./output/fire\_mask.png", "./output/fire\_heatmap.png"]

},

{

"task\_id": "T2",

"summary": "安全区位于西北角。",

"output\_files": ["./output/safe\_zone.json"]

},

{

"task\_id": "T3",

"summary": "推荐救援路线为北门→楼梯A→安全区。",

"output\_files": ["./output/route.json"]

}

],

"intent\_context": {

"goal": "为应急指挥中心生成最优救援决策报告",

"shared\_variables": {

"location": "南区A栋",

"risk\_level": "high",

"priority": ["安全", "时间效率"]

}

},

"user\_query": "请告诉我当前最优的救援方案，并输出图文态势报告。"

}

**📤 输出：决策级汇总结果**

{

"final\_summary": {

"situation\_overview": "南区A栋火势严重，火源集中于东侧；安全区位于西北角；救援路线建议由北门进入。",

"recommended\_actions": [

"立即派遣消防队从北门进入；",

"在西北角设置临时医疗点；",

"加强东侧外围隔离以防火势蔓延。"

],

"risk\_assessment": {

"fire\_intensity": "High",

"evacuation\_risk": "Moderate"

}

},

"visual\_outputs": [

"./output/fire\_heatmap.png",

"./output/route.json"

],

"report\_file": "./output/final\_report.txt"

}

**四、模块组成与功能说明**

┌──────────────────────────────┐

│ Summarizer（总结器） │

├──────────────────────────────┤

│ ① 输入整合模块（Input Merger） │

│ ② 用户意图分析模块（Intent Rebuilder） │

│ ③ 决策Prompt构建器（Decision Prompt Builder） │

│ ④ LLM汇总与生成模块（LLM Decision Synthesizer） │

│ ⑤ 输出结构化模块（Report Formatter） │

└──────────────────────────────┘

**① 输入整合模块（Input Merger）**

**功能：**

* 读取来自调用器的所有任务结果；
* 将文本结果、结构化JSON、图像路径统一转为多模态摘要；
* 形成统一输入上下文（Unified Context）。

**示例输出：**

{

"context\_text": "T1: 火源已定位于东侧；T2: 安全区西北角；T3: 建议北门进入。",

"context\_files": ["fire\_heatmap.png", "route.json"],

"shared\_intent": {

"goal": "生成救援指令",

"location": "南区A栋"

}

}

**② 用户意图分析模块（Intent Rebuilder）**

**功能：**

* 重新解析用户最初输入（user\_query）；
* 与系统共享变量（shared\_intent）融合；
* 生成“高层意图目标”（High-level Goal），指导后续汇总。

**输出示例：**

{

"user\_goal": "为应急指挥中心生成综合决策建议",

"expected\_output": "包含图文态势与指挥建议的总结"

}

**③ 决策Prompt构建器（Decision Prompt Builder）**

**功能：**

* 构建面向LLM的决策生成Prompt；
* 动态注入：
  + 用户输入；
  + 各任务结果；
  + 共享意图变量；
  + 输出格式约束。

**Prompt 模板示例：**

你是一个应急指挥智能体。

【用户目标】

{user\_goal}

【系统上下文】

地点：{shared\_intent.location}

风险等级：{shared\_intent.risk\_level}

任务优先级：{shared\_intent.priority}

【模型结果汇总】

{context\_text}

【用户需求】

{user\_query}

【输出要求】

请基于以上信息生成：

1. 统一的态势总结；

2. 明确的行动决策建议；

3. 结构化JSON输出；

4. 语气正式，面向应急指挥汇报。

**④ LLM 汇总与生成模块（LLM Decision Synthesizer）**

**功能：**

* 调用大语言模型（GPT-4 / Qwen-Max / InternLM-Chat）；
* 对多模态输入执行逻辑整合与推理；
* 生成结构化决策内容与自然语言总结。

**执行过程：**

response = llm.generate(prompt=decision\_prompt)

parsed = json.loads(response)

**输出内容：**

* “situation\_overview”：整体态势；
* “recommended\_actions”：决策建议；
* “risk\_assessment”：风险等级；
* “visual\_mapping”：需要展示的文件路径。

**⑤ 输出结构化模块（Report Formatter）**

**功能：**

* 将LLM输出包装为标准化JSON；
* 同步保存文本报告与图文态势文件；
* 输出路径注册至系统日志。

**输出规范：**

{

"report\_id": "R001",

"summary": "火源集中东侧，建议北门进入。",

"actions": [

"派遣消防队由北门进入",

"在西北角设医疗点"

],

"risk": "High",

"generated\_at": "2025-10-24T18:00:00",

"attachments": ["fire\_heatmap.png", "route.json"]

}

**五、Prompt 构建逻辑与智能融合策略**

**✅ 1. 用户需求驱动（User-centered Prompting）**

总结器的生成过程**必须围绕用户初始输入构建**，而不是仅依赖中间任务结果。  
这保证了最终报告真正解决“用户提问”，而非“系统任务本身”。

**✅ 2. 共享意图变量强化一致性**

所有任务执行阶段传下来的 **shared\_intent** 在总结阶段重新注入：

* location、goal、priority 确保语义对齐；
* 风格参数（style: “正式汇报”）控制输出体裁；
* 上下文标识符（context\_id）保证多轮决策一致性。

**✅ 3. 多模态结果整合策略**

| **模态** | **整合方法** | **输出目标** |
| --- | --- | --- |
| 文本摘要 | LLM合并 + 去冗余 | 形成语义统一的态势说明 |
| 图像结果 | 文件路径→描述映射（caption） | 在报告中插入视觉说明 |
| 结构化数据 | JSON合并与数值统计 | 生成数据化结论（如面积、距离） |

**六、最终输出规范（供展示或上传）**

标准输出文件：

/output/

├─ final\_report.txt ← LLM自然语言总结

├─ final\_summary.json ← 结构化结果

├─ fire\_heatmap.png

├─ route.json

└─ summary\_visual.html ← 可视化页面（可选）

**七、执行流程总结**

调用器输出 → 输入整合模块

↓

用户意图重构

↓

决策Prompt构建

↓

LLM汇总推理

↓

输出标准化与文件生成

**八、模块协同闭环总结**

| **模块** | **主要输入** | **核心逻辑** | **输出** |
| --- | --- | --- | --- |
| 规划器（Planner） | 用户多模态输入 | 任务与步骤结构化 | 层次化任务流程 |
| 调用器（Executor） | 任务流程 | 模型调度与执行 | 模型结果集合 |
| 总结器（Aggregator） | 模型结果 + 用户需求 | LLM综合推理生成 | 决策与报告输出 |

**三层形成完整闭环：**

**理解（Planner） → 执行（Executor） → 生成决策（Aggregator）**

**九、总结器的价值**

| **维度** | **描述** |
| --- | --- |
| 决策智能 | 从执行层输出生成高层决策 |
| 用户导向 | 直接响应用户的原始意图 |
| 多模态融合 | 融合图像、文本、结构化数据 |
| 可解释性 | 输出中包含各任务摘要与行动依据 |
| 可扩展性 | 可根据任务类型动态选择Prompt模板 |

**🔚 十、结论**

✅ 总结器是系统的决策生成中心。  
它通过整合调用器的多模型输出、共享意图变量与用户需求，  
利用大语言模型进行高层语义推理，  
最终生成形式化、可视化、可执行的综合应急决策输出。