消歧交互模块

```
如果 不确定性 > 阈值:
    触发澄清
否则:
    直接执行
```

这是在原有的DINO基础上加了一个反馈，或者说中途补全机制

```
用户指令："把那个螺丝拧到那里"
            |
            ▼
    ┌──────────────────┐
    | Ambiguity        |  → 检测到"那个"和"那里"有歧义
    | Detection        |     不确定性 = 0.8 > 阈值0.5
    └──────────┬───────┘
               | 触发澄清
               ▼
    ┌──────────────────┐
    | 计算 Information |  → 问颜色：增益0.9
    | Gain             |     问大小：增益0.3
    └──────────┬───────┘     问材质：增益0.0
               | 选增益最大的
               ▼
    ┌──────────────────┐
    | Clarifying       |  → "您指的是哪个颜色的螺丝？"
    | Question         |
    └──────────┬───────┘
               | 用户回答："红色的"
               ▼
          不确定性降低，继续执行或再问一轮
```

机器人选择信息增益最大的方向去问问题，然后人对其进行回答与澄清

> 在工业场景中，这个阈值的设定很关键：
> - **阈值太低**（太容易触发）→ 机械臂老是问，效率低，工人烦
> - **阈值太高**（太难触发）→ 机械臂经常猜错，误操作，可能损坏零件

所以这个前面项目跑的这个阈值的设计是用来触发澄清的

```
> 好问题 = 能把候选"劈"成尽量均匀的几份
```

3个候选，问完能变成：
- 1个 vs 2个 → 不错
- 1个 vs 1个 vs 1个 → 最好（完美三分）
- 3个 vs 0个 → 白问（没有区分）

```
### 与你课题的关系
```

熵的降低和属性的完美归类，我好像在这个机器学习中看过这个信息增益的概念

关于澄清和评估标准

设计问题	核心思路	工业场景考量
何时触发	Top-K置信度接近时	阈值要根据安全要求调整
问什么	选信息增益最大的	问题要让工人容易回答
评估-轮数	越少越好	流水线效率
评估-成功率	越高越好	最终目标
评估-误澄清	越低越好	避免把工人问懵

Lin 等 - 2025 - Ask-to-Clarify Resolving Instruction Ambiguity through Multi-turn Dialogue论文的阅读

```
问题：VLM 输出的是"语言token"
      Diffusion 需要的是"视觉条件"
      两者格式不匹配！

解决：连接模块做"翻译"
      把语言信息融合到视觉观测中
      生成 Diffusion 能理解的条件
```

翻译，解码，转换，这些东西在很多论文中看到了

技术实现：FiLM（Feature-wise Linear Modulation，特征级线性调制）

FiLM 的直觉理解：

```python
# 伪代码
def FiLM(visual_feature, language_info):
    # language_info 生成两个参数：缩放因子γ 和 偏移因子β
    gamma = linear1(language_info)  # 缩放："放大/缩小某些特征"
    beta = linear2(language_info)   # 偏移："整体调整"

    # 用语言信息去"调制"视觉特征
    output = gamma * visual_feature + beta
    return output
```

生活比喻：

> 假设你戴了一副"智能眼镜"，当听到"红色螺丝"时，眼镜会自动高亮红色物体—这就是语言"调制"视觉

根据文本放大一些特征是吗，真是有意思的一个技术

FiLM 调制过程：

- 提取"红色螺丝"相关的语义
- 在视觉特征中"高亮"红色螺丝的位置
- 弱化其他物体的特征

输出：调制后的视觉条件

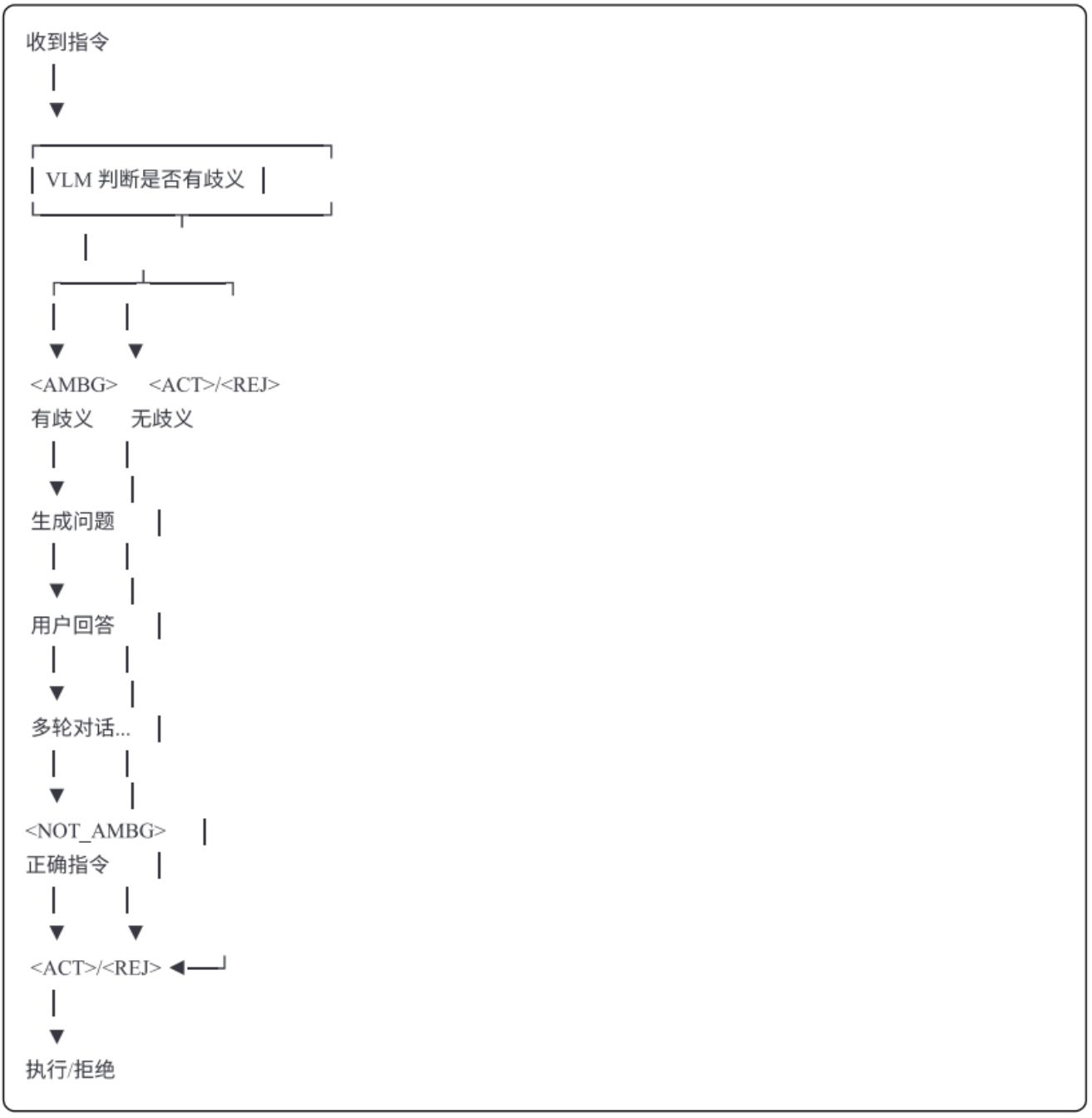
（可以理解为：一张"注意力集中在红色螺丝上"的特征图）

这个东西和这个注意力集中机制也有关系吗

Token	中文	含义	触发时机
<AMBG>	有歧义	指令模糊，需要提问	检测到歧义时输出
<NOT_AMBG>	无歧义	已推断出正确指令	对话结束，歧义消除
<ACT>	执行	可以开始动作	目标物体在视野中
<REJ>	拒绝	不执行	目标物体不在视野中

这篇论文其实很简单，总共也就分了两个模块，一个大脑，一个机械臂执行

状态转移图：



真是非常棒的一个状态转移图

为什么要"知识隔离"？

问题：如果 Stage 2 继续训练 VLM，会发生什么？
答案：灾难性遗忘（Catastrophic Forgetting）！
VLM 会忘记 Stage 1 学到的对话能力。

解决：冻结 VLM，保护对话知识
用 Connection Module 补偿 VLM 和 Diffusion 之间的连接

好一个隔离，从哲学上来讲是降低了这个熵，从人的特征来讲，人在学习一类的知识的过程中，后学的正确知识会覆盖掉前面学习的知识

一、Signal Token 机制 → 设计歧义检测触发器

Ask-to-Clarify 是怎么做的?

用 4 个特殊 token 控制系统状态：

```
<AMBG>      → 检测到歧义，进入提问模式
<NOT_AMBG>  → 歧义已消除，得到明确指令
<ACT>       → 可以执行
<REJ>       → 拒绝执行（目标不在视野中）
```

信号令牌，就是告诉你，什么时候触发这个检测机制，什么时候开始进行具体操作的执行，接受信号的意思是吗，真有趣

```
prompt = """
你是一个工业场景对话数据生成器。

场景描述：
{scene_description}
# 例如：工作台上有3颗M6螺丝（左、中、右），2颗M8螺丝，1个电动螺丝刀

物体列表：
{object_list}
# 例如：[M6螺丝-左，M6螺丝-中，M6螺丝-右，M8螺丝-前，M8螺丝-后，电动螺丝刀]

请生成：
1. 一条模糊指令（包含"那个"、"这个"、"那里"等指代词）
2. 机器人应该问的澄清问题
3. 用户的回答
4. 最终的明确指令

要求：
- 问题要针对工业场景的区分属性（位置、规格、工位号）
- 问题要简洁，工人能快速回答
- 考虑工业场景的专业术语
```

这个prompt模版的设计，需要包含前面的背景设置部分，在此基础上，再根据后面的模版约束生成这个具体的模版

该论文的启发总结：四个启发的行动清单

设计点	你可以做的具体行动
Signal Token 机制	设计工业版状态机，增加 `<AMBG_ACT>`、`<UNSAFE>` 等工业特有状态

设计点	你可以做的具体行动
数据驱动问句生成	用 LLM 批量生成工业场景对话数据，覆盖规格、工位、参数等歧义类型
两阶段知识隔离	先训消歧，再冻结消歧模块训执行，防止能力遗忘
FiLM 特征融合	用 FiLM 把指令语义注入视觉特征，让模型"关注"正确的物体

图表	页码	内容	重要性
Figure 1	p.1	Executor vs Collaborator 对比	⭐⭐⭐
Figure 2	p.3	两阶段训练流程图	⭐⭐⭐

这个协作模式就是通过这个大语言模型进行反问消歧

```
VLM 输出:
  文本: "把橙子放到盘子里"
  信号: <NOT_AMBG>

Signal Detector 检测到 <NOT_AMBG>:
  → 进入 correct ins. (正确指令) 分支
  → 提取出: "把橙子放到盘子里"
  → 把正确指令喂回 VLM，判断能否执行

【第3轮】
──────────────────────────────────────────────

正确指令 → 再输入 VLM
  图像: 🍎🍑🍊🍽️
  指令: "把橙子放到盘子里"

VLM 检查:
  "橙子在视野中吗? → 在! "

VLM 输出:
```

检测到这个正确的指令后，还需要这个将这个正确的指令喂给大脑，然后让大脑判断这个指令是否可以执行

概念层面

1）“歧义检测的阈值”到底怎么定？

你在问的本质：
系统什么时候应该“停下来问一句”，什么时候应该“直接干”？

通俗比喻：
你让同学“把那个杯子拿过来”。如果桌上只有一个杯子，同学就直接拿；如果桌上有 3 个杯子，同学会反问：“你指哪个？”
这个“要不要反问”的分界线，就是“阈值”。

一般会怎么做（两条路）：

- **显式规则（手工阈值）**：模型有个“不确定度/置信度”，比如候选目标前两名分数差太小，就判为“有歧义”，触发提问。
- **隐式学习（学出来的阈值）**：直接训练一个“是否歧义”的分类器/判别头，让模型从数据里学会“像这种情况就该问”。

工业机械臂例子：
“拿左边那个螺丝”——左边有两颗很像的螺丝。你需要一个机制判断：“我现在分不清 → 先问”。

两种阈值一种是学出来的阈值，一种是直接定义的阈值，根据这个置信度的差值

2）“问句的信息增益”没显式算，够不够？

你在问的本质：
问的问题是不是“真有用”？能不能**最大程度减少不确定性**？

通俗比喻：
你问“你喜欢什么颜色？”对“拿哪个杯子”帮助不大；
你问“你要左边那个还是右边那个？”一下就把候选缩到 1 个。
后者信息增益更高。

两种主流做法：

- **显式信息增益**：把当前可能答案当作一个集合/分布，问一句后期望把集合缩小最多（数学上像“熵下降”）。
- **数据驱动**：不算公式，直接用大量对话示例训练“该问什么”，用成功率/轮数等指标间接保证“问得值”。

工业机械臂例子：
与其问“你要哪个螺丝？”（太宽泛），不如问：

- “是靠近电机那颗，还是靠近夹具那颗？”（直接二选一）
- “要拧到贴合就停，还是要上紧到规定扭矩？”（把动作参数歧义一次问掉）

直接计算和数据驱动，真是两种精妙的方法，一种是通过这个深度学习，一种是通过这个逻辑数学公式