Problem Statement

With the announcement of GPT-4o, which supports text, video, image, and audio inputs, we expect an increased interest among developers in incorporating multimodality support in AutoGen for their applications. The core challenge is determining the most effective way for agents with different modalities to communicate seamlessly with each other.

Objective

Explore the current state of multimodality support in AutoGen, identify existing challenges, and propose a potential solution to take multimodality support in AutoGen to the next step.

Current State

In the main branch of AutoGen, there's the `MultimodalConversableAgent` that can "understand" images. This agent detects HTML-style tags (e.g., <image some_image.jpg>) within text messages, loads the image, and converts it to OpenAI's image API standard for processing. This approach enables basic multimodal interaction, allowing the system to process and respond to image data embedded in text messages.

Limitations of the current approach:

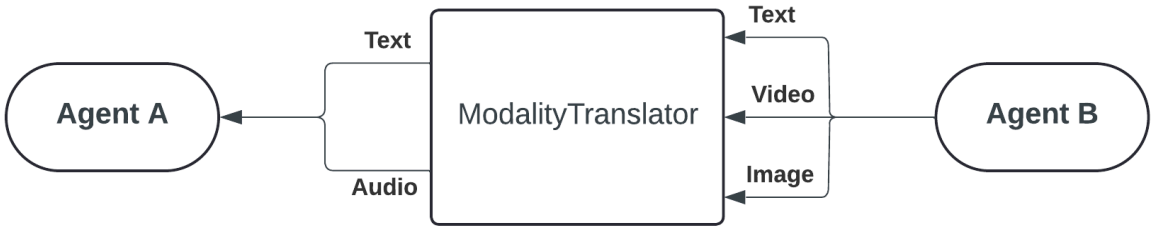
- No support for video.
- No support for audio.
- Use of HTML-style tags: could interact negatively with other components of AutoGen (e.g. `WebSurferAgent`).
- Lack of extensibility: The current approach is not easily extensible to other types of agents or modalities. Only the `MultimodalConversableAgent` can interpret the specific HTML tags, limiting flexibility and making it challenging to adapt to new modalities.

Assumptions

1. **Text as the Common Denominator:**
    a. All LLMs fundamentally operate on text data. This shared characteristic makes text the common denominator for multimodal interactions.
    b. Regardless of the input or output modality (e.g., text, audio, image, video), converting data to and from text ensures compatibility across different agents and models.
2. **Tag System for Multimodal Communication:**
    a. To support multimodality, all agents in AutoGen should be capable of communicating through different modalities using text-based representations. A tag system can facilitate this process.
    b. Tags (e.g., <image>, <audio>, <video>) can be embedded within text messages to indicate the presence of non-text data. This allows agents to recognize, process, and generate responses involving various modalities.
    c. Example: An image file could be referenced within a text message as `<image src="path/to/image.jpg">`, enabling the receiving agent to identify and handle the image appropriately.
    d. HTML-style tags might not be the solution, but I'll propose a new tagging style in the solutions section.
3. **Modality-Specific Message Types:**
    a. LLM providers (e.g., OpenAI, Google, Claude) have varied API standards. Modality-specific message types can be tailored to match these standards in the model client.
    b. Modal message types don’t address the integration of modalities within the text message. For example, embedding an audio file within a text message requires a different approach.
    c. This still leads to a tag system solution, as it provides a way to embed different modalities within text messages.

Potential Solution

I propose the `ModalityTranslator` agent capability (I need suggestions for better names). This capability hooks onto the `process_all_messages_before_reply` method and translates all messages into a format the receiving agent can understand.



Here's how it works in practice:

- Agent A:
    - Only understands text.
    - Has the `ModalityTranslator` capability
- Agent B:
    - Can output text and/or images.
- Agent B sends an image of blue skies with text saying "Hello!".
- Agent A receives this message.
- Before generating a reply, the `ModalityTranslator` capability translates the image into a text description since Agent A can only understand text.
- The message turns into: "Hello! (you received an image of blue skies)".

Sounds familiar? The idea behind `ModalityTranslator` was modeled after `VisionCapability` in Beibin's PR https://github.com/microsoft/autogen/pull/2025.

Let's look at this second example:

- Agent A:
    - Understands text and audio.
    - Has the `ModalityTranslator` capability.
- Agent B:
    - Only understands text.

- Agent B sends a message saying, "Hey, what does this audio say? {% tag='audio' src='some_audio.mp3' %}".
- Agent A receives this message.
- Before generating a reply, the `ModalityTranslator` capability finds the audio tag and converts it to the OpenAI message standard.
- The new message turns into {"content": [{"text": "Hey, what does this audio say?"}, {"audio": "some_audio_format"}]}.

In this example, I also introduced a new tagging style `{% %}` (similar to Jinja's), which doesn't interact with other markdown types.

`ModalityTranslator` will take these arguments in the constructor:

- image_to_text: Optional[ImageToText]
- video_to_text: Optional[VideoToText]
- audio_to_text: Optional[ImageToText]

We do not need text_to_* as I assume we'll be using a tag system.

Other Thoughts

Why can't it be another transform?

- It can be, `TransformMessages` was designed with this use case in mind. However, we do lose some information like modalities the LLM supports. However, it can become too verbose when setting up `ModalityTranslator` as a transform, since you need to instantiate a new transform for each one of your agents (assuming they all support different modalities).
- If `ModalityTranslator` was an agent capability instead, the dev can set up one `ModalityTranslator` for all their agents and can dynamically detect which modality to translate.