動画を用いた料理手順の動詞の機械翻訳

竹藤 樹　高田 一慶　後藤 功雄　二宮 崇
愛媛大学
{takefuji, takatak}@ai.cs.ehime-u.ac.jp
{goto.isao.fn, ninomiya.takashi.mk}@ehime-u.ac.jp

概要

本研究では，料理手順動画の手順説明文に頻出する多義動詞により，テキストのみや静止画を追加した翻訳では適切な訳語選択が困難になる問題に取り組む．動画は時間軸情報を含むため，動作の認識を助け，動詞の意味推定を改善できると考え，手順説明文と動画を入力とするマルチモーダル機械翻訳を提案する．まず，予備実験として GPT4o によるテキスト翻訳を分析し，視覚情報がないと誤訳や揺らぎが生じやすい 14 動詞を選定し，YouCook2-JP から抽出したものをデータセットとして構築した．実験より，画像を用いた場合に比べて動画を用いる有効性を確認した．

1　はじめに

近年，YouTube などの動画共有サービスの普及により，料理動画のような手順を説明する動画（手順動画）を視聴する機会が大きく増えている．そのため，教育や実用コンテンツとして手順動画が世界中で利用されており，多言語化の需要が高まっている．一方で，手順動画のテキストのみを用いた機械翻訳や，画像を利用した機械翻訳では，手順で重要な動作を表す語，特に動詞の訳語を適切に選択することが困難な場合があり，特に料理動画の翻訳において問題となっている．

料理動画の翻訳が難しい理由として以下の 3 点が挙げられる．１点目は、料理手順では同じ動詞でも文脈によって意味が大きく変わるためである．例えば英語の cook は，日本語では「焼く」「炒める」「煮る」「蒸す」「炊く」「ゆでる」「揚げる」など，具体的な調理法に分かれる．この訳語選択を誤ると，手順の正しい理解ができなくなる．2 点目は，視覚情報への依存である．多くの説明文は動画を見ている前提で書かれていて，目的語や細かい描写が省略される傾向にある．そのため，テキストの情報のみでは，どの調理動作を指しているのかを一意に決めにくく，適切な訳語を選べない場合がある．3 点目に，静止画情報の限界である．マルチモーダル機械翻訳 [1] では，視覚情報として静止画（1 枚画像）を入力に用いる手法が広く採用されている．しかし，料理手順動画のような動作理解が重要なタスクでは，静止画のみでは十分に状況をとらえられない可能性がある．例えば，高田らの研究 [2] は，静止画から画像説明文を中間表現として生成し，それを翻訳に利用する手法を提案している．この手法は Chain-of-Thought（CoT）[3] を明示的に導入し，翻訳時の推論過程を補助する点に特徴がある一方で，扱う視覚情報は静止画に限られ，動画は利用していない．そのため，静止画では時間軸の情報が欠落し，動きの速さや継続性，動作の前後関係といった要素をとらえにくい．結果として，動詞の訳語選択に誤りが生じる要因となり得る．

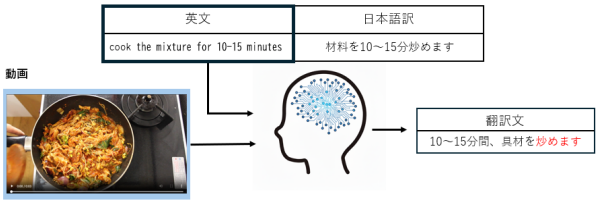

図 1: 提案手法の概要

本研究では，これらの課題に対応するため，動画の視覚情報を使うことで動作を表す動詞をより適切に推定するマルチモーダル機械翻訳手法を提案する．料理手順動画の動詞に着目したデータセットの構築を行い，訳語選択が必要な動詞を含む文の翻訳性能を評価する．実験では，動画を用いたマルチモーダル機械翻訳の有効性を確認した．

2　提案手法

本研究では，料理手順文に含まれる動詞の意味曖昧性を低減し，より適切な訳語選択を行うために，

動画情報を利用したマルチモーダル機械翻訳手法を提案する．料理動画においては，動詞が表す調理動作が映像として明示されている場合が多く，テキストのみでは判別が困難な動作の違いを視覚情報から補間できると考えられる．

2.1 動画を用いたマルチモーダル翻訳

提案手法では，料理動画の英語手順文と，それに対応する動画を同時に入力として用いる．具体的には，翻訳対象となる英文と，その文が発話される時間帯に対応する動画をモデルに入力する．

図 1 に示すように，モデルは英文情報と動画情報を統合的に処理し，調理中の食材の状態や調理器具の使用状況などの視覚的手がかりを考慮することで，動詞が表す具体的な動作を推定する．例えば「cook the mixture for 10-15 minutes」という英文に対して，動画中でフライパンを用いた撹拌・加熱動作が確認できる場合，「調理する」ではなく「炒める」といった具体的な訳語を選択することが可能となる．

2.2 翻訳生成プロセス

翻訳生成は，以下の流れで行う．

1. 翻訳対象となる英文を入力として与える．
2. 同時に、対応する料理動画を入力として与える．
3. モデルが動画の視覚情報を参照し、調理動作や状況を把握する．
4. 英文と視覚情報を考慮した日本語訳文を生成する．

これにより，料理動画翻訳における動詞の誤訳を低減し，翻訳品質の向上を目指す．

3 データセット構築

本研究では，料理手順文に含まれる動詞の多義性に着目し，動画情報の有効性を検証するための動詞に着目したデータセットを構築した．本節では，使用したコーパス，対象動詞の選定，および翻訳例の整理方法について述べる．

3.1 コーパス

本研究では，料理動画を対象とした英日対訳コーパスである YouCook2-JP[1)] を利用する．YouCook2-JP は，英語の料理動画データセット YouCook2[2)] [4] に対して日本語を付与したものであり，動画と料理手順が時間的に対応付けられている点に特徴がある．

データの構築に際し，YouCook2-JP に含まれる料理手順文のテキストを対象とし，その中から動詞の多義性が問題となりやすい文を抽出する．なお，検証においては，翻訳モデルの学習には用いず，分析及び評価目的でのみ使用している．

3.2 対象動詞選定のための予備実験

まず，対象とする動詞を選定するため，GPT-4o[5] を用いたテキストのみ入力による翻訳結果を分析した．ここでは，文脈情報や視覚情報が与えられない条件下において，料理手順文中の動詞がどのように翻訳されるかを確認し，適切な訳語選択が困難な動詞を特定することを目的とした．

予備実験では，YouCook2-JP に含まれる料理手順文を対象とし，各文に含まれる動詞について，翻訳結果の誤りや意味の抽象化，誤訳の揺らぎを分析した．文脈や視覚情報がなければ動作内容を正しく解釈することが難しく，誤訳が生じやすい動詞を整理した．

予備実験より，動作の区別がテキストのみでは困難であり，視覚情報の利用が不可欠と判断した以下の 14 語の動詞を選定した．

boil, coat, cook, drizzle, fold, fry, move, roll, score, slice, spread, sprinkle, toss, whisk

これらの動詞は，調理方法，手の動き，調理器具の使用状況などを視覚的に確認することで初めて適切な訳語を選択できる点で共通している．

3.3 データセット

本データセットは，YouCook2-JP に含まれる料理手順のうち，対象とする 14 動詞を含む文を抽出したものである．YouCook2-JP は全 10,337 文から構成される．本研究では，このうち誤訳が生じやすい 14 動詞を含む文を抽出し，訓練用 1,128 件，検証（評価）用 1,135 件に分割して実験に用いた．

4 評価実験

料理手順文に含まれる動詞の訳語選択において，動画情報を利用することの有効性を検証するため，評価実験を行った．本章では，実験設定，評価指標，実験結果，および分析について述べる．

1) https://github.com/nlab-mpg/YouCook2-JP

2) http://youcook2.eecs.umich.edu/

表 1: 翻訳結果（0-shot / few-shot）

(a) Qwen3-VL

入力	Prec.	Rec.	F1	BLEU	COMET
0-shot					
英文のみ	0.5642	0.4101	0.4618	28.184	0.8741
英文+画像	0.5976	0.5110	0.5371	29.018	0.8814
英文+動画	**0.6732**	**0.5810**	**0.5957**	**31.050**	**0.8871**
few-shot					
英文のみ	0.7207	0.7087	0.7047	37.380	0.8942
英文+画像	**0.7299**	**0.7110**	**0.7131**	37.172	0.8965
英文+動画	0.7253	0.7060	0.7082	**37.867**	**0.8986**

(b) Qwen3-Omni

入力	Prec.	Rec.	F1	BLEU	COMET
0-shot					
英文のみ	0.5727	0.4060	0.4606	30.425	0.8889
英文+画像	0.6857	0.5547	0.5956	34.668	0.8955
英文+動画	**0.7161**	**0.6465**	**0.6653**	**37.026**	**0.8998**
few-shot					
英文のみ	0.7464	0.7084	0.7182	40.784	0.9046
英文+画像	0.7620	0.7210	0.7320	41.478	0.9041
英文+動画	**0.7628**	**0.7320**	**0.7377**	**42.227**	**0.9057**

4.1 実験設定

第 3 節で構築したデータセットを用いて評価を行った．翻訳モデルとして，近年高い性能を示している Vision-Language Model（VLM）を 2 種類用いた．具体的には，Qwen3-VL[6] および Qwen3-Omni[7] である．これらのモデルはいずれも，テキストと視覚情報を同時に入力として扱うことが可能であり，料理動画のような視覚依存のあるタスクに適している．80 億パラメータを持つ Qwen3-VL-8B-Instruct[3)]，および 300 億パラメータを持つ Qwen3-Omni-30B-A3B-Instruct[4)] を利用した．

各モデルについて，翻訳条件として 0-shot 翻訳および few-shot 翻訳を実施し，0-shot 翻訳では，翻訳対象となる英文および手法に応じて画像と動画を追加したもののみをを入力とし，例文を与えずに翻訳を行った．一方，few-shot では，翻訳対象文の前に少数の例文を与えた．具体的には，各対象動詞ごとに，出現する訳語の種類を網羅することを目的として例文を選定した．例文の選定にあたっては，同一動詞内で同じ訳語が重複して掲示されないようにし，各訳語につき原則 1 例のみを採用した．その結果，few-shot で提示する例文数は動詞によって異なり，1 から 11 例の範囲となった．なお，データセット中に特定の訳語が 1 文しか出現しない場合など，すべての訳語に対して例文を用意できないケースがあり，一部の訳語は few-shot 例に含まれない．さらに，few-shot 例は手法間で条件をそろえるため，画像入力手法・動画入力手法いずれにおいても例文部分には視覚情報を付与せず，テキストのみを提示した．

入力条件として以下の 3 条件で翻訳を行った．付録の図 3 にプロンプトを示す．

- テキストのみ入力：英文手順文のみを入力として与える．
- テキスト＋画像入力：英文手順文と対応する料理動画区間内の中央の単一画像を同時に入力として与える．
- テキスト＋動画入力：英文手順文と対応する料理動画区間の動画を同時に入力として与える．

これにより，0-shot と few-shot の違いに加え，視覚情報の種類（画像・動画）が訳語選択に与える影響を比較可能とした．

4.2 評価指標

翻訳結果の評価には，動詞に着目した評価指標と，文全体を対象とした評価指標の 2 種類を用いた．これにより，翻訳品質を多角的に分析できるようにした．

動詞に着目した評価　本研究の目的は，料理手順文に含まれる動詞の訳語選択が，動画情報の利用によってどの程度改善されるかを明らかにすることである．そのため，文全体の評価に加えて，動詞単位での評価を行った．翻訳結果における動詞の適切性を評価するために，Precision，Recall，および F1-score を用いた．

$$\text{Precision} = \frac{\text{正しく翻訳された動詞の数}}{\text{生成文に含まれる全動詞数}} \tag{1}$$

$$\text{Recall} = \frac{\text{正しく翻訳された動詞の数}}{\text{参照文に含まれる全動詞数}} \tag{2}$$

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

上式の評価により，文全体としては意味が通っている場合であって，料理動詞の具体的な意味が正し

3） https://huggingface.co/Qwen/Qwen3-VL-8B-Instruct
4） https://huggingface.co/Qwen/Qwen3-Omni-30B-A3B-Instruct

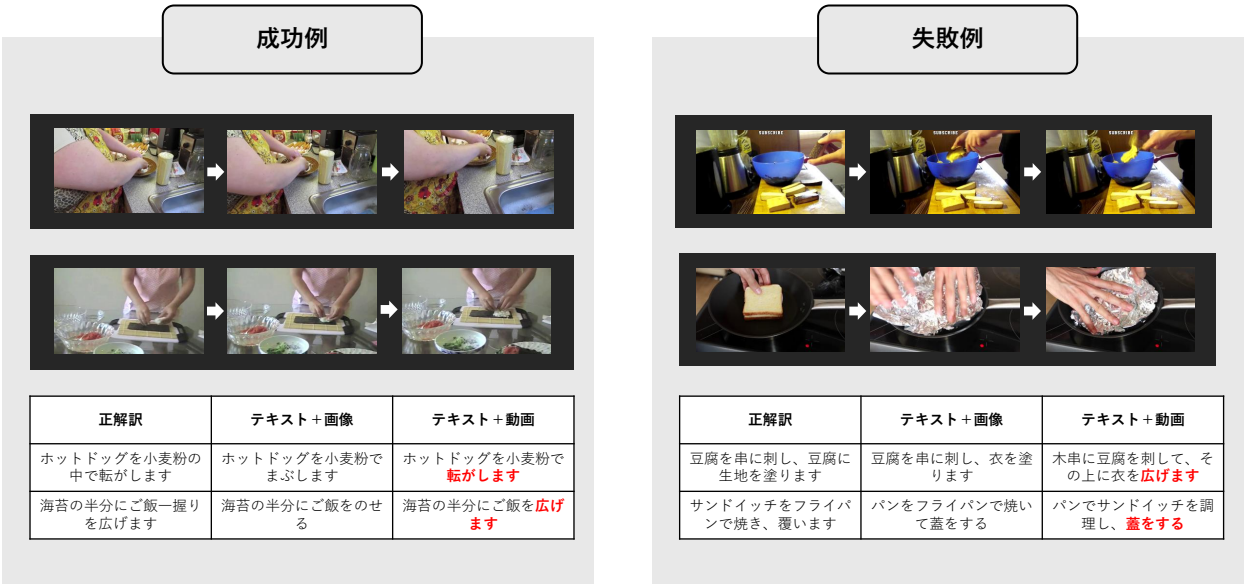

図 2: 0-shot 時の Qwen3-Omni の翻訳例

く反映されているかどうかを評価することが可能となる．「正しく翻訳された動詞」は，参照訳と生成訳からそれぞれ動詞を取り出し，辞書形（基本形）に正規化したうえで，同じ動詞が両方に含まれているかで判定した．一致した動詞の個数を「正しく翻訳された動詞の数」として数えた．

文全体の評価　文全体の翻訳品質を評価するため，BLEU[8] および COMET[5)] [9] を用いた．これらの指標により，動画情報の有無や入力条件の違いが，翻訳文全体の品質にどのように影響を与えるかを確認する．

4.3　実験結果

表 1 に翻訳結果を示す．結果よりテキストのみに比べ，視覚情報を追加することで評価が向上した．0-shot では，両モデルで動画入力が画像入力を一貫して上回った．Qwen3-VL は，動詞に着目した評価において動画入力が最良であり，文全体評価でも動画入力が最も高かった．Qwen3-Omni でも同様に，動詞指標・文全体指標の双方で動画入力が最良となった．動画を用いた翻訳の成功例を図 2 に示す．以上より，0-shot 条件において，静止画よりも時間的情報を含む動画の方が，料理動詞の訳語選択と文全体の意味整合の両面での有効性を確認した．

また，図 2 に示すように，翻訳の大意は保たれているが，対象動詞の訳語が誤る例があり，動画入力によって動作の手掛かりが増え訳語が改善する事例がある一方で，訳のニュアンスはあっていても動詞の訳が別の語に置き換わるなど，動詞レベルでは誤訳が残ることが確認された．

次に few-shot では，全体として性能が向上したことで動画入力と画像入力の差は 0-shot より小さくなる傾向が見られた．それでも，文全体評価では動画入力が優位であり，Qwen3-VL と Qwen3-Omni の両方で動画入力が最良の値を示した．一方，F1 については，Qwen3-Omni では動画入力が最良であったが，Qwen3-VL では画像入力が僅差で上回った．few-shot により動詞訳語が例文から補強されることで，入力の画像と動画でのモダリティ差が相対的に弱まった可能性がある．

5　おわりに

本研究では，料理手順動画翻訳における多義的動詞の訳語選択の難しさに対し，動画情報を用いるマルチモーダル機械翻訳手法を提案した．テキストのみ翻訳の分析から誤訳しやすい 14 動詞を抽出し，YouCook2-JP を用いて動詞に着目したデータセットを構築した．0-shot ／ few-shot を，テキストのみ・テキスト＋画像・テキスト＋動画の 3 条件で比較した結果，動詞評価と文全体評価の双方で動画入力が最も高性能であり，さらに動詞に着目した few-shot により訳語選択が向上した．

5)　https://huggingface.co/Unbabel/wmt22-comet-da

謝辞

本研究は国立研究開発法人情報通信研究機構の委託研究（課題番号：225）および JSPS 科研費 JP24K15071 および大学発新産業創出基金事業スタートアップ・エコシステム共創プログラム JPMJSF2316 の助成を受けたものです.

参考文献

[1] Umut Sulubacak, Ozan Caglayan, Stig-Arne Grönroos, Aku Rouhe, Desmond Elliott, Lucia Specia, and Jörg Tiedemann. Multimodal Machine Translation through Visuals and Speech. *Machine Translation*, pages 97–147, 2020.

[2] 高田 一慶, 二宮 崇, and 後藤 功雄. 画像説明による思考連鎖を用いた料理動画の機械翻訳. **人工知能学会全国大会論文集**, JSAI2025:3N6GS705–3N6GS705, 2025.

[3] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. Chain of Thought Prompting Elicits Reasoning in Large Language Models. *CoRR Journal*, 2022.

[4] Luowei Zhou, Chenliang Xu, and Jason J. Corso. Towards automatic learning of procedures from web instructional videos, 2017.

[5] GPT-4V(ision) System Card. 2023.

[6] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-Vl Technical Report. *arXiv:2511.21631*, 2025.

[7] Jin Xu, Zhifang Guo, Hangrui Hu, Yunfei Chu, Xiong Wang, Jinzheng He, Yuxuan Wang, Xian Shi, Ting He, Xinfa Zhu, Yuanjun Lv, Yongqi Wang, Dake Guo, He Wang, Linhan Ma, Pei Zhang, Xinyu Zhang, Hongkun Hao, Zishan Guo, Baosong Yang, Bin Zhang, Ziyang Ma, Xipin Wei, Shuai Bai, Keqin Chen, Xuejing Liu, Peng Wang, Mingkun Yang, Dayiheng Liu, Xingzhang Ren, Bo Zheng, Rui Men, Fan Zhou, Bowen Yu, Jianxin Yang, Le Yu, Jingren Zhou, and Junyang Lin. Qwen3-Omni Technical Report. *arXiv:2509.17765*, 2025.

[8] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[9] Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. COMET: A Neural Framework for MT Evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

```python
SYSTEM_PROMPT = (
    "You are a professional translator. Translate the given English sentences "
    "into natural Japanese. Output Japanese only, one line per item."
)

def build_messages(en_text: str):
    en_text = en_text or ""
    return [
        {"role": "system", "content": SYSTEM_PROMPT},
        {"role": "user", "content": en_text},
    ]
```

(a) Text (0-shot)

```python
BASE_SYSTEM_PROMPT = (
    "You are a professional Japanese translator for cooking instructions.\n"
    "Translate the given English instruction into natural Japanese.\n"
)

def build_system_prompt(fewshot_block: str) -> str:
    prompt = BASE_SYSTEM_PROMPT

    if fewshot_block:
        prompt += "\nHere are example translations:\n\n" + fewshot_block + "\n\n"
    prompt += (
        "Now translate the following English instruction.\n"
        "Output ONLY the Japanese translation as a single line."
    )
    return prompt

def build_messages(en_text: str, fewshot_block: str):
    en_text = str(en_text or "").strip()
    system_prompt = build_system_prompt(fewshot_block)
    return system_prompt, [
        {"role": "system", "content": system_prompt},
        {"role": "user", "content": en_text},
    ]
```

(b) Text (few-shot)

```python
SYSTEM_PROMPT = (
    "You are a professional Japanese subtitle translator for cooking images.\n"
    "Translate the given English sentence into natural Japanese, "
    "while considering the content of the provided cooking image.\n"
    "Output ONLY the Japanese translation as a single line."
)

def build_messages(en_text: str, img_path: str):
    en_text = en_text or ""
    content = []

    if isinstance(img_path, str) and img_path.strip():
        if os.path.exists(img_path):
            pil_image = Image.open(img_path).convert("RGB")
            content.append({"type": "image", "image": pil_image})

    prompt_text = SYSTEM_PROMPT + "\n\nEnglish instruction:\n" + en_text
    content.append({"type": "text", "text": prompt_text})

    return [
        {"role": "system", "content": SYSTEM_PROMPT},
        {"role": "user", "content": content},
    ]
```

(c) Image (0-shot)

```python
BASE_SYSTEM_PROMPT = (
    "You are a professional Japanese subtitle translator for cooking images.\n"
    "Translate the given English instruction into natural Japanese, "
    "considering the provided cooking image.\n"
)

def build_system_prompt(verb: str, fewshot_block: str) -> str:
    prompt = BASE_SYSTEM_PROMPT
    if fewshot_block:
        prompt += "\nHere are example translations:\n\n" + fewshot_block + "\n\n"
    prompt += (
        "Now translate the following English instruction.\n"
        "Output ONLY the Japanese translation as a single line."
    )
    return prompt

def build_messages(en_text: str, img_path: str, system_prompt: str):
    en_text = en_text or ""
    content = []

    pil_image = None
    if isinstance(img_path, str) and img_path.strip() and os.path.exists(img_path):
        pil_image = Image.open(img_path).convert("RGB")
        content.append({"type": "image", "image": pil_image})

    content.append({"type": "text", "text": "English instruction:\n" + en_text})

    return [
        {"role": "system", "content": system_prompt},
        {"role": "user", "content": content},
    ]
```

(d) Image (few-shot)

```python
SYSTEM_PROMPT = (
    "You are a professional Japanese subtitle translator for cooking videos.\n"
    "Translate the given English instruction into natural Japanese, "
    "while considering the content of the provided cooking video segment.\n"
    "Output ONLY the Japanese translation as a single line."
)

def build_messages(en_text: str, video_path: str):
    en_text = en_text or ""
    video_path = (video_path or "").strip()

    content = []
    if video_path:
        content.append({"type": "video", "video": video_path})

    prompt_text = SYSTEM_PROMPT + "\n\nEnglish instruction:\n" + en_text
    content.append({"type": "text", "text": prompt_text})

    return [{"role": "user", "content": content}]
```

(e) Video (0-shot)

```python
BASE_SYSTEM_PROMPT = (
    "You are a professional Japanese subtitle translator for cooking videos.\n"
    "Translate the given English cooking instruction into natural Japanese.\n"
)

def build_system_prompt(verb: str, fewshot_block: str) -> str:
    prompt = BASE_SYSTEM_PROMPT
    if fewshot_block:
        prompt += "\nHere are example translations:\n\n" + fewshot_block + "\n\n"
    prompt += (
        "Now translate the following English instruction.\n"
        "Output ONLY the Japanese translation as a single line."
    )
    return prompt

def build_messages(en_text: str, video_path: str, system_prompt: str):
    en_text = en_text or ""
    video_path = (video_path or "").strip()
    use_video = bool(video_path and os.path.exists(video_path))

    content = []
    if use_video:
        content.append({"type": "video", "video": video_path})

    prompt_text = system_prompt + "\n\nEnglish instruction:\n" + en_text
    content.append({"type": "text", "text": prompt_text})

    return [
        {"role": "system", "content": system_prompt},
        {"role": "user", "content": content},
    ]
```

(f) Video (few-shot)

図 3: 入力条件ごとのプロンプト例

付録 (Appendix)

入力条件ごとのプロンプト設計

図 3 に示すように入力条件は，テキストのみの入力，テキスト＋画像の入力，テキスト＋動画の入力ごとで，モデルに与える情報を切り替えた．全条件で翻訳タスクの指示と出力制約は共通とし，0-shot は例を与えず，few-shot は英日ペアの少数例を同一プロンプト内に追加して誤訳選択と文体を誘導した．条件間の差分は，0-shot と few-shot の例示の有無と追加モダリティの有無に限定した．