Human Reconstruction MSE Calculations

November 30, 2025

[1]:
```
import os
import numpy as np
import pandas as pd
from collections import defaultdict

from wcs_helper_functions import (
    readChipData,
    readClabData,
    readNamingData,
    readFociData,
    readSpeakerData,
    plotValues
)

base_dir = 'WCS_data_core'
chip_path = os.path.join(base_dir, 'chip.txt')
clab_path = os.path.join(base_dir, 'cnum-vhcm-lab-new.txt')
term_path = os.path.join(base_dir, 'term.txt')
foci_path = os.path.join(base_dir, 'foci-exp.txt')
spkr_path = os.path.join(base_dir, 'spkr-lsas.txt')
```

[2]:
```
cielab_raw = readClabData(clab_path)
naming_data = readNamingData(term_path)

def to_float_triplet(triplet):
    return np.array([float(triplet[0]) / 50.0 - 1.0, float(triplet[1]) / 128.0,␣
 ↪float(triplet[2]) / 128.0], dtype=np.float64)

cielab_by_chip = {int(k): to_float_triplet(v) for k, v in cielab_raw.items()}
```

[3]:
```
def compute_label_centroids(cielab_by_chip, speaker_labels):
    sums = defaultdict(lambda: np.zeros(3, dtype=np.float64))
    counts = defaultdict(int)
    for chip_id, label in speaker_labels.items():
        if chip_id in cielab_by_chip:
            sums[label] += cielab_by_chip[chip_id]
            counts[label] += 1
```

```
    return {label: total / counts[label] for label, total in sums.items() if␣
 ↪counts[label] > 0}

def reconstruct_chip(cielab_by_chip, label_centroids, label):
    return label_centroids.get(label, None)

def mse_for_speaker(cielab_by_chip, speaker_labels):
    label_centroids = compute_label_centroids(cielab_by_chip, speaker_labels)
    diffs_sq = []
    for chip_id, label in speaker_labels.items():
        x_true = cielab_by_chip.get(chip_id)
        x_hat = reconstruct_chip(cielab_by_chip, label_centroids, label)
        if x_true is not None and x_hat is not None:
            diffs_sq.append(np.sum((x_true - x_hat) ** 2))
    return np.mean(diffs_sq) if diffs_sq else np.nan
```

[4]:
```
per_speaker = []
language_summary = []

for lang_id, speakers in naming_data.items():
    speaker_mses = []
    for spk_id, chip_to_label in speakers.items():
        cleaned = {int(ch): str(lbl) for ch, lbl in chip_to_label.items() if␣
 ↪int(ch) in cielab_by_chip}
        k_terms = len(set(cleaned.values()))
        n_chips = len(cleaned)
        mse = mse_for_speaker(cielab_by_chip, cleaned)
        per_speaker.append([lang_id, spk_id, k_terms, n_chips, mse])
        speaker_mses.append((spk_id, mse))

    valid_mses = [m for (_, m) in speaker_mses if not np.isnan(m)]
    if valid_mses:
        best_spk, best_mse = min(speaker_mses, key=lambda x: x[1])
        median_mse = np.median(valid_mses)
    else:
        best_spk, best_mse, median_mse = None, np.nan, np.nan
    language_summary.append([lang_id, best_spk, best_mse, median_mse,␣
 ↪len(speaker_mses)])
```

[5]:
```
os.makedirs(os.path.join(base_dir, 'outputs'), exist_ok=True)

df_speaker = pd.DataFrame(per_speaker, columns=['language_id', 'speaker_id',␣
 ↪'k_terms', 'n_labeled_chips', 'mse_cielab'])
df_lang = pd.DataFrame(language_summary, columns=['language_id',␣
 ↪'best_speaker_id', 'best_mse_cielab', 'median_mse_cielab', 'n_speakers'])
```

```
df_speaker.to_csv(os.path.join(base_dir, 'outputs', 'wcs_per_speaker_mse.csv'),␣
 ↪index=False)
df_lang.to_csv(os.path.join(base_dir, 'outputs', 'wcs_language_best_mse.csv'),␣
 ↪index=False)

df_lang.head()
```

[5]:

	language_id	best_speaker_id	best_mse_cielab	median_mse_cielab	\
0	1	12	0.148637	0.185464	
1	2	14	0.130442	0.164561	
2	3	15	0.100081	0.150797	
3	4	3	0.119989	0.158096	
4	5	3	0.138955	0.153671	

	n_speakers
0	25
1	24
2	25
3	35
4	6