Writing in the Air: Turning Quest Hand Tracking into Text

Xavier Jara

20 November 2026

PDF · github.com/TR-MZ/airwrite-xr · huggingface.co/trmz/airwrite-xr · tr-mz.com

Abstract

Typing in VR is still awkward: you poke at a floating keyboard or speak out loud. I wanted to write instead, by tracing words in the air with my finger, with nothing in my hand and no pinch to mark the pen going down (my tests showed that pinching to write doesn't feel natural). AirWrite XR takes the fingertip path from Meta Quest hand tracking and turns it into English sentences as you write.

The recognizer is six small motion models (three Conformers and three BiLSTMs, 2.85M parameters in total) trained with CTC, followed by an English decoder. It also works with only three models, but the error rate goes up by about 50% (10.7% against 7.5% CER). I pre-trained them on synthetic 3D handwriting, then fine-tuned them on 170 recordings of my own writing. On 36 held-out recordings in four folds, the released model, **v13**, reaches:

- **7.5%** character error rate (CER) and **16.4%** word error rate, with **19 of 36** sentences exactly right;
- against **10.7%** CER for my first release, v9.

Flattening the same movements into images and reading them with strong handwriting OCR models fails: on a held-out sentence that v13 reads perfectly, TrOCR-large and Qwen3-VL get 2 and 0 words out of 5 right. A lighter version of the six models runs entirely in the headset's browser, in about 15 MB. All the real data comes from one writer, so these numbers say nothing yet about other people.

1. Problem

Handwriting recognition usually starts from ink: a pen touches a surface, so you know where each stroke begins and ends, and the letters sit on a flat page. In the air, none of that is true.

- **There is no pen-up.** The finger never leaves the page, because there is no page. The movement between two letters, or two words, is recorded exactly like the letters themselves.
- **The page moves.** The writer drifts in depth and height, changes size mid-word, and turns their head.
- **Words pile up.** Without a visible line, you naturally write the next word over the previous one.
- **Spaces must be guessed.** The only sign of a space is a slightly longer jump between two letters.



Figure 1. "they watched the entire movie", written in the air and seen from the writer's point of view. Colour goes from dark (start) to yellow (end). Every word is written over the previous one, and the travel between letters is part of the line. As a picture, it is unreadable.

Real data is also scarce. Nobody has a large dataset of people writing sentences in the air with headset hand tracking, so I had to record my own, which is slow and boring.

My goals were:

1. read whole sentences of lowercase English, not single letters or a fixed list of words;
2. stream the result while writing, word by word;
3. need no pen-down gesture, controller or keyboard;
4. get close to 5% CER, the point where correcting the odd letter is faster than typing.

2. The models in this paper

I trained many versions. These are the two releases, with the names used from here on.

Name	Motion models	Params	Real training data	Extra training
v9	3 Conformers	1.65M	27 test-fold takes + ~93 new takes	–
v13	3 Conformers + 3 BiLSTMs	2.85M	136 lowercase takes	capitals and slow writing [1]

[1] Four of the six v13 models were retrained with capital-letter writing and a time-warp that slows writing down by up to 2.2×. The other two are kept from lowercase-only training (§6.3).

The six v13 models:

Member	Architecture	Params	Synthetic pre-training
primary	2-block Conformer, rotary positions	548k	single-style corpus
secondary	2-block Conformer, rotary positions	548k	13 handwriting styles
tertiary	2-block Conformer + self-conditioned CTC [2]	552k	single-style corpus
bilstm	BiLSTM	400k	many styles
bilstm_old	BiLSTM, lowercase only	400k	many styles
bilstm_s13	BiLSTM	400k	13 handwriting styles

[2] Self-conditioned CTC (Nozaki & Komatsu, 2021): the model's own intermediate letter guesses are fed back into its later layers.

3. Capture

I built a WebXR app for the Quest 3 that records hand tracking while I write a sentence shown in front of me, and a viewer to replay each take in 3D and check that nothing was missed. Each frame stores 25 hand joints (75 numbers), a timestamp and the head pose.

The recognizer only looks at two points on the index finger: the fingertip and the base of the finger (the knuckle). For each take I:

1. move the fingertip into the head's frame at the start of the take, so it doesn't matter where I stand or look;
2. resample to 60 Hz and smooth out tracking jitter;
3. scale by the writing height, so big and small writing look the same.

That gives 12 numbers per frame: fingertip position (3), velocity (3), fast detail left after smoothing (3), and the fingertip relative to the knuckle (3), which shows how the finger is bent, so a finger that curls to draw a small loop doesn't look the same as a whole hand moving. No camera image, video or audio is used.

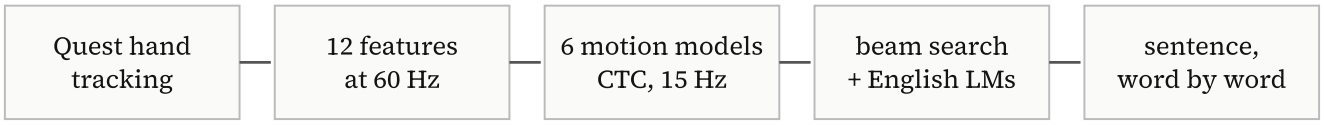

Figure 2. From fingertip to text.

4. Data

4.1 Real recordings

I recorded 170 takes of myself writing, roughly 40 minutes of writing for the first 136. The sentences come from the MacKenzie–Soukoreff phrase set, which is standard for testing text entry. Most takes are in lowercase cursive; a few are in capitals or were used to test calibration.

36 of the lowercase takes are the evaluation set, split into four fixed folds of nine (§5).

4.2 Synthetic handwriting in 3D

170 recordings are far too few to train on alone. So I generated handwriting: a PyTorch port of the Graves handwriting RNN (Graves, 2013) writes a sentence as 2D pen strokes, which I lift into a 3D fingertip path. The lifting is calibrated on real Quest hand poses and adds the things that make air-writing hard: slant, a baseline that drifts, uneven tempo, depth wobble, and the finger's travel between letters and words, which is kept instead of removed.

I made several corpora: 10k and 100k sentences in one style, a 13-style version, and a many-style one. Different members are pre-trained on different corpora, which matters when they are pooled (§6.2).

5. Evaluation

All lowercase scores in this paper use the same test: 36 real takes, 1,023 characters, in four frozen folds of nine. For each fold, the models train on everything except that fold's nine takes, then decode those nine. No checkpoint is picked using the test takes.

- **CER**: character edit distance divided by the number of reference characters.
- **WER**: the same over words.
- **Exact**: sentences decoded with no error at all.

Important note: everything is measured only on my own writing.

6. Recognition

6.1 Motion models with CTC

Each motion model reads the 12 features and outputs, 15 times a second (one step every 4 frames), a probability for each of 28 symbols: the 26 letters, space, and a "blank" for "nothing new here". They are trained with CTC (connectionist temporal classification), which only needs the sentence, not where each letter starts and ends. That is essential here, because nobody can say exactly where an air-written letter starts.

Every model is pre-trained on synthetic data, then fine-tuned on real takes with synthetic examples mixed back in, so it doesn't forget.

On its own, read greedily (the most likely symbol at each step, no language help), each model gets about 28% CER; for example, the self-conditioned Conformer alone gets **27.8% CER**. On the sentence in Figure 1, the primary model alone reads "they watchad the ntire mavrre". The rest of the gain comes from pooling models and from the decoder.

6.2 Pooling at the sentence level

The obvious way to combine models is to average their probabilities frame by frame. It works badly: the models don't agree on timing, so averaging smears their peaks. In my browser decoder, frame averaging gave 20% CER and no exact sentence.

What works is to let each model propose sentences, then score every candidate sentence under every model and add the scores. Each model keeps its own timing, and they only have to agree on the text. The same browser decoder went from 20% to 14.2% CER with this change alone.

Diversity matters more than count. Members pre-trained on different synthetic styles make different mistakes, so pooling them cancels more errors than pooling copies of the same model.

6.3 Capitals and slow writing

When I wrote in capitals, or slowly, the lowercase models fell apart: about 38% CER on the capitals test. People in VR often write that way. I tested two fixes on the same folds:

Setup	Capitals CER	Lowercase errors / 1,023
Lowercase-only members	38%	74
4 members, all retrained with capitals + slow writing	6.1%	87
6 members: the 4 retrained + 2 lowercase-only BiLSTMs (v13)	**4.2%** [3]	**77**

[3] Capitals CER for the final production models. Lowercase errors are from the fold models.

Retraining everything fixed capitals but cost 13 lowercase errors. Keeping two lowercase-only members in the pool brought almost all of that back (3 errors out of 1,023, within noise) and still read capitals best. That is v13.

7. Decoding

The decoder turns the six models' letter probabilities into a sentence:

1. **Beam search.** A CTC prefix beam search (width 64) builds candidate sentences letter by letter, guided by a character 5-gram and a lexicon trie, so it prefers real words but can still spell new ones.
2. **Rescoring.** A small character Transformer and SmolLM2-135M rescore the candidates.
3. **Widening.** Two rounds add close dictionary words (a letter or two away) for words the search got almost right.
4. **Reranking.** SmolLM2-1.7B picks the best of the top 8 sentences.

This runs as you write. A draft updates about 8 times a second. A word is locked about 200 ms after you move on to the next one, and the sentence so far is re-decoded each time, so earlier words can still be fixed with new context.

The language models matter, but they can't make up for weak motion evidence. If the right sentence never makes it into the beam, no reranker can pick it.

Adapting to where it's used. Sometimes you know what people will write, and it pays to adapt. In my VR workspace, VibeAR.dev, people write short commands like "open terminal" rather than prose, so I also adapted the text models to about 12k short commands. On 120 synthetic test trajectories, adapting both motion and language models took CER from 14.1% to 3.8%. These are synthetic, so they show the size of the domain effect, not real accuracy. All other numbers in this paper use the general-English decoder.

8. Results

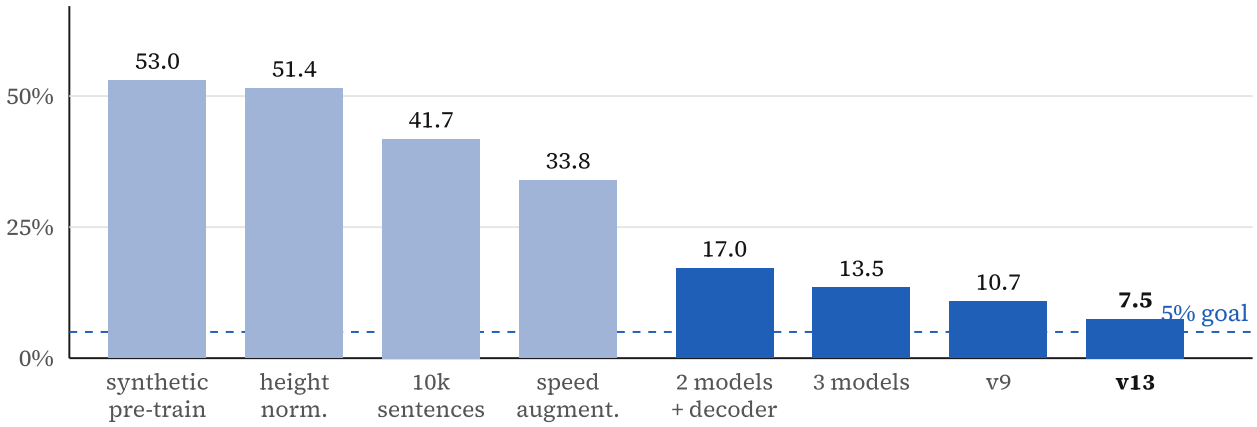

Figure 3. CER through the project. Light bars: a single model, greedy, averaged over folds during training. Dark bars: the full system on the 36 test takes.

The first model, a temporal CNN on raw joints, was at 77% CER. The light bars in Figure 3 are single models without a language model: synthetic pre-training, height normalisation, more varied sentences and speed augmentation brought them to about 34%. The dark bars are the full system with pooling and the decoder.

Model	CER	WER	Exact
v9 (3 models)	10.7%	22.1%	–
v13 (6 models)	**7.5%**	**16.4%**	**19 / 36**

v13's CER per fold is 5.4%, 6.0%, 4.8% and 13.9%. Three folds are already near the 5% goal. The fourth holds the takes where I wrote the worst: overlapping words, and letters that collapse into a single stroke.

8.1 More real data helps, with diminishing returns

v9 came from one experiment: same models, same decoder, same test, but with new real takes added to every fold's training set.

New real takes added	Errors / 1,023	CER
0	138	13.5%
12	120	11.7%
24	117	11.4%
~93 (v9)	109	10.7%

Real data was the only thing that moved the score in that experiment: training longer and averaging seeds gave nothing. But the gains shrink quickly. A rough power-law fit says one writer would need thousands of takes to reach 5% this way. More writers, not more of my own writing, is the next step.

9. Why not just flatten it and use OCR?

If the fingertip path were a picture, I could use an existing handwriting recognizer, trained on far more data than I will ever record. So I tried it.

The pilot. I projected the path onto the writing plane, cut it into word images and read them with TrOCR. On four older takes, cutting by time order gave **65.98% CER**; cutting by pauses in the movement gave **60.82% CER**. WER was **94.44%** either way: almost no word came out right. (These four takes were looked at during development, so they are not a clean test, but the gap is too large to matter.)

A fair comparison on one sentence. To show why, I took the sentence from Figure 1, which is in the test set, and gave the OCR models every advantage: I cut it into words using AirWrite's own timing, so each image contains exactly one word, and told the model it is lowercase cursive English. I used two strong models: TrOCR-large-handwritten, a dedicated handwriting recognizer, and Qwen3-VL-4B, a general vision-language model.



TrOCR-large: they	TrOCR-large: malod	TrOCR-large: the	TrOCR-large: nbre	TrOCR-large: more
Qwen3-VL-4B: chey	Qwen3-VL-4B: mukdol	Qwen3-VL-4B: who	Qwen3-VL-4B: wire	Qwen3-VL-4B: marre
AirWrite v13: they	**AirWrite v13: watched**	**AirWrite v13: the**	**AirWrite v13: entire**	**AirWrite v13: movie**

Figure 4. The five words of Figure 1, each flattened to an image. Green is correct, red is wrong. TrOCR-large gets 2 of 5 words, Qwen3-VL-4B gets 0 of 5. AirWrite v13, using a fold model that never saw this take, reads the whole sentence correctly. Given the whole sentence as one image, TrOCR-large returned "0" and Qwen3-VL-4B did not attempt a transcription.

"watched" shows the problem best: the "a", "t", "c" and "h" are written over each other, and the up-and-down travel between them looks like extra letters. In the image, that information is gone. In the movement, it is still there: the order, the speed, and the small pauses at each letter. That is what the motion models read.

9.1 Other things that didn't work

- **Other output units and decoders.** Subword CTC (59.8% vs 34.1% for letters), an RNN-T (38.7% vs about 35%) and a free-running Transformer decoder (87.4%) all lost to plain letter CTC on the same fold.
- **More synthetic data alone.** Going from 10k to 100k synthetic sentences changed nothing (33.8% vs 34.0%). Variety of style mattered; volume didn't.
- **A BiGRU member.** It was worse than the BiLSTM, so I dropped it.
- **Frame-level pooling** (§6.2).

One small win: rotary position encoding beat absolute positions in the Conformer (32.6% vs 33.4%), and all Conformers use it.

10. Running in the headset

The full decoder needs a GPU because of the 1.7B reranker. To run with no computer at all, I made a lighter version that runs entirely in the Quest browser: the same six

motion models with small language models, about 15 MB in total. It is less accurate than the full decoder. More details will be on GitHub.

11. Limitations

- **One writer.** All the real training data comes from me. Expect a clear drop for other people.
- **Lowercase cursive (and sometimes print and capitals) only.** Capitals were tested on only six recordings, and print is still very much research only.
- **English only**, with no digits or punctuation.
- **Decoder cost.** The full decoder needs a GPU because of the 1.7B reranker. The motion models alone run fine on CPU, and a lighter version (character 5-gram and lexicon only) runs entirely in the headset browser, with lower accuracy.
- **Development set.** The 36 test takes guided many decisions, so 7.5% is optimistic.

12. Conclusion

Writing in the air with bare-hand tracking can be read at 7.5% CER for one writer, with six small motion models, synthetic 3D handwriting and an English decoder, and it can run entirely in the headset. Flattening the movement into an image throws away exactly what makes it readable, which is why strong OCR models fail on it. Three things mattered most: synthetic pre-training that keeps the travel between letters, pooling diverse models at the sentence level rather than frame by frame, and real data, which was the only lever that kept helping. The 5% goal is close on three folds out of four. Reaching it reliably, and for other people, needs recordings from many writers. That is the next step.

References

- Graves, A. (2013). *Generating Sequences with Recurrent Neural Networks.* arXiv:1308.0850. PyTorch port: sjvasquez/handwriting-synthesis.
- Graves, A., Fernández, S., Gomez, F., Schmidhuber, J. (2006). *Connectionist Temporal Classification.* ICML.

- Gulati, A. et al. (2020). *Conformer: Convolution-augmented Transformer for Speech Recognition.* Interspeech.
- Nozaki, J., Komatsu, T. (2021). *Relaxing the Conditional Independence Assumption of CTC-based ASR by Conditioning on Intermediate Predictions.* arXiv:2104.02724.
- Li, M. et al. (2023). *TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models.* AAAI.
- MacKenzie, I. S., Soukoreff, R. W. (2003). *Phrase Sets for Evaluating Text Entry Techniques.* CHI.
- Allal, L. B. et al. (2025). *SmolLM2.* Hugging Face.