



STAC for Embeddings

Combining metadata search with semantic search

STAC answers what scenes exist.

The next question is what's in them.

The gap

Ask STAC for **solar farms in California, 2023**:

- you get every scene over California in 2023
- STAC has no idea which ones have a solar farm in them
- embeddings can rank them by what's in them, once STAC cuts the list down

So: STAC first, then semantic. But once embeddings are in STAC, the question I care about more is **can you tell where an embedding came from?**

Where this goes

01 **Format**

Pixels or chips? That decides it.

02 **Provenance**

Can you get back to the imagery? For AlphaEarth, no.

03 **Two STAC patterns**

An asset on the scene's Item, or its own Item.

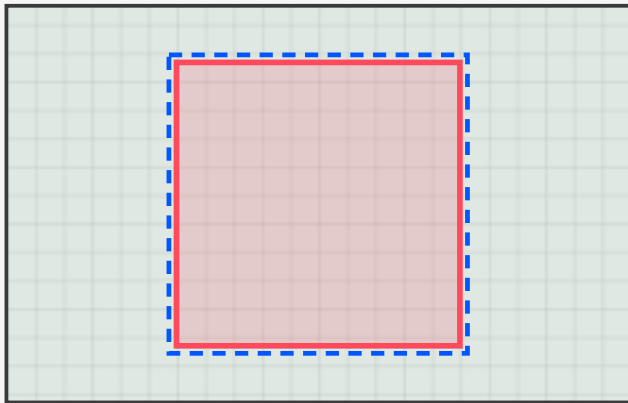
04 **Search**

STAC first, then vectors.

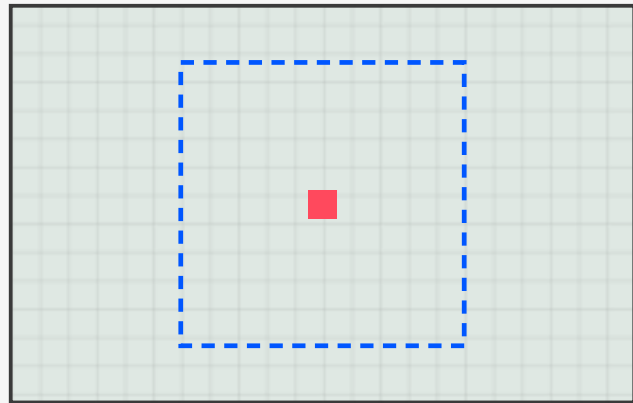
01 What format?

GeoParquet, COG, or Zarr

Pixel or chip?



chip embedding



pixel embedding

red: the area one vector describes · dashed blue: context geometry, what the model saw to make it

A **chip** embedding (Clay, Earth Index) is one vector per chip: what it describes is what the model saw.

A **pixel** embedding (AlphaEarth) is one vector per 10 m pixel, but the model can see well beyond that pixel. That bigger area is its **context geometry**.

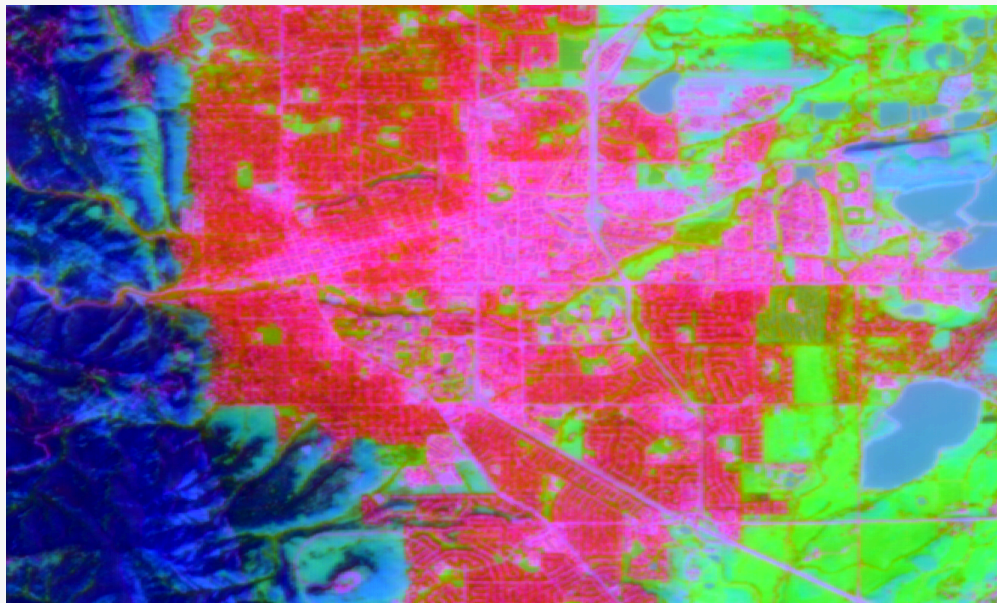
What's shipping today

All on Source Cooperative.

Product	Embedding	Format
AlphaEarth (<code>tge-labs/aef</code>)	per pixel · 64-d int8 · annual	COG , one set per UTM zone
AlphaEarth (<code>tge-labs/aef-mosaic</code>)	the same, reprojected to EPSG:4326	Zarr v3, one global store
TESSERA (<code>tessera/tessera</code>)	per pixel · 128-d · annual	Zarr (and <code>.npy</code>)
LGND Clay v1.5 (<code>clay/lgnd-embeddings</code>)	per chip · Major TOM cells · monthly	GeoParquet
Earth Index (<code>earthgenome/earthindexembeddings</code>)	per chip · 320 m · 384-d · annual mosaic	GeoParquet , one file per MGRS tile-year

THE LINE FALLS BETWEEN PIXEL AND CHIP PRODUCTS, NOT BETWEEN SENSORS.

A pixel embedding is a 64-band raster



Boulder, Colorado, 2024. AlphaEarth embeddings, not imagery: 64 dimensions reduced to 3 with PCA (56% of the variance) and shown as RGB. Read from [tge-labs/aef-mosaic](https://tge-labs.github.io/aef-mosaic). The AlphaEarth Foundations Satellite Embedding dataset is produced by Google and Google DeepMind.

GeoParquet or COG / Zarr?

	GeoParquet	COG / Zarr
One embedding is	a row, with its own geometry	a pixel in a band stack
Grid	any grid, overlaps are fine	one regular grid per file, so AlphaEarth is split by UTM zone
Item id, cell id, time	columns you can filter and join in SQL	nowhere to put them
Size	float32 doesn't compress	int8 compresses ~3x: an AlphaEarth COG is 4.3 GB raw, 1.4 GB on disk
Dense pixels	120 M rows per S2 scene, ~8.7 GB	what it's built for
Search	top-k in DuckDB, no index	similarity and change maps over an area; top-k across files is awkward

My take

- Pixels on a regular grid: COG or Zarr.
- Chips, or a grid like Major TOM: GeoParquet.
- Image-aligned chips *could* be a raster. I don't think it's worth losing the per-row Item id.

Either way it's a STAC asset, so the embedding fields belong **on the asset**.

02 Provenance

An embedding is a **summary.**

A summary should point to its sources.

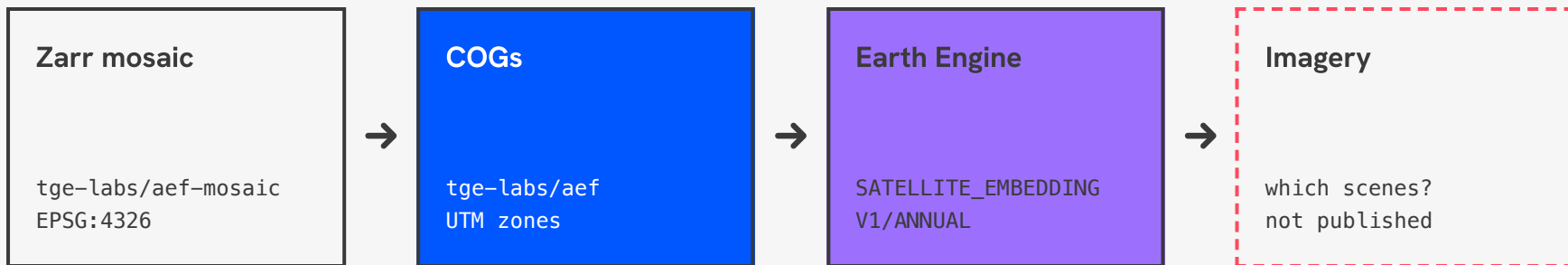
The file can describe itself. It can't link.

Put the model, dimensions and quantization in the file: Zarr attributes, the Parquet footer. Do this.

What you can't do from inside a file:

- link to the source Items (`derived_from`), and through them the pixels the model actually saw
- link to the model and its version
- say what software made it, and when (`processing:*`)
- carry a licence per product
- get **searched**. "Every embedding made from this scene" is a catalog query, not a file read.

Follow AlphaEarth back



The Zarr mosaic says its source is the COG copy, which is also embeddings. The COG README points at Earth Engine. And that's where it stops: which scenes went in isn't published.

AlphaEarth can't show its work

Would I explore with it?

Yes.

Would I use it as evidence?

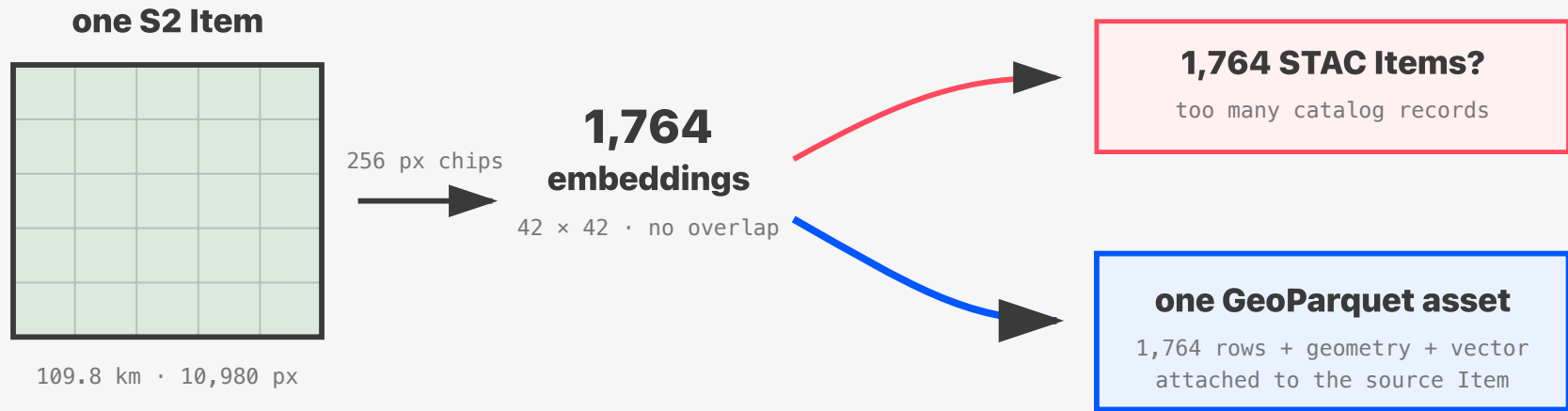
No.

They're really useful. But as evidence, it's a bit of a "trust me bro" situation. The model is closed, but the real problem is the inputs: they aren't published, so you can't check what it saw.

UNPUBLISHED INPUTS • CLOSED MODEL • CC-BY OUTPUTS

03 Two STAC patterns

One scene is not 1,764 Items

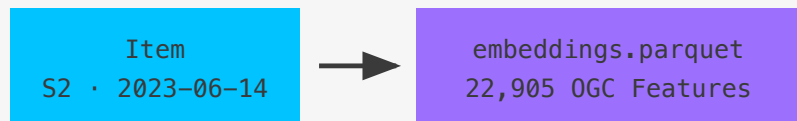


A Sentinel-2 tile is 10,980 pixels across. Cut it into 256-pixel chips and you get $42 \times 42 = 1,764$ embeddings. Don't make those Items. They're rows in one file, and that file is an asset.

Two products, two STAC patterns

One scene in, chips out

optical, per acquisition



an asset on the Item it came from – the provenance is the parent

Many scenes in, one surface out

per-pixel · annual · time series



Made from one scene? Put it on that scene's Item.

Made from many, like a year of Sentinel-2 or S1 + S2 together? Give it its own Item, with a `derived_from` link to every source. Do that too if the source Items aren't yours to edit.

A mosaic is the same thing with bigger files.

Pattern 1: an asset on the S2 Item

```
{
  "type": "Feature",
  "stac_version": "1.0.0",
  "id": "S2B_53SKU_20240614_0_L2A",
  "collection": "sentinel-2-l2a",
  "properties": {
    "datetime": "2024-06-14T02:07:51.098Z",
    "platform": "sentinel-2b",
    "eo:cloud_cover": 7.65,
    ...
  },
  "assets": {
    "red": { "href": ".../B04.tif", ... },
    "nir": { "href": ".../B08.tif", ... },
    "visual": { "href": ".../TCI.tif", ... },
    ...
  }
}
```

```
    "embeddings-clay": {
      "href": ".../S2B_53SKU_20240614_0_L2A.parquet",
      "type": "application/vnd.apache.parquet",
      "roles": ["data", "embeddings"],
      "emb:type": "chip",
      "emb:dimensions": 256,
      "data_type": "float32",
      "emb:model": "clay-v1.5",
      "emb:normalization": "none",
      "emb:distance_metric": "cosine"
    }
  },
  "links": [
    { "rel": "derived_from",
      "href": ".../sentinel-2-l1c/items/S2B_53SKU_20240614_0_L1C" },
    ...
  ]
}
```

A REAL EARTH SEARCH ITEM, TRIMMED, PLUS ONE ASSET. IT ALREADY HAS A DERIVED_FROM LINK TO ITS L1C ITEM.

Pattern 2: its own Item, linked back

```
{
  "type": "Feature",
  "id": "53SKU-2024-clay-composite",
  "properties": {
    "datetime": null,
    "start_datetime": "2024-03-16T00:00:00Z",
    "end_datetime": "2024-09-07T23:59:59Z",
    "emb:type": "chip",
    "emb:dimensions": 256
  },
  "assets": {
    "embeddings": {
      "href": ".../53SKU-2024.parquet",
      "type": "application/vnd.apache.parquet",
      "emb:model": "clay-v1.5",
      "emb:distance_metric": "cosine"
    }
  },
}
```

```
"links": [
  { "rel": "derived_from",
    "href": ".../items/S2B_53SKU_20240316_0_L2A" },
  { "rel": "derived_from",
    "href": ".../items/S2B_53SKU_20240614_0_L2A" },
  { "rel": "derived_from",
    "href": ".../items/S2A_53SKU_20240907_0_L2A" },
  { "rel": "emb:model",
    "href": "https://huggingface.co/made-with-clay/Clay" },
  { "rel": "collection",
    "href": "./collection.json" }
]
```

The three source Items are real Earth Search Items. The embedding Item is made up.

ASSET-LEVEL EMB:* IS MY PROPOSAL, NOT THE DRAFT · FULL FILES IN EXAMPLES/

One Item can carry more than one embedding

STAC Item · one Sentinel-2 scene

```
|— visual          image/tiff
|— embeddings-clay application/vnd.apache.parquet
|   |— emb:dimensions 1024
|   |— emb:model      clay-v1.5
|— embeddings-terramind application/vnd.apache.parquet
|   |— emb:dimensions 768
|   |— emb:model      terramind-v1
```

Put `emb:dimensions` on the Collection and it's wrong for one of these. On each asset, both are right.

There's already a draft extension

From the first Geo-Embeddings Sprint, in March: `geo-embeddings/embeddings-stac-specification` , v0.0.1.

Embeddings as an asset on the source Item

Needs asset scope.

A Collection of derived embedding Items

Works today.

The asset schema rejects `emb:*` . Allow it there and both patterns work, without breaking anything.

Proposed change

Allow the embedding fields on assets.

Put on the Item or Collection	Put on the embedding asset
spatial and temporal extent	dimensions and data type
intended use	model identity and version
product-level provenance	normalization and distance metric
searchable product characteristics	chip layout and vector column
links to source Items	preprocessing and quantization

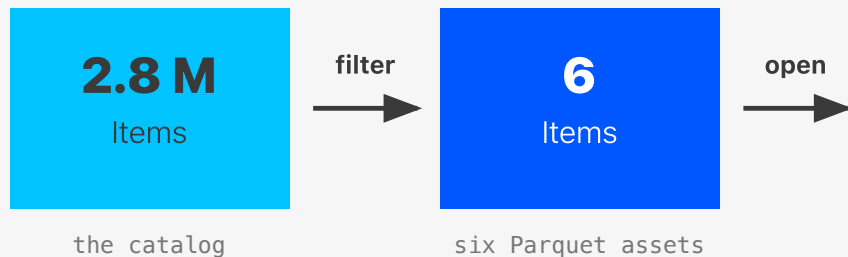
Some fields belong in both places. If the asset says something more specific, use that.

04 **Search the embeddings**

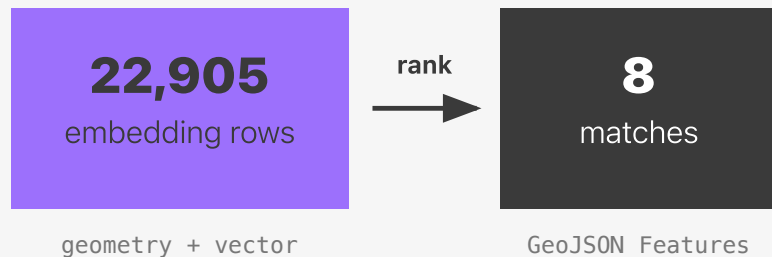
What works today, and what I'd like to work

Start with the files. Add an index when you need one.

STAC SEARCH



VECTOR SEARCH



STAC search finds the Items. Their embedding assets point at files, and DuckDB searches just those. That works for both patterns.

Running queries all day, or global ones with nothing for STAC to narrow? Then build an index, from the files.

Today: STAC search, then DuckDB

```
items = client.search(
    collections=["sentinel-2-l2a"], bbox=aoi, datetime="2024-06-01/2024-09-30",
    query={"eo:cloud_cover": {"lt": 10}},
).items()
hrefs = [i.assets["embeddings"].href for i in items]

con.execute("""
    SELECT stac_item_id, geometry,
           list_cosine_similarity(embedding, $q) AS sim
    FROM read_parquet($hrefs)
    ORDER BY sim DESC LIMIT 20
""", {"q": query_vector, "hrefs": hrefs}).df()
```

That's all of it. No vector database, no index. STAC picks the files and DuckDB reads them.

RUN ON 12 CLONED HIROSHIMA ITEMS, AUG 2026: 6 ITEMS → 22,905 CHIPS → TOP 8 IN ~1 S ON A LAPTOP

What I'd add to STAC API search

POST /search

```
{
  "collections": ["sentinel-2-l2a"],
  "bbox": [-119.0, 35.0, -117.5, 35.6],
  "datetime": "2023-01-01/2023-12-31",
  "filter": { "op": "<", "args": [{"property": "eo:cloud_cover"}, 10] },
  "embedding": {
    "asset": "embeddings-clay",
    "vector": [0.013, -0.225, 0.084, 0.019],
    "metric": "cosine",
    "limit": 20
  }
}
```

Everything outside `embedding` is a normal STAC search. `asset` says which embeddings to use, since one Item can carry more than one, and the vector has to match its dimensions.

The response is a FeatureCollection

```
{
  "type": "FeatureCollection",
  "features": [{
    "type": "Feature",
    "id": "chip-903",
    "geometry": { "type": "Polygon", "coordinates": ["..."] },
    "properties": {
      "score": 0.87,
      "item": "S2B_11SMU_20230818",
      "asset": "embeddings-clay"
    },
    "links": [{ "rel": "item", "href": ".../items/S2B_11SMU_20230818" }]
  }]
}
```

Each match is a Feature with a link to its Item, so you can always get back to the imagery.

The server is your choice

The API is a request and a response. What runs behind it is up to you:

- pgSTAC + eoAPI for the Items, and a vector index for the rows
- DuckDB over the selected Parquet files, no index at all
- a vector database, if you already run one

Where to argue about it

GEO-EMBEDDINGS SPRINT II

Zürich · Oct 27–28

IBM · CNG · Clark University · Planet — three weeks from now

If you're going to the sprint, that's the place to push on any of this. Applications closed in August, so if you're not going:

- `geo-embeddings/embeddings-stac-specification` — issues and PRs
- the monthly Geo-Embeddings calls and the mailing list

Thank you

Questions?

Open questions

- `derived_from` says which Items went into an Item. It doesn't say which source went into which row. For S1 + S2 with different footprints, I don't know what the convention should be.
- Chips across sensors: same pixel count, or same ground size?
- When the model changes, is that a new asset, a new Item, or a new Collection?

Isn't an index faster?

Yes. It also costs more.

	Files + DuckDB	Vector index (S3 Vectors)
Storage	\$0.023 / GB-month	\$0.06 / GB-month, about 2.6×
Queries	the bytes you read	+ \$0.0025 per 1,000 queries
Between queries	nothing running	a service, or a managed one
Build	none	HNSW at 200k × 1,024: 62-90 s, pays off after ~400-800 queries
Results	exact	approximate; filtered queries can return fewer than k

IF YOU RUN THE SAME KIND OF QUERY ALL DAY, BUILD THE INDEX. FOR AN OPEN ARCHIVE WITH THE OCCASIONAL USER, THE FILES ARE CHEAPER.

AWS us-west-2 list prices (S3 Standard, S3 Vectors) · HNSW: DuckDB vss, measured Aug 2026

How big does this get?

	Vectors	Size
LGND Clay v1.5 · S2 chips · 2017–2026	15.2 billion	9.6 TB
Major TOM's copy of AlphaEarth	71 billion	5.5 TiB
AlphaEarth · 10 m pixels · per year, land	~1.5 trillion	
AlphaEarth · 2017–2025	~13 trillion	~0.5 PB
Everything open on Source Coop		2.59 PB

Index limits: S3 Vectors tops out at 2 billion per index. DuckDB HNSW wants all of it in RAM. pgvector + PostGIS is comfortable to about 50 million.

IT'S BEEN DONE: DESCARTES SEARCHED 2 BILLION TILES IN ~0.1 S IN 2019. IT'S EXPENSIVE, NOT IMPOSSIBLE. YOU PAY TO KEEP EVERY VECTOR IN THE INDEX WHETHER ANYONE QUERIES IT OR NOT.

Isn't opening hundreds of files slow?

Every remote file costs about **0.13 s** before any data shows up: a HEAD, a redirect, a footer read. That's latency, so more threads barely help.

- Under about 1 MB, a file takes longer to open than to read. A 3.5 MB file spends about 25% of its time opening.
- One S2 embedding asset is 7.3 MB, so a file per Item is fine. Sub-megabyte files are what hurt.
- The count is the real problem. 500 files is ~60 s just opening them. Keep a search under ~100 files, or point STAC at a partition instead of a file per Item.
- Row groups matter a lot. 200k × 1,024 in one row group: 3.9 s. In 100 groups of 2,000: 0.12 s.

What comes back



query

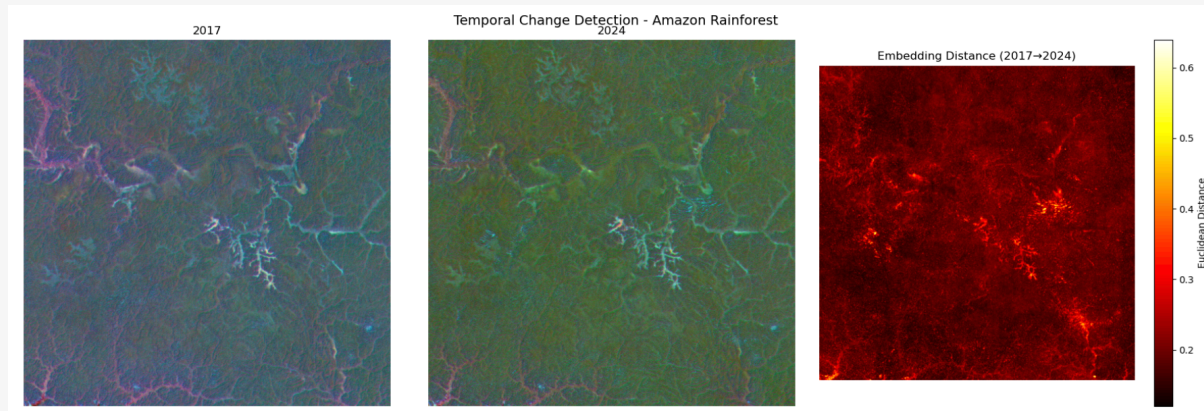


six nearest matches in the embedding space

One example chip, no labels and no classifier. These are its six nearest neighbours.

A REAL SIMILARITY SEARCH, BUT NOT THROUGH THE API ON THE MAIN SLIDES. THAT DOESN'T EXIST YET.

Change through time



Same place, two years, and the distance between them. The next thing you want is the imagery for both years. For AlphaEarth you can't get to the scenes it was made from. And an annual embedding misses anything that came and went inside the year.

Amazon, AlphaEarth 2017 and 2024 – both panels are embeddings shown as colour, not imagery · right: per-pixel distance between the two years · figure: [geospatial-jeff/aef-mosaic](#) (Apache-2.0) · The AlphaEarth Foundations Satellite Embedding dataset is produced by Google and Google DeepMind.

Why keep the sources?

When a match looks wrong, the first thing you want is the original pixels. Was it cloud? A seam? A missing date? You need to see **what the model actually saw**.

You also need the sources to:

- reproduce the preprocessing and inference
- rebuild everything when the model changes
- decide whether it's good enough for a use nobody planned for

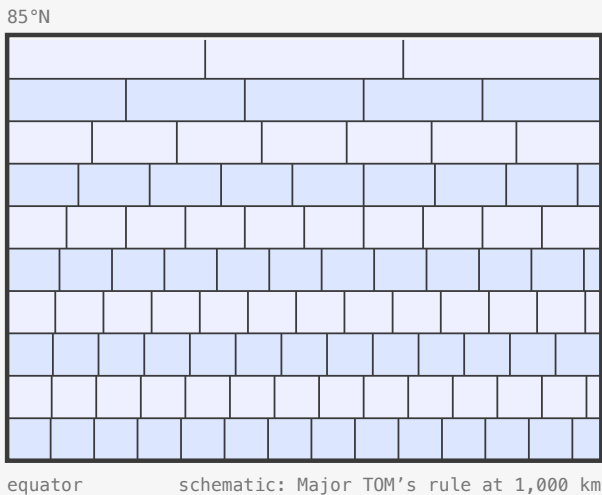
Some source data cannot be published

Sometimes you can't publish the sources. AlphaEarth is the obvious case. Then the Item should say what you do know:

- source datasets and date range
- model and processing version
- missing or inaccessible inputs
- licences and access restrictions
- a reproducibility or provenance status

That's enough for someone to decide what they'd use it for.

Major TOM isn't a raster



At the real 10 km spacing

4,008 columns at the equator

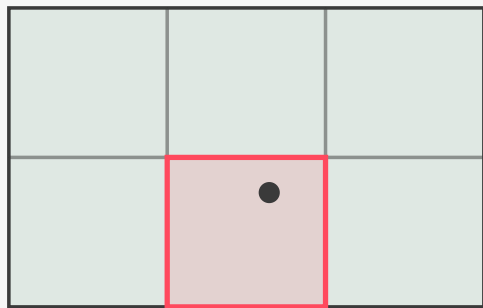
2,004 columns at 60°N

350 columns at 85°N

rows: equal steps in latitude
columns: recounted every row
→ column edges don't line up

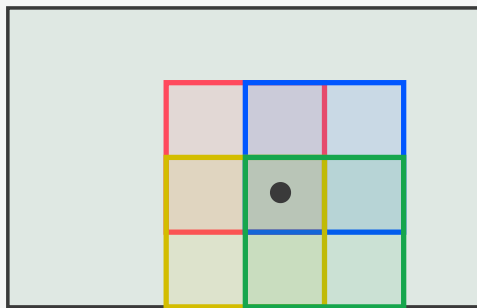
A raster needs one affine transform. Major TOM recounts the columns on every row, so there isn't one. If your cells aren't rectilinear, put the geometry **on the row**. That's GeoParquet.

How much ground does one vector cover?



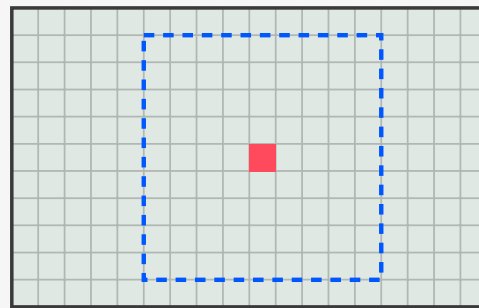
no overlap

1 vector for this place



50% overlap

4 vectors for this place



per pixel

1 vector, read from wider

The model doesn't set the overlap, whoever runs inference does. Record the chip size **and** the stride.

Is AlphaEarth open?

Model	Output	What is open?
Clay 1.5	1,024-d, single observation	Apache-2.0 weights + code
AlphaEarth	64-d per 10 m pixel, annual	CC BY outputs; model closed
TESSERA v2	128-d per 10 m pixel, annual	CC0 weights; MIT code
Prithvi 2.0	768-1,280-d tokens	Apache-2.0
TerraMind	patch tokens per modality	Apache-2.0; corpus CC BY-SA
OlmoEarth 1.2	768-d features	weights, code, data; custom licence

"Open" is really five questions: weights, code, training data, licence, and **can you run inference yourself**. A catalog should record all five.

What does the Collection look like?

```
{
  "type": "Collection",
  "id": "clay-annual-embeddings",
  "license": "CC-BY-4.0",
  "summaries": { "emb:type": ["chip"], "emb:dimensions": [256] },
  "item_assets": {
    "embeddings": {
      "type": "application/vnd.apache.parquet",
      "roles": ["data", "embeddings"],
      "emb:model": "clay-v1.5",
      "emb:distance_metric": "cosine"
    }
  },
  "links": [
    { "rel": "emb:model", "href": "https://huggingface.co/made-with-clay/Clay" },
    { "rel": "emb:source-data",
      "href": "https://earth-search.aws.element84.com/v1/collections/sentinel-2-l2a" }
  ]
}
```

REQUIRED BY THE DRAFT: THE TWO LINKS. PROPOSED: `EMB:\*` IN `ITEM\_ASSETS`.

Who's built this before?

When	Who	What
2016-05	Terrapattern — Levin, Newbury, McDonald (CMU)	visual search over aerial imagery. First.
2017-03	Descartes Labs GeoVisual Search	two months before the <code>stac-spec</code> repo existed
2024	Clay <code>stacchip</code>	chip ids built from the STAC Item id
2024-07	Earth Genome	Items with an embeddings Parquet asset, before any extension
2026-04	TerraBit — Corley, Robinson	49.8 M Clay vectors in DuckDB-WASM, no backend
None of them used STAC search as the narrowing step.		

Why not H3, or another DGGS?

- Nobody shipping embeddings uses one. Clay, Earth Index, AlphaEarth and TESSERA all use the imagery's own UTM / MGRS grid, and Major TOM made its own.
- OGC API – DGGS Part 1 was adopted on 2025-10-24, so it's there if you want it.
- The asset doesn't care. Store real geometry on each row and add whatever grid id you like as a column.