Explaining Point Features in the AF2 Structure Module

Study Notes

December 8, 2025

1 Introduction

This note explains how "point features" arise in the invariant point attention (IPA) mechanism of the AlphaFold 2 structure module, and how they relate to standard Transformer attention (queries, keys, values), frames, local vs. global coordinates, and distance-based attention.

The discussion is structured in two main parts:

1. Locate the quoted text and ideas in the AF2 manuscript and supplement.
2. Build a layered explanation of point features starting from ordinary self-attention and moving up to IPA.

2 Location of the Quoted Text in AF2

Pieces of the quoted description are directly in the supplementary material and main paper; the rest is a faithful paraphrase.

From Supplementary Methods 1.8.2, "Invariant point attention (IPA)":

> Invariant Point Attention (IPA) (Suppl. Fig. 8 and Algorithm 22) is a form of attention that acts on a set of frames (parametrized as Euclidean transforms (T_i)) and is invariant under global Euclidean transformations (T_{global}) on said frames.

From Algorithm 22, the definition of IPA (schematically) is:

```
def InvariantPointAttention({s_i}, {z_ij}, {T_i},
    N_head = 12, c = 16,
    N_query_points = 4, N_point_values = 8):
```

and the first projection steps can be written as

$$\begin{aligned} q_i^h, k_i^h, v_i^h &= \text{LinearNoBias}(s_i), && \in \mathbb{R}^c, \\ \tilde{q}_i^{hp}, \tilde{k}_i^{hp} &= \text{LinearNoBias}(s_i), && \in \mathbb{R}^3, \quad p = 1, \ldots, N_{\text{query points}}, \\ \tilde{v}_i^{hp} &= \text{LinearNoBias}(s_i), && \in \mathbb{R}^3, \quad p = 1, \ldots, N_{\text{point values}}. \end{aligned}$$

Supplementary Figure 8 labels the shapes of the point arrays as:

query pts. $(r_q, h, p, 3)$, key pts. $(r_v, h, p, 3)$, value pts. $(r_v, h, p', 3)$.

In the main paper, the IPA description says:

> The IPA augments each of the usual attention queries, keys and values with 3D points that are produced in the local frame of each residue ... The 3D queries and keys also impose a strong spatial/locality bias on the attention.

So:

- "IPA acts on a set of frames $\{T_i\}$" is verbatim.
- "We use $N_{\text{query pts}} = 4$ and $N_{\text{point values}} = 8$" comes from the default arguments of Algorithm 22.
- The sentence about scalar features being linearly projected to queries, keys, values and point features with shape $N_{\text{res}} \times N_{\text{pts}} \times 3$ is not word-for-word in the paper, but is exactly what Algorithm 22 plus Fig. 8 imply.

Thus, the snippet you are working from is a correct paraphrase of the supplementary material.

3 From Standard Attention to Point Features

We now build up the notion of point features by starting from plain self-attention and adding the AF2 structure-module ingredients.

3.1 Standard Self-Attention in Plain Language

Consider a sequence of tokens indexed by i (for proteins, residues). Each has a feature vector

$$s_i \in \mathbb{R}^{d_{\text{model}}}.$$

In a standard Transformer head, you compute:

$$\begin{aligned} q_i &= W_Q s_i \in \mathbb{R}^{d_{\text{head}}}, \\ k_i &= W_K s_i \in \mathbb{R}^{d_{\text{head}}}, \\ v_i &= W_V s_i \in \mathbb{R}^{d_{\text{head}}}, \end{aligned}$$

where W_Q, W_K, W_V are learned matrices.

The attention score (logit) from position i to j is

$$\ell_{ij} = \frac{1}{\sqrt{d_{\text{head}}}} q_i^\top k_j,$$

and the attention weight is

$$a_{ij} = \text{softmax}_j(\ell_{ij}).$$

The output for position i is

$$o_i = \sum_j a_{ij} v_j.$$

In this mental model:

- Queries, keys, values are learned vectors in an abstract feature space.
- The score is a dot product between query and key.
- There is no explicit notion of geometry; positional information (if any) is injected via positional encodings or relative position embeddings.

3.2 What Changes in the AF2 Structure Module?

In the structure module, each residue carries more structure:

- A scalar feature vector s_i,
- A backbone frame T_i that describes where this residue currently is in 3D. This is a rigid transform, unpacked below.

IPA's job is to update the scalar features s_i while taking into account both

- the pair features z_{ij} (a pairwise "memory" between residues), and
- the current geometry encoded in the frames T_i.

The key idea is:

> Do self-attention not only in feature space (dot products) but also in 3D space using learned points, and make the 3D part invariant to how the whole protein is rotated or translated.

This is where point features and local/global frames enter.

4 Frames and Local vs. Global Coordinates

This is often the most confusing piece, so we go slowly.

4.1 What Is a Frame T_i?

For each residue i, AlphaFold maintains a rigid body transform

$$T_i = (R_i, t_i),$$

where

- $R_i \in \mathbb{R}^{3\times 3}$ is a rotation matrix,
- $t_i \in \mathbb{R}^3$ is a translation vector.

You can think of T_i as a small coordinate system attached to the residue's backbone (constructed from N, Cα, C using Gram–Schmidt).

We distinguish:

- **Local coordinates**: a point expressed in residue i's own frame, $\tilde{x} \in \mathbb{R}^3$.
- **Global coordinates**: the same point in a shared 3D space, $x \in \mathbb{R}^3$.

The relationship is:

- Local to global:
$$x = T_i \circ \tilde{x} = R_i \tilde{x} + t_i.$$
- Global to local (using the inverse transform):
$$\tilde{x} = T_i^{-1} \circ x = R_i^\top (x - t_i).$$

Conceptually:

- Each residue has its own ruler and axes (local frame).
- The backbone frame tells you where to place that local frame inside the global protein coordinate system.
- If you update (R_i, t_i), you move the residue in 3D space, but any point described in local coordinates moves with it.

4.2 Why Care About Local vs. Global?

We want the network to ignore arbitrary choices like "I drew the whole protein rotated 90°" in the global frame.

If we represented everything directly in global coordinates as features, then a global rotation or translation would change every coordinate and the network would have to relearn invariance. This is undesirable.

Instead, AlphaFold:

- Stores features in local frames whenever possible,
- Uses frames T_i explicitly whenever geometry must be compared.

This yields two key properties:

1. Quantities expressed in local frames can remain the same even if we rotate the whole protein.
2. Distances between points in global space are invariant to a common rigid motion applied to all of them.

IPA is designed to exploit these facts.

5 Two Kinds of Features in IPA: Scalar and Point

We can now explain point features.

Within IPA, each residue i has:

- The usual scalar features $s_i \in \mathbb{R}^{c_m}$,
- A backbone frame T_i.

From s_i the algorithm creates two different families of features.

5.1 Scalar Q / K / V (Standard Part)

Algorithm 22, line 1:

$$q_i^h, k_i^h, v_i^h = \text{LinearNoBias}(s_i), \quad q_i^h, k_i^h, v_i^h \in \mathbb{R}^c.$$

This is exactly the standard Transformer pattern:

- For each head h and residue i we get a query vector q_i^h, a key vector k_i^h, and a value vector v_i^h, each in $\mathbb{R}^c$.

If IPA stopped here, we would just have regular attention on scalars (plus a pair bias term).

5.2 Point Q / K / V (Geometric Part)

Now the new part.

Algorithm 22, lines 2–3:

$$\begin{aligned} \tilde{q}_i^{hp}, \tilde{k}_i^{hp} &= \text{LinearNoBias}(s_i), \quad \tilde{q}_i^{hp}, \tilde{k}_i^{hp} \in \mathbb{R}^3, \quad p = 1, \ldots, N_{\text{query points}}, \\ \tilde{v}_i^{hp} &= \text{LinearNoBias}(s_i), \quad \tilde{v}_i^{hp} \in \mathbb{R}^3, \qquad p = 1, \ldots, N_{\text{point values}}. \end{aligned}$$

Interpretation:

- For each head h and residue i, we get a small set of 3D points.
- For queries and keys: $p = 1, \ldots, N_{\text{query points}} = 4$.
- For value points: $p = 1, \ldots, N_{\text{point values}} = 8$.

Stacking across residues, heads, points gives tensors of shapes:

- Query points: $\tilde{q} \in \mathbb{R}^{N_{\text{res}} \times N_{\text{head}} \times N_{\text{query points}} \times 3}$,
- Key points: same shape,
- Value points: $\mathbb{R}^{N_{\text{res}} \times N_{\text{head}} \times N_{\text{point values}} \times 3}$.

These are what we call point features. They are features in the sense that they are learned functions of s_i, but they live in $\mathbb{R}^3$ and are interpreted as coordinates of points in the local frame of residue i.

You can think of them as a small cluster of "sensors" or "virtual atoms" attached to each residue, whose relative arrangement encodes a learned preferred geometry.

Per head and per residue we now have:

- A scalar query vector q_i^h,
- A set of query points in 3D, $\{\tilde{q}_i^{hp}\}_p$,
- Similarly, key points and value points.

This is the crucial mental shift:

> In IPA, each attention head has two representations of a residue: one in an abstract feature space (the usual Q/K/V vectors) and another as a small constellation of learned points in 3D.

6 Why Four Query Points and Eight Value Points?

The specific numbers come from the default arguments of Algorithm 22.

Intuitively:

- With a single point per residue one can only represent distances between residues,
- With several points, one can capture richer shape information: directions, orientations, and local "patches" of space around the residue.

You can imagine four query points as defining a small tetrahedron around the residue, and eight value points as a richer constellation. The network learns how to place these points in the residue-local frame so that they encode geometry in a useful way.

These numbers are kept small to keep computation manageable while still giving each head enough geometric expressive power.

7 Attention Logits: Combining Dot Products and Distances

We now connect this to the mental model of attention as "dot products" and show precisely how distances enter.

Algorithm 22, line 7 gives the attention weights:

$$a_{ij}^h = \text{softmax}_j \left(w_L \left[\sqrt{\frac{1}{c}}\, q_i^{h\top} k_j^h + b_{ij}^h - \gamma_h w_C^2 \sum_p \left\| T_i \circ \tilde{q}_i^{hp} - T_j \circ \tilde{k}_j^{hp} \right\|^2 \right] \right).$$

We break this expression down term by term.

7.1 The Standard Part

The term

$$\sqrt{\frac{1}{c}}\, q_i^{h\top} k_j^h$$

is exactly the standard dot-product attention score. The factor $1/\sqrt{c}$ is the usual variance-normalization used in Transformers.

The term b_{ij}^h is a learned bias from the pair representation z_{ij}. It depends only on the residue pair and not on frames; it is obtained via a linear map

$$b_{ij}^h = \text{LinearNoBias}(z_{ij}).$$

If we kept only these two terms, we would have ordinary attention with a learned pairwise bias, familiar from many Transformer variants.

7.2 The New Geometric Term

The geometric term is

$$-\gamma_h w_C^2 \sum_p \left\| T_i \circ \tilde{q}_i^{hp} - T_j \circ \tilde{k}_j^{hp} \right\|^2.$$

For a fixed head h and residues i, j:

1. Take each query point in residue i in local coordinates, $\tilde{q}_i^{hp}$.

2. Map it to global coordinates using the frame:

$$x_{i,\text{global}}^{hp} = T_i \circ \tilde{q}_i^{hp} = R_i \tilde{q}_i^{hp} + t_i.$$

3. Take each key point in residue j in local coordinates, $\tilde{k}_j^{hp}$, and map to global:

$$x_{j,\text{global}}^{hp} = T_j \circ \tilde{k}_j^{hp} = R_j \tilde{k}_j^{hp} + t_j.$$

4. Compute the squared Euclidean distance between corresponding global points and sum over p:

$$\sum_p \left\| x_{i,\text{global}}^{hp} - x_{j,\text{global}}^{hp} \right\|^2.$$

5. Multiply by $\gamma_h w_C^2$ and place a minus sign in front.

Because of the minus sign, if the predicted point constellations of residues i and j are far apart in global space (for head h), this term is large and negative and pushes the logit down. If the constellations are close, the penalty is small.

We can read this as:

> Head h prefers that residue i attends to residues j whose learned key-point constellations lie close in 3D to the learned query-point constellations of i, under the current frames.

The total logit thus has three contributions:

- Content similarity (dot product in feature space),
- Pair-bias term (from z_{ij}),
- Distance penalty term (from 3D geometry).

So the intuition that attention is "about dot products" remains valid: IPA extends this with additional additive terms, one of which is geometrical.

7.3 Why Squared Distances, and Why Use Frames?

The distance term is built using global coordinates obtained from local points and frames. The crucial property is:

> Euclidean distance between points is invariant to a global rigid motion.

If we replace all frames with

$$T_i' = T_{\text{global}} \circ T_i$$

for a fixed rigid transform T_{global} (a rotation and translation applied uniformly to every residue), then

$$\left\| T_i' \circ \tilde{q}_i^{hp} - T_j' \circ \tilde{k}_j^{hp} \right\|^2 = \left\| T_{\text{global}} \circ (T_i \circ \tilde{q}_i^{hp}) - T_{\text{global}} \circ (T_j \circ \tilde{k}_j^{hp}) \right\|^2 = \left\| T_i \circ \tilde{q}_i^{hp} - T_j \circ \tilde{k}_j^{hp} \right\|^2.$$

The last equality holds because rigid transforms preserve distances. The paper spells this out as the proof of IPA's invariance.

Thus:

- The dot-product and pair-bias terms do not depend on frames at all,
- The distance term depends on frames only through distances that are invariant to a global transform.

It follows that each logit ℓ_{ij}^h and each attention weight a_{ij}^h is invariant to the global orientation or location of the protein.

This resolves the question "how have distances entered attention?":

- They are computed from learned point features plus the current frames,
- They act as an additional bias in the logits,
- They are designed precisely to be invariant to global SE(3) actions.

8 Outputs of IPA: Scalar and Point Outputs

Once we have attention weights a_{ij}^h, IPA produces three kinds of outputs per head, which are concatenated and linearly projected back into the single representation.

1. **Pair-weighted pair representation**

$$\tilde{o}_i^h = \sum_j a_{ij}^h z_{ij}.$$

   This pulls information directly from the pair representation into the single representation.

2. **Standard scalar value aggregation**

$$o_i^h = \sum_j a_{ij}^h v_j^h.$$

   This is ordinary self-attention on scalar values.

3. **Point aggregation with frames**

   Algorithm 22, line 10:

$$\tilde{o}_i^{hp} = T_i^{-1} \circ \left(\sum_j a_{ij}^h (T_j \circ \tilde{v}_j^{hp}) \right).$$

   Step by step:

   - For each residue j, map its value point $\tilde{v}_j^{hp}$ from local to global:

$$x_{j,\text{global}}^{hp} = T_j \circ \tilde{v}_j^{hp}.$$

   - Take the attention-weighted sum in global space:

$$y_{i,\text{global}}^{hp} = \sum_j a_{ij}^h x_{j,\text{global}}^{hp},$$

     which is like a barycenter of neighbors' value points.

- Map that global point back to residue i's local frame:

$$\tilde{o}_i^{hp} = T_i^{-1} \circ y_{i,\text{global}}^{hp}.$$

Thus $\tilde{o}_i^{hp}$ is a new set of point features in residue i's local coordinates, summarizing geometry suggested by the neighbors.

The invariance logic is analogous to the one used for the logits:

- If we apply a global transform T_{global} to all T_i, each $T_j \circ \tilde{v}_j^{hp}$ gets transformed by T_{global},
- The sum in global space is transformed by T_{global},
- The inverse frame of residue i picks up T_{global}^{-1}, cancelling the effect.

So the local coordinates $\tilde{o}_i^{hp}$ are unchanged: despite moving through global coordinates, the final point outputs live in local frames and are invariant to global roto-translations.

Finally, IPA concatenates:

- all $\tilde{o}_i^h$ (pair-aggregation outputs),
- all o_i^h (scalar outputs),
- all $\tilde{o}_i^{hp}$ and their norms,

then applies a linear layer back to the single-representation dimension.

9 Putting It All Together in a Transformer Mental Model

We can summarize IPA in language close to standard Transformer practice.

9.1 If You Strip Out Geometry

If we remove:

- the point projections $\tilde{q}, \tilde{k}, \tilde{v}$,
- the distance term in the logits,
- the point-valued output aggregation,

and keep:

- scalar Q/K/V from s_i,
- pair bias b_{ij}^h,
- scalar value aggregation,

then IPA reduces to a standard multi-head self-attention block on the single representation s_i with an extra pair-bias term.

9.2 What IPA Adds, in Transformer Language

IPA adds two ideas already familiar in Transformer literature, but in a geometric form:

1. **Relative positional bias in attention logits.**

   Many Transformers add a term

   $$\text{bias}(i, j)$$

   to ℓ_{ij} that depends on relative position $i - j$ or graph structure. IPA's distance term is a continuous, learned version of this where "position" is not a chain index but learned points in 3D. One can view

   $$-\gamma_h w_C^2 \sum_p \left\| T_i \circ \tilde{q}_i^{hp} - T_j \circ \tilde{k}_j^{hp} \right\|^2$$

   as an SE(3)-invariant relative positional bias.

2. **Geometric value channels.**

   Instead of values being only scalars $v_j^h \in \mathbb{R}^c$, IPA also uses value points $\tilde{v}_j^{hp} \in \mathbb{R}^3$. The aggregation

   $$\tilde{o}_i^{hp} = T_i^{-1} \circ \sum_j a_{ij}^h \left(T_j \circ \tilde{v}_j^{hp} \right)$$

   is an SE(3)-equivariant value aggregation: if we rotate and translate everything, the local point outputs transform correctly (or, when viewed back in local frames, remain invariant).

Thus, in compact form:

> IPA is a multi-head self-attention block on scalars, augmented with a co-attention on a set of 3D points per residue and head, constructed so that:
>
> - The attention weights are invariant to global rotation and translation,
> - The point outputs transform correctly under SE(3),
> - The overall computation still looks like queries, keys, values, softmax, and weighted sums.

9.3 Why Call Them "Point Features"?

In equivariant networks, channels are often categorized by how they transform:

- Scalars: unchanged by rotation (e.g. temperature),
- Vectors/points: rotate like coordinates (e.g. displacement vectors).

All of these are still "features" in the sense of learned activations, just with different transformation rules.

So point features in AF2 are:

> Learned three-dimensional features that transform as points under rigid motions, which the model uses to construct geometric biases in attention.

They are point-like because we literally plug them into formulas involving positions, distances, and norms.

10 Implementation-Style Summary

Sometimes it is helpful to think in pseudocode to connect with standard attention implementations. In rough Python-like pseudocode:

```
# s: [N_res, c_m]
# z: [N_res, N_res, c_z]
# T: list of frames (R_i, t_i)

# 1. Scalar Q, K, V (like standard attention)
qkv = linear_qkv(s)   # [N_res, 3 * H * c]
q, k, v = split_and_reshape(qkv, shape=[N_res, H, c])

# 2. Point Q, K, V (new)
q_pts = linear_q_pts(s)   # [N_res, H * P_q * 3] -> [N_res, H, P_q, 3]
k_pts = linear_k_pts(s)   # same shape as q_pts
v_pts = linear_v_pts(s)   # [N_res, H, P_v, 3]

# 3. Pair bias
b = linear_pair(z)        # [N_res, N_res, H]

# 4. Compute logits for each head h, pair (i, j)
for h in heads:
  # scalar dot-product term
  scalar_logit_ij = (q[i, h] @ k[j, h]) / sqrt(c)

  # geometric term
  geo_penalty_ij = 0
  for p in range(P_q):
      # local to global
      qi_global = T[i].apply(q_pts[i, h, p])
      kj_global = T[j].apply(k_pts[j, h, p])
      # squared distance
      geo_penalty_ij += || qi_global - kj_global ||^2

  logit_ij = wL * (scalar_logit_ij + b[i, j, h]
                   - gamma[h] * wC**2 * geo_penalty_ij)

# 5. Softmax over j -> a[i, j, h]

# 6. Outputs
o_scalar = sum_j a[i, j, h] * v[j, h]        # standard value aggregation
o_pair   = sum_j a[i, j, h] * z[i, j]        # pair-weighted pair repr

# Point output: aggregate in global, map back to local
for p in range(P_v):
  y_global = sum_j a[i, j, h] * T[j].apply(v_pts[j, h, p])
  o_pts[i, h, p] = T[i].inv().apply(y_global)  # back to local
```

```
# 7. Concatenate [o_scalar, o_pair, o_pts, ||o_pts||] across heads and points
#    then project back to s-dimension with a Linear layer
s_out = linear_concat(...)
```

From the perspective of standard attention code:

- The lines computing q, k, v are identical to usual practice,
- The distance-based term is an extra contribution to the logits built from the additional projections $q_{\text{pts}}, k_{\text{pts}}$ and frames T_i,
- The point-valued outputs are a special kind of value aggregation with coordinate transforms.

11 Plain-English Recap

We conclude with a very informal summary.

- Each residue has a vector of numbers (its standard embedding) and a local 3D coordinate system (its frame).
- From these numbers, each attention head constructs:
  - ordinary query, key, value vectors (as in Transformers),
  - a small cloud of 3D query points, a cloud of 3D key points, and a cloud of 3D value points in the residue's local coordinates.
- The current backbone frames tell us where these local clouds sit in global 3D space.
- The attention head decides "who attends to whom" using:
  - dot products of query and key vectors (content similarity),
  - a learned bias from the pair representation,
  - a penalty proportional to the squared distances between query and key point clouds in 3D.
- It aggregates both scalar values and value point clouds with the attention weights; for the point clouds, it:
  - moves each neighbor's value points into global space,
  - averages them,
  - moves the result back into the current residue's local frame.
- All geometric components are built so that rotating or translating the entire protein does not change attention decisions or outputs in local coordinates.

Thus, point features are not mysterious new objects; they are simply

> 3D coordinates of a small, learned constellation of points per residue and head, produced by linear projections of the same scalar features that produce Q/K/V.

They allow attention to depend directly on geometry (via distances) while still fitting into the usual Q/K/V plus softmax pattern and guaranteeing SE(3) invariance.