4 Coordinate Free Geometry

Coordinate free geometry (CFG) is a style of expressing geometric objects and relations that avoids unnecessary reliance on any specific coordinate system. Representing geometric quantities in terms of coordinates can frequently lead to confusion, and to derivations that rely on irrelevant coordinate systems.

We first define the basic quantities:

1. A **scalar** is just a real number.
2. A **point** is a location in space. It *does not* have any intrinsic coordinates.
3. A **vector** is a direction and a magnitude. It *does not* have any intrinsic coordinates.

A point is not a vector: we cannot add two points together. We cannot compute the magnitude of a point, or the location of a vector.
Coordinate free geometry defines a restricted class of operations on points and vectors, even though both are represented as vectors in matrix algebra. The following operations are the **only** operations allowed in CFG.

1. $\|\vec{v}\|$: magnitude of a vector.
2. $\bar{p}_1 + \vec{v}_1 = \bar{p}_2$, or $\vec{v}_1 = \bar{p}_2 - \bar{p}_1$.: point-vector addition.
3. $\vec{v}_1 + \vec{v}_2 = \vec{v}_3$.: vector addition
4. $\alpha\vec{v}_1 = \vec{v}_2$: vector scaling. If $\alpha > 0$, then $\vec{v}_2$ is a new vector with the same direction as $\vec{v}_1$, but magnitude $\alpha\|\vec{v}_1\|$. If $\alpha < 0$, then the direction of the vector is reversed.
5. $\vec{v}_1 \cdot \vec{v}_2$: dot product $= \|\vec{v}_1\|\|\vec{v}_2\|\cos(\theta)$, where θ is the angle between the vectors.
6. $\vec{v}_1 \times \vec{v}_2$: cross product, where $\vec{v}_1$ and $\vec{v}_2$ are 3D vectors. Produces a new vector perpedicular to $\vec{v}_1$ and to $\vec{v}_2$, with magnitude $\|\vec{v}_1\|\|\vec{v}_2\|\sin(\theta)$. The orientation of the vector is determined by the right-hand rule (see textbook).
7. $\sum_i \alpha_i\vec{v}_i = \vec{v}$: Linear combination of vectors
8. $\sum_i \alpha_i\bar{p}_i = \bar{p}$, **if** $\sum_i \alpha_i = 1$: affine combination of points.
9. $\sum_i \alpha_i\bar{p}_i = \vec{v}$, **if** $\sum_i \alpha_i = 0$

Example:

- $\bar{p}_1 + (\bar{p}_2 - \bar{p}_3) = \bar{p}_1 + \vec{v} = \bar{p}_4$.
- $\alpha\bar{p}_2 - \alpha\bar{p}_1 = \alpha\vec{v}_1 = \vec{v}_2$.
- $\frac{1}{2}(p_1 + p_2) = p_1 + \frac{1}{2}(\bar{p}_2 - \bar{p}_1) = \bar{p}_1 + \frac{1}{2}\vec{v} = \bar{p}_3$.

Note:
In order to understand these formulas, try drawing some pictures to illustrate different cases (like the ones that were drawn in class).

Note that operations that are *not* in the list are undefined.
These operations have a number of basic properties, e.g., commutivity of dot product: $\vec{v}_1 \cdot \vec{v}_2 = \vec{v}_2 \cdot \vec{v}_1$, distributivity of dot product: $\vec{v}_1 \cdot (\vec{v}_2 + \vec{v}_3) = \vec{v}_1 \cdot \vec{v}_2 + \vec{v}_1 \cdot \vec{v}_3$.
CFG helps us reason about geometry in several ways:

1. When reasoning about geometric objects, we only care about the intrinsic geometric properties of the objects, not their coordinates. CFG prevents us from introducing irrelevant concepts into our reasoning.

2. CFG derivations usually provide much more geometric intuition for the steps and for the results. It is often easy to interpret the meaning of a CFG formula, whereas a coordinate-based formula is usually quite opaque.

3. CFG derivations are usually simpler than using coordinates, since introducing coordinates often creates many more variables.

4. CFG provides a sort of "type-checking" for geometric reasoning. For example, if you derive a formula that includes a term $\bar{p} \cdot \vec{v}$, that is, a "point dot vector," then there may be a bug in your reasoning. In this way, CFG is analogous to type-checking in compilers. Although you could do all programming in assembly language — which does not do type-checking and will happily led you add, say, a floating point value to a function pointer — most people would prefer to use a compiler which performs type-checking and can thus find many bugs.

In order to *implement* geometric algorithms we need to use coordinates. These coordinates are part of the representation of geometry — they are not fundamental to reasoning about geometry itself.

Example:
CFG says that we cannot add two points; there is no meaning to this operation. But what happens if we try to do so anyway, using coordinates?
Suppose we have two points: $\bar{p}_0 = (0, 0)$ and $\bar{p}_1 = (1, 1)$, and we add them together coordinate-wise: $\bar{p}_2 = \bar{p}_0 + \bar{p}_1 = (1, 1)$. This is not a valid CFG operation, but we have done it anyway just to tempt fate and see what happens. We see that the

resulting point is the same as one of the original points: $\bar{p}_2 = \bar{p}_1$.
Now, on the other hand, suppose the two points were represented in a different coordinate frame: $\bar{q}_0 = (1, 1)$ and $\bar{q}_1 = (2, 2)$. The points $\bar{q}_0$ and $\bar{q}_1$ are the *same* points as $\bar{p}_0$ and $\bar{p}_1$, with the same vector between them, but we have just represented them in a different coordinate frame, i.e., with a different origin. Adding together the points we get $\bar{q}_2 = \bar{q}_0 + \bar{q}_1 = (3, 3)$. This is a *different* point from $\bar{q}_0$ and $\bar{q}_1$, whereas before we got the same point.
The geometric relationship of the result of adding two points depends on the coordinate system. There is no clear geometric interpretation for adding two points.

Aside:
It is actually possible to define CFG with far fewer axioms than the ones listed above. For example, the linear combination of vectors is simply addition and scaling of vectors.