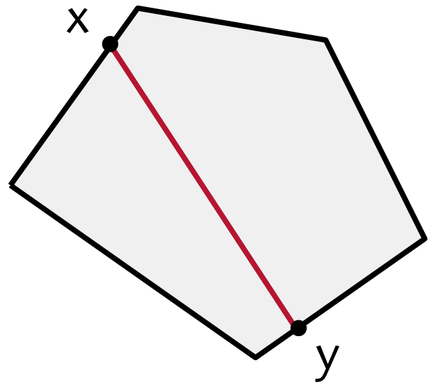
Collision Detection for Convex Objects

- Definition of “convex polyhedron”:

$$P \subset \mathbb{R}^3 \quad \text{convex} \Leftrightarrow$$

$$\forall x, y \in P : \overline{xy} \subset P \Leftrightarrow$$

$$P = \bigcap_{i=1\ldots n} H_i \quad , H_i = \text{half-spaces}$$

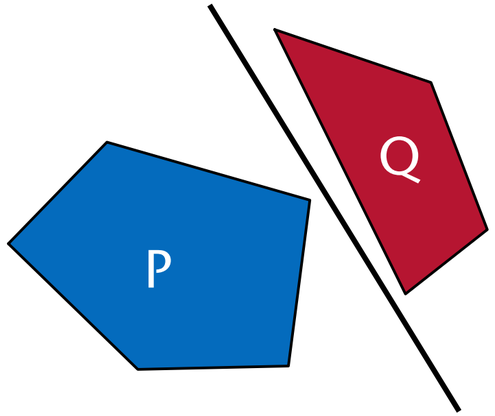


- A condition for "non-collision":

P and Q are "linearly separable" $:\Leftrightarrow$

$\exists$ half-space $H : P \subseteq H^- \wedge Q \subseteq H^+ :\Leftrightarrow$

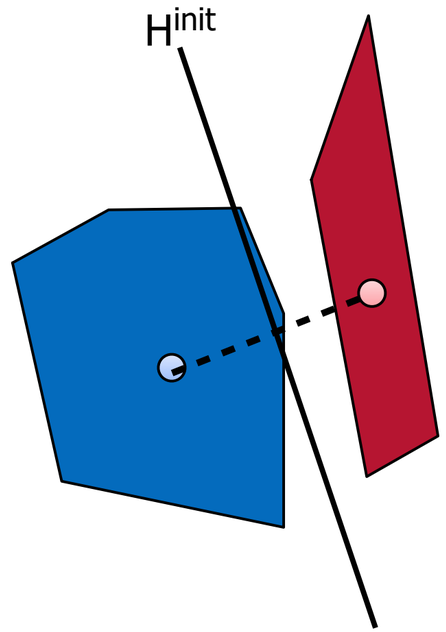
$$\exists \mathbf{h} \in \mathbb{R}^4 \; \forall \mathbf{p} \in P, \mathbf{q} \in Q : \; (\mathbf{p}, 1) \cdot \mathbf{h} > 0 \wedge (\mathbf{q}, 1) \cdot \mathbf{h} < 0$$



Separating plane H

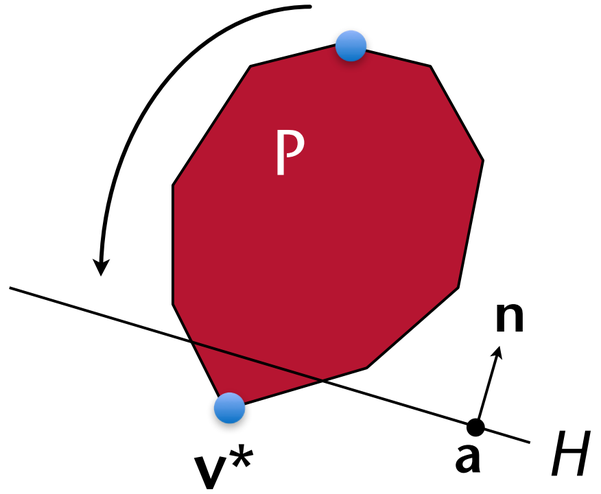
The Separating Planes Algorithm

```
init candidate H:
  connect barycenters of P, Q → line pq
  H := plane perpendicular to pq, half-way between barycenters
repeat max n times
    if exists v ∈ vertices(P) on the back side of H:
        update H such that v is now on the front side of H
    if exists v ∈ vertices(Q) on the front side of H:
        update H such that v is now on the back side of H
    if there are no vertices on the "wrong" side of H, resp.:
        return "no collision"
if there are still vertices on the "wrong" side of H:
    return "collision"   {could be wrong}
```



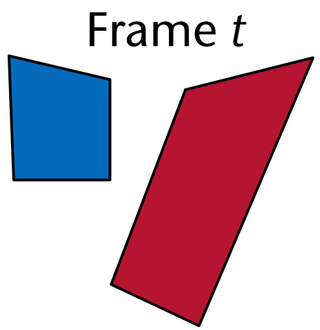
How to Find a Vertex on the "Wrong" Side *Quickly*

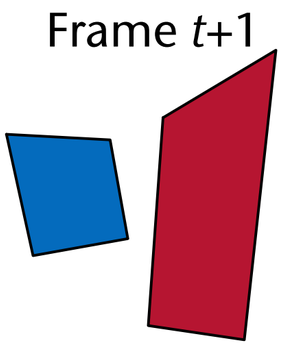
- The brute-force method:
  test all vertices $\mathbf{v}$ whether $f(\mathbf{v}) = (\mathbf{v} - \mathbf{a}) \cdot \mathbf{n} > 0$
- Observation:
  1. f is linear in v_x, v_y, v_z,
  2. P is convex $\Rightarrow$ $f(x)$ has (usually) exactly *one* minimum over all points $\mathbf{x}$ on the surface of P, consequently ..
  3. $\exists^1 \mathbf{v}^* \; : \; f(\mathbf{v}^*) = \min$
- The algorithm (steepest descent on the surface wrt. f):
  - Start with an arbitrary vertex $\mathbf{v}$
  - Walk to that neighbor $\mathbf{v}'$ of $\mathbf{v}$ for which $f(\mathbf{v}') = \min.$ (among all neighbors)
  - Stop if there is no neighbor $\mathbf{v}'$ of $\mathbf{v}$ for which $f(\mathbf{v}') < f(\mathbf{v})$



The Temporally-Coherent Separating Planes Algorithm

- Temporal coherence = objects/positions don't move far from frame to frame
  - Utilized in many algos, e.g., video compression
- Utilize here, too, to improve performance:
  - New separating plane of frame t+1 is (hopefully) just a slight translation/rotation of old plane of frame t
  - New vertex attaining $f = \min$ is close neighbor of old v*
  - Consequently: init main loop with H as of *last frame*, and init findMinVertex with v* found in *last call*
  - Store new separating plane and new v* for the pair P, Q for next frame





Updating the Candidate Plane, *H*

- In the following, represent all vertices **p** as (**p**, 1), i.e., use *homogeneous coords*
- We want $\forall \mathbf{p} \in P : \mathbf{h} \cdot \mathbf{p} > 0$ (and $\forall \mathbf{q} \in Q : \mathbf{h} \cdot \mathbf{q} < 0$)
- Let $\bar{P} \subseteq P$ be the "offending" points for a given plane **h**, i.e. $\forall \mathbf{p} \in \bar{P} : \mathbf{h} \cdot \mathbf{p} < 0$
- Define a cost function $c = c(\mathbf{h}) = -\sum_{\mathbf{p} \in \bar{P}} \mathbf{h} \cdot \mathbf{p}$
- Change **h** so as to drive *c* down towards 0
- Gradient descent: change **h** by negative gradient of *c*, i.e. $\mathbf{h}' = \mathbf{h} - \frac{d}{d\mathbf{h}} c(\mathbf{h})$
- Cost fct *c* is linear in **h**, so $\frac{d}{d\mathbf{h}} c = -\sum_{\mathbf{p} \in \bar{P}} \mathbf{p}$
- Therefore, $\mathbf{h}' = \mathbf{h} + \eta \sum_{\mathbf{p} \in \bar{P}} \mathbf{p}$, with η = "learning speed" (usually $\eta \ll 1$)
- In practice, one decelerates, i.e., $\eta' = 0.97\eta$, to prevent cycling
- (For object Q, some signs need to be changed)

- Perceptron Learning Rule (known in machine learning for a long time):
  whenever we find $\mathbf{p} \in P$ with $\mathbf{h} \cdot \mathbf{p} < 0$, update $\mathbf{h}$ using $\mathbf{h}' = \mathbf{h} + \eta \mathbf{p}$.
  (Analog for Q, with some signs reversed.)
- Theorem:
  If P, Q are linearly separable, then repeated application of the perceptron learning rule will terminate after a finite number of steps.
- Corollary:
  If P, Q are linearly separable, then the algorithm will find a separating plane in a finite number of steps.

  (When algo terminates, none of P, Q's vertices are on the wrong side. I.e., each step brings H closer to the solution.)

Proof of the Theorem

- Let **h*** be a separating plane, w.l.og. $\|\mathbf{h}^*\| = 1$
- There is a d, such that $\forall p \in P : \mathbf{h}^* \cdot \mathbf{p} \geq d > 0 \quad , \quad \forall q \in Q : \mathbf{h}^* \cdot \mathbf{q} \leq -d < 0$
  - Such a value d is called the "margin" of **h***
- Assume further **h*** is optimal w.r.t. the margin d (i.e., has the largest margin)
- Let $V = P \cup \{-\mathbf{q} \,|\, \mathbf{q} \in Q\}$
  - Thus, P, Q is linearly separable $\Leftrightarrow$

$$\forall p \in P : \mathbf{h} \cdot \mathbf{p} > 0 \ \wedge \ \forall q \in Q : \mathbf{h} \cdot \mathbf{q} < 0 \ \Leftrightarrow \ \forall v \in V : \mathbf{h} \cdot \mathbf{v} > 0$$

- Let $\mathbf{v} \in V$ be an "offending" vertex in k-th iteration
- After k iterations, $\mathbf{h}^k = \mathbf{h}^{k-1} + \eta\mathbf{v} = \mathbf{h}^{k-2} + \eta\mathbf{v}' + \eta\mathbf{v} = \ldots = \eta \sum_{\mathbf{v}\in V} k_v \mathbf{v}$ where k_v = #iterations in which $\mathbf{v}$ was the offending vertex
- Consider $\mathbf{h}^*\mathbf{h}^k$:

$$\mathbf{h}^* \cdot \mathbf{h}^k = \mathbf{h}^* \cdot \left(\eta \sum_{\mathbf{v}\in V} k_v \mathbf{v}\right) = \eta \sum_{\mathbf{v}\in V} k_v \mathbf{h}^* \cdot \mathbf{v} \geq \eta d \sum_{\mathbf{v}\in V} k_v = \eta d k$$

- Now, we use a trick to find a lower bound on $|\mathbf{h}^k|$:

$$\|\mathbf{h}^k\|^2 = \|\mathbf{h}^*\|^2 \cdot \|\mathbf{h}^k\|^2 \geq \|\mathbf{h}^* \cdot \mathbf{h}^k\|^2 = \eta^2 d^2 k^2$$

- Now, find an upper bound
- Let $D = \max_{\mathbf{v} \in V} \{ \|\mathbf{v}\| \}$
- Consider one iteration:

$$\|\mathbf{h}^k\|^2 - \|\mathbf{h}^{k-1}\|^2 = \|\mathbf{h}^{k-1} + \eta\mathbf{v}\|^2 - \|\mathbf{h}^{k-1}\|^2$$

$$= \|\mathbf{h}^{k-1}\|^2 + 2\eta\mathbf{h}^{k-1}\mathbf{v} + (\eta\mathbf{v})^2 - \|\mathbf{h}^{k-1}\|^2$$

$$< 0 + \eta^2 D^2$$

- Taking this over k iterations:

$$\|\mathbf{h}^k\|^2 < k\eta^2 D^2 + \|\mathbf{h}^0\|^2$$

- Putting lower and upper bound together gives:

$$\eta^2 d^2 k^2 \leq \|\mathbf{h}^k\|^2 \leq k\eta^2 D^2$$

- Solving for k:

$$k \leq \frac{D^2}{d^2}$$

- In other words, the factor $\frac{D^2}{d^2}$ gives a hint, how many iterations could be needed; i.e., to some extent, $\frac{D}{d}$ is a measure of the "difficulty" of the problem (except, we don't know d or D in advance)

Properties of this Algorithm

+ Expected running time is in O(1)!
  The algo exploits *frame-to-frame coherence*:
  if the objects move only very little, then the algo just checks whether the old separating plane is still a separating plane;
  if the separating plane has to be moved, then the algo is often finished after a few iterations.
+ Works even for deformable objects, so long as they stay convex
– Works only for convex objects
– Could return the wrong answer if P and Q are extremely close but not intersecting (bias)
• Research question: can you find an un-biased (deterministic) variant?

Visualization