# Collision Detection: An Optimization Perspective

**Louis Montaut,** Quentin Le Lidec, Antoine Bambade, Vladimir Petrik, Josef Sivic and Justin Carpentier

ENS | PSL

PR[AI]RIE
PaRis Artificial Intelligence Research InstitutE

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Inria
INVENTEURS DU MONDE NUMÉRIQUE

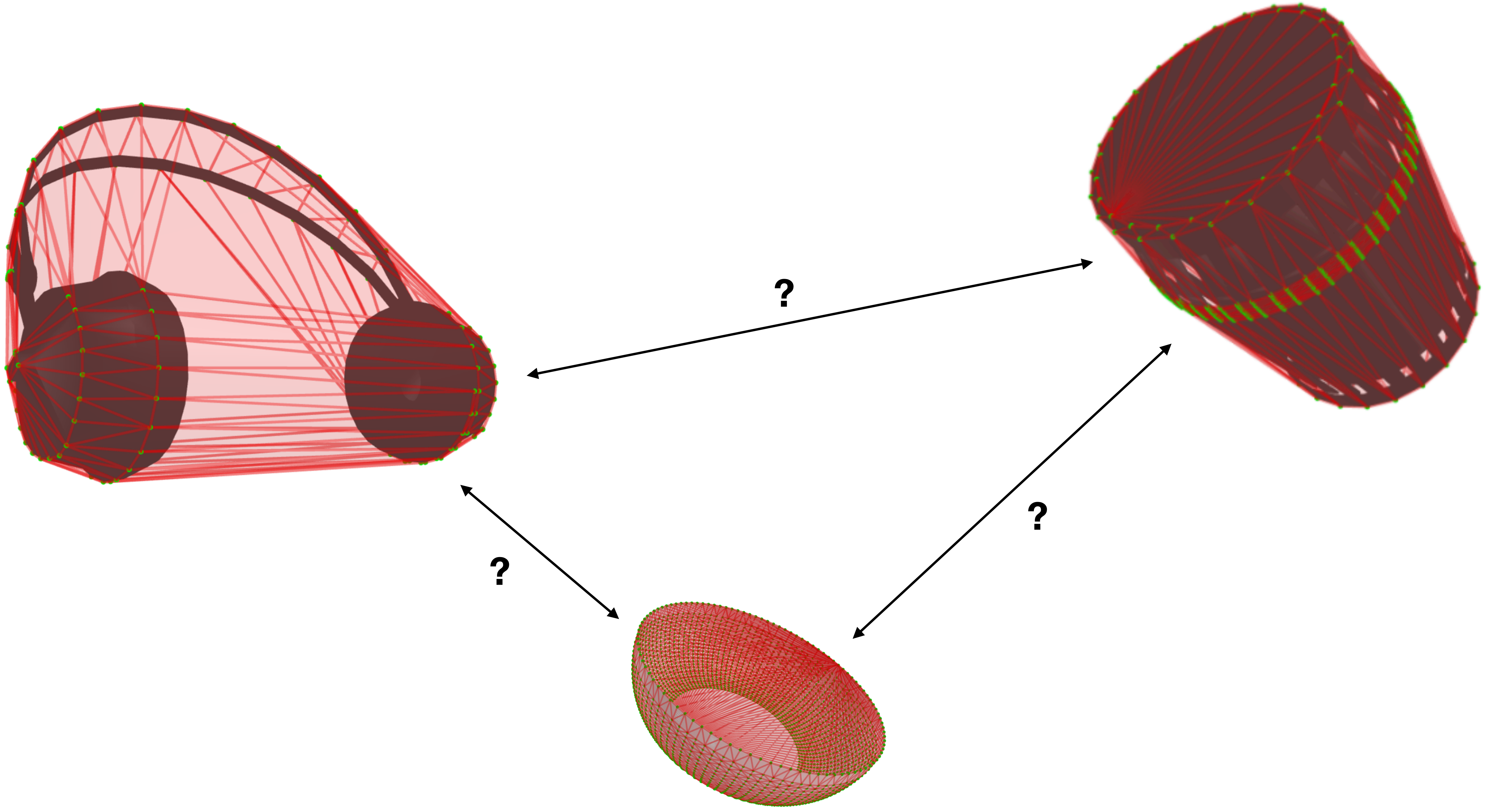
## Step 1 - What is collision detection?

## Step 2 - How to formulate a collision detection problem

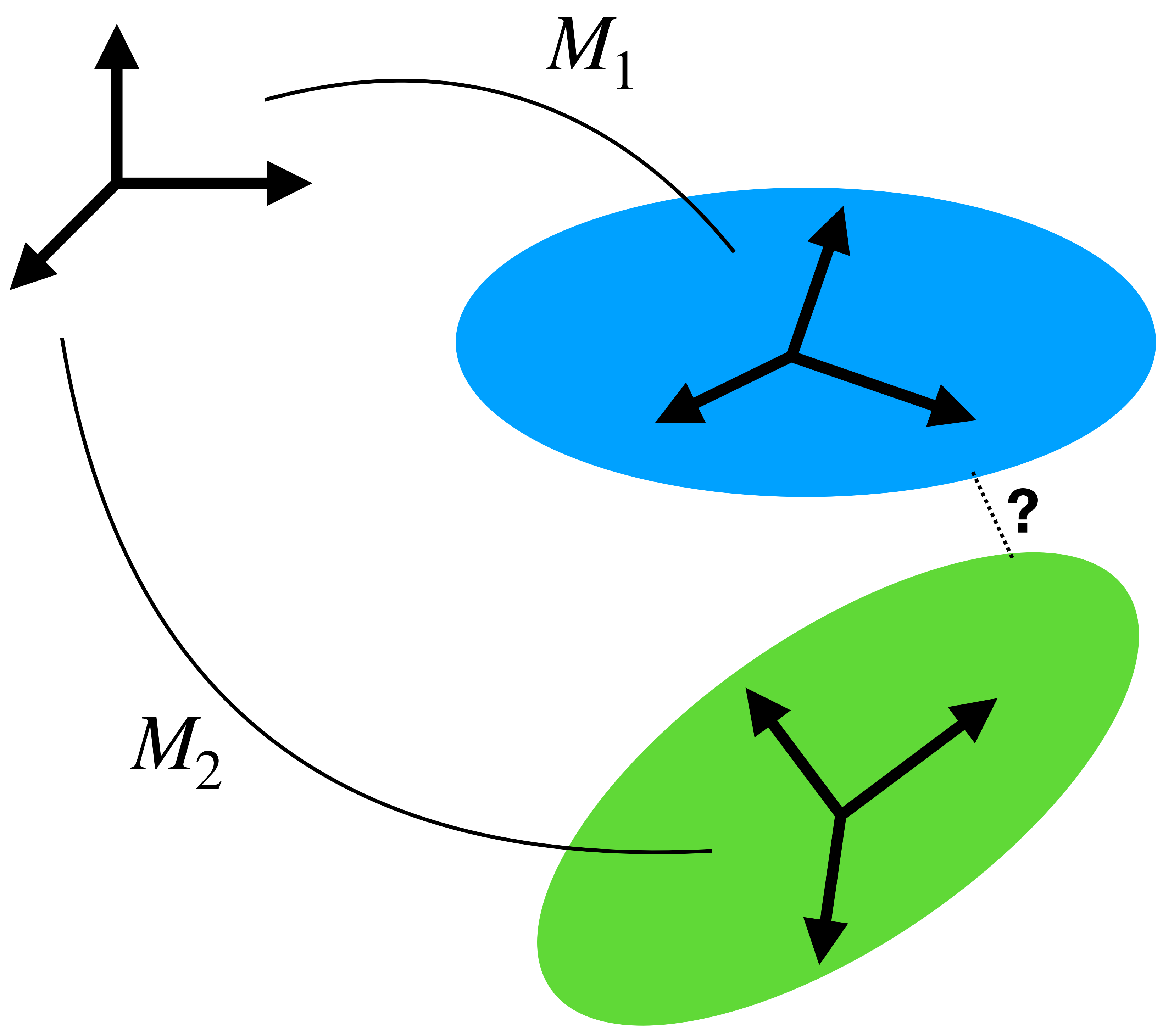
## Step 3 - Solving a collision detection problem with Frank-Wolfe

## Step 4 - Accelerating Frank-Wolfe: the GJK algorithm and beyond

# Step 1 - What is collision detection?



# 1 - HPP-FCL tutorial



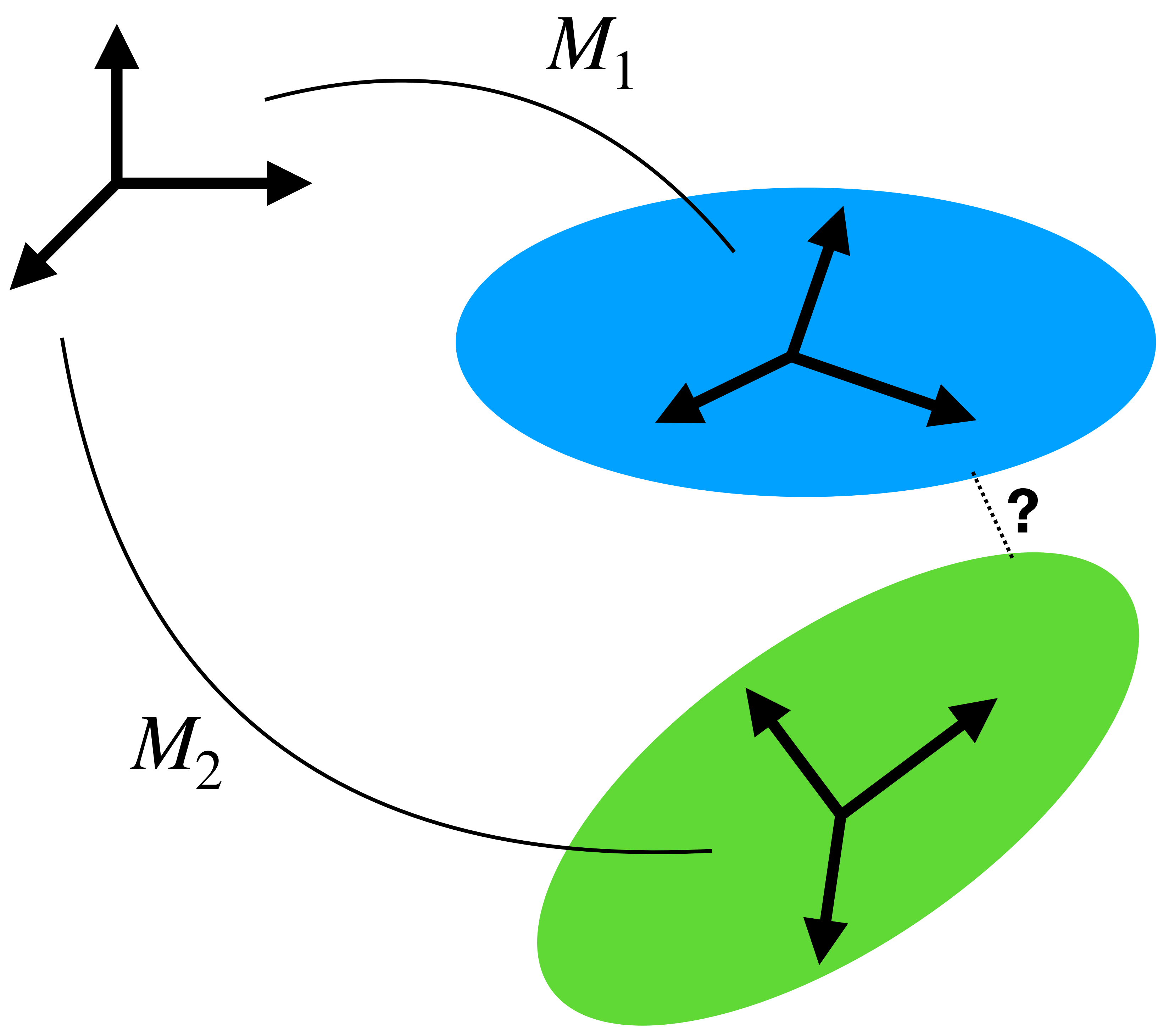

### In the terminal:

```
λ conda install hpp-fcl
```

### In a python script:

```python
import hppfcl
import pinocchio as pin

shape1 = hppfcl.Ellipsoid(np.array([0.2, 0.3, 0.1]))
M1 = pin.SE3.Random()

shape2 = hppfcl.Ellipsoid(np.array([0.4, 0.2, 0.5]))
M2 = pin.SE3.Random()

req = hppfcl.CollisionRequest()
res = hppfcl.CollisionResult()

is_collision = hppfcl.collide(shape1, M1, shape2, M2, req, res)
```

# 1 - HPP-FCL tutorial




**In the terminal:**

```
λ conda install hpp-fcl
```

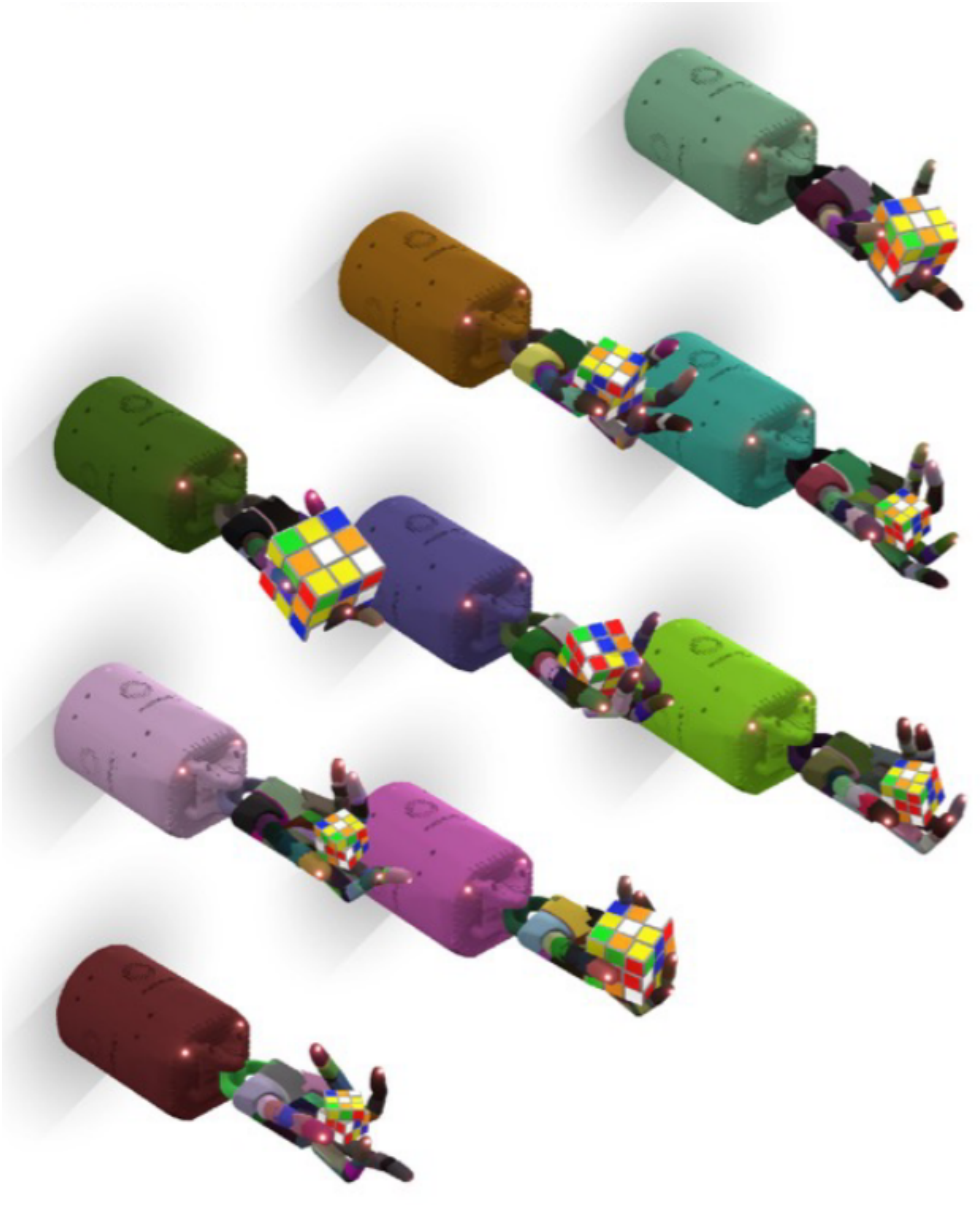
**In a python script:**

```
import hppfcl
import pinocchio as pin

shape1 = hppfcl.Ellipsoid(np.array([0.2, 0.3, 0.1]))
M1 = pin.SE3.Random()

shape2 = hppfcl.Ellipsoid(np.array([0.4, 0.2, 0.5]))
M2 = pin.SE3.Random()

req = hppfcl.CollisionRequest()
res = hppfcl.CollisionResult()

is_collision = hppfcl.collide(shape1, M1, shape2, M2, req, res)
```

↓

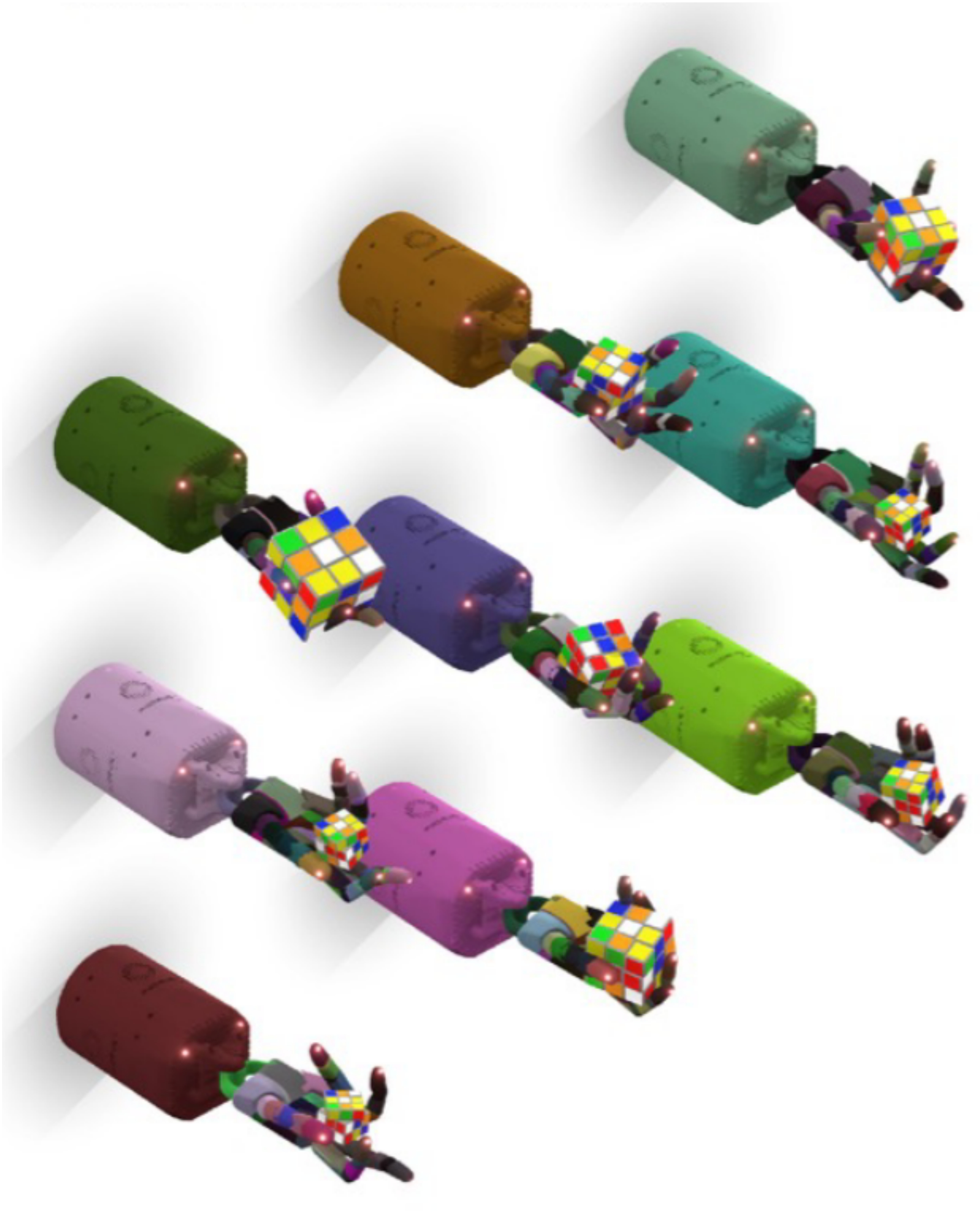
**Calls GJK**

# 1 - Collision detection is a computational bottleneck



- ABA: ~ 1-10 micro-seconds
- Collision detection timing for 1 pair of objects: ~ 1-10 micro-seconds
- Contact solving: ~1-10 micro-seconds

# 1 - Collision detection is a computational bottleneck

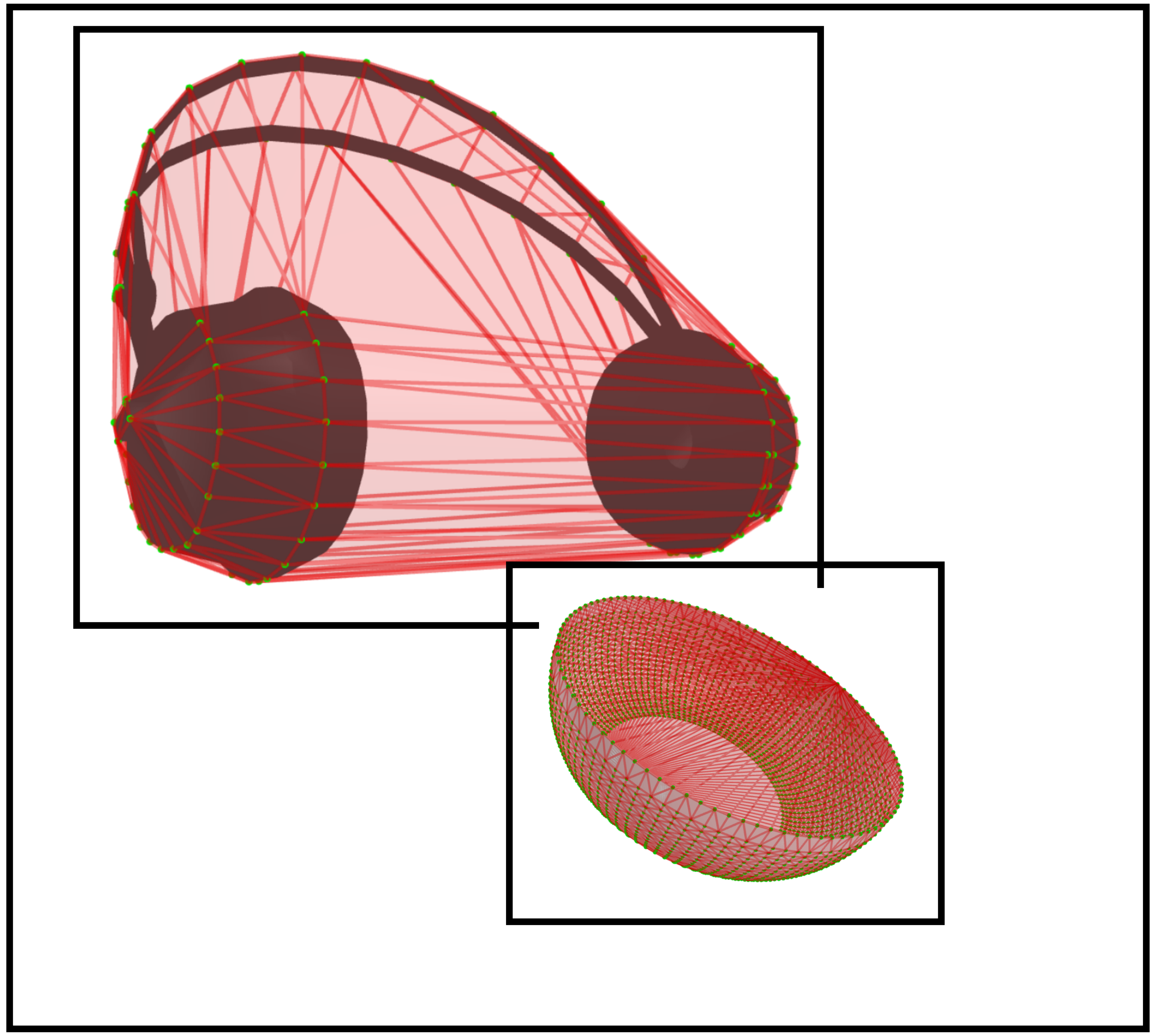


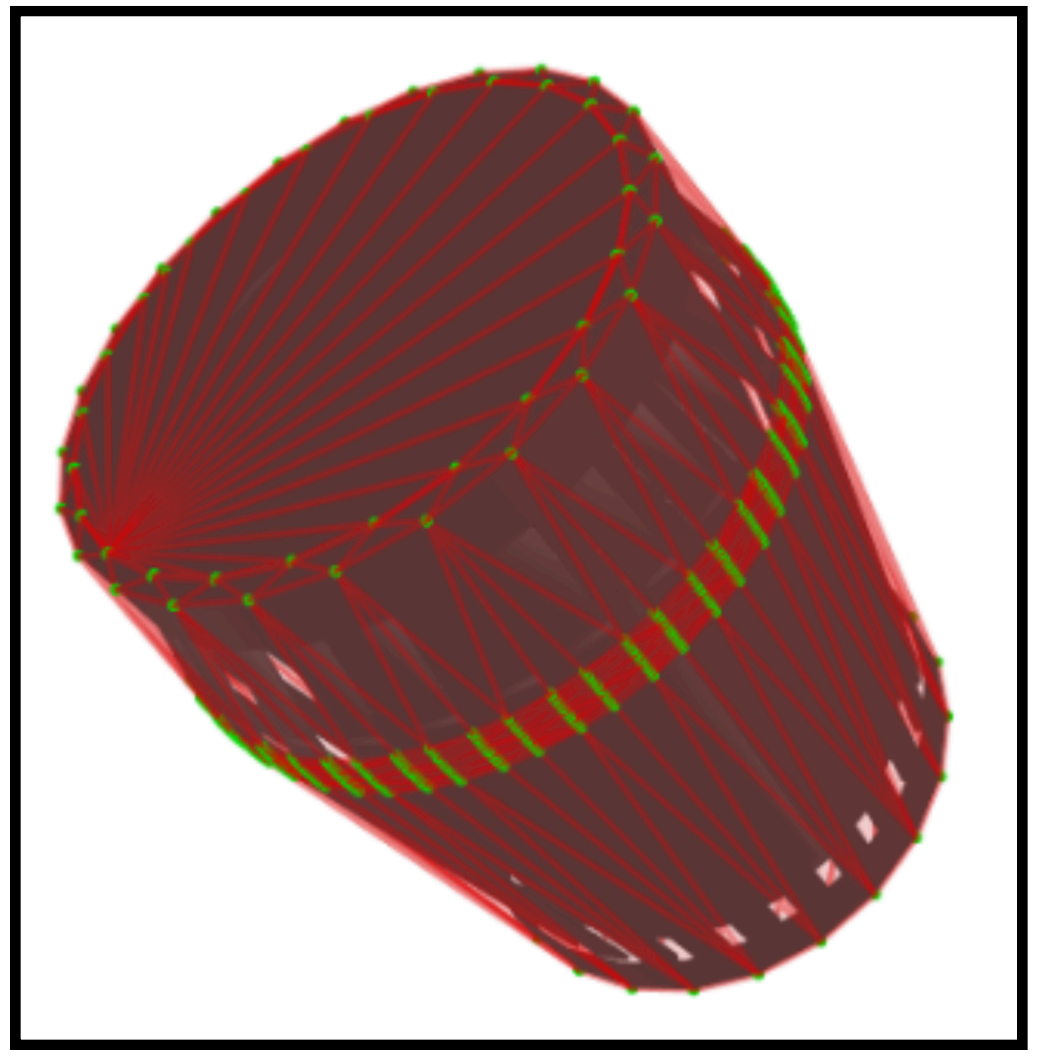
- ABA: ~ 1-10 micro-seconds
- Collision detection timing for 1 pair of objects: ~ 1-10 micro-seconds
- Contact solving: ~1-10 micro-seconds

N objects in a scene
-> O(N x N) possible collision pairs!

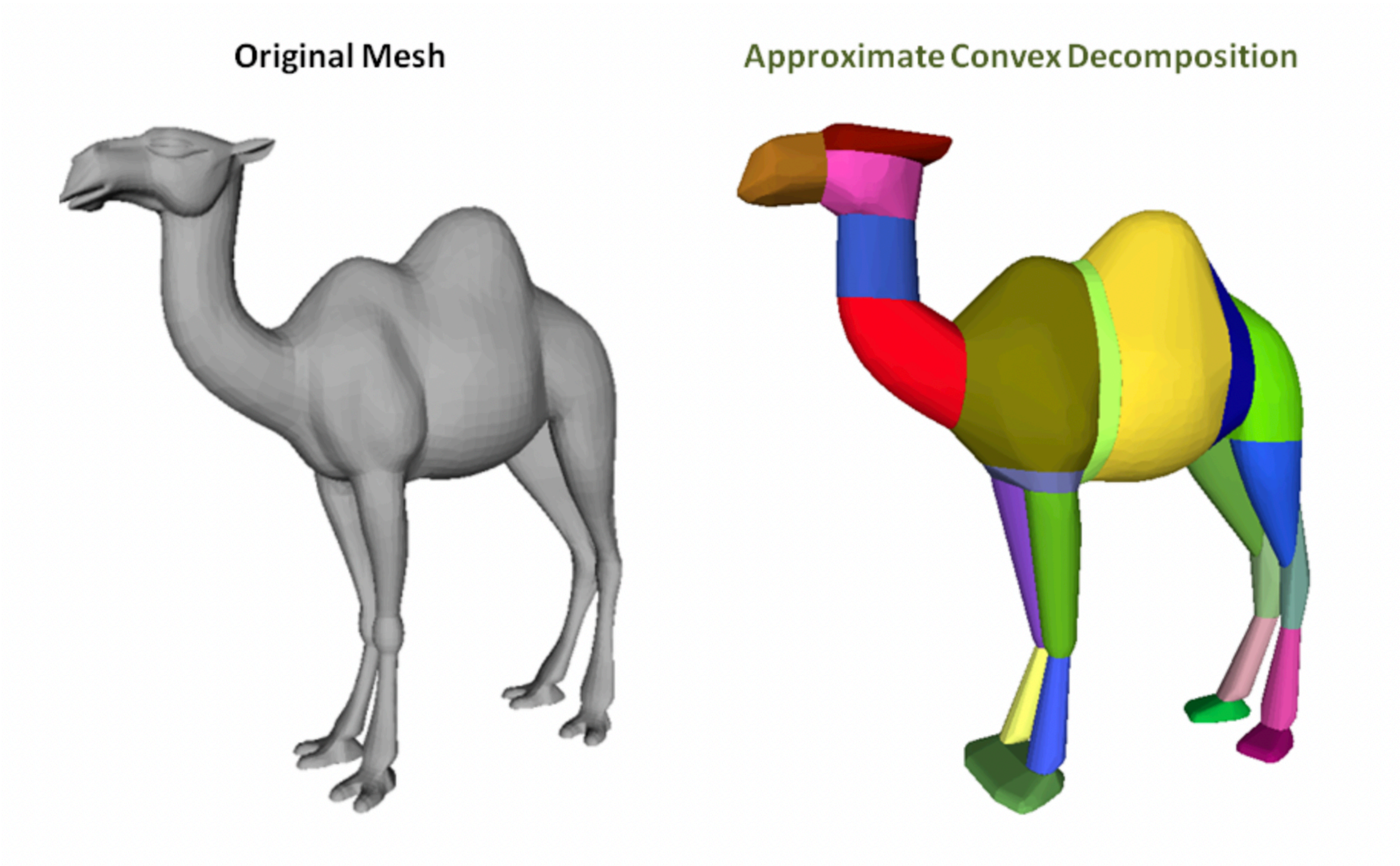
# 1 - Collision detection: broad phase vs. narrow phase

- Use bounding volumes (BVs) to prune collisions
- Only check overlapping BVs





# 1 - Collision detection: convex shapes decomposition

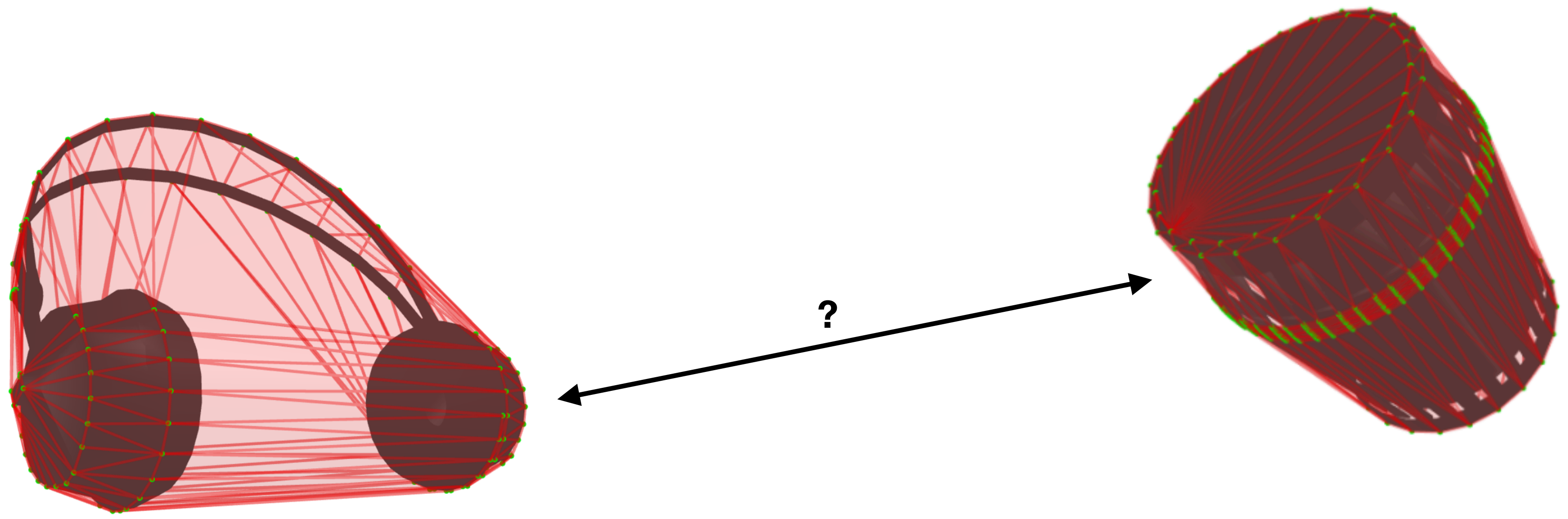



Credit: https://github.com/Unity-Technologies/VHACD

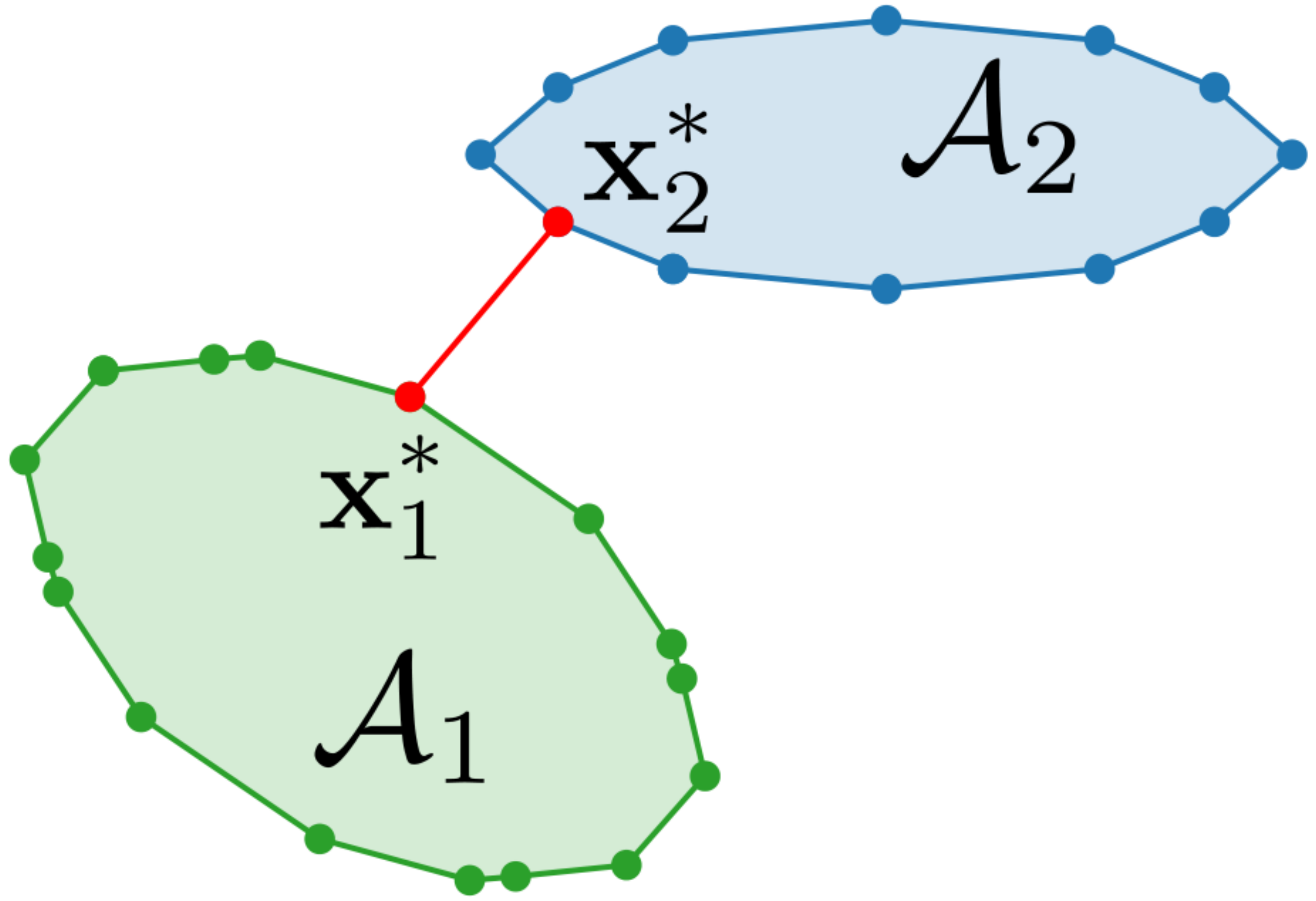
# Step 1 - What is collision detection?

# Step 2 - How to formulate a collision detection problem

# 2 - Collision detection: problem formulation




$$\min_{x_1 \in \mathcal{A}_1, x_2 \in \mathcal{A}_2} \frac{1}{2} || x_1 - x_2 ||^2$$



# 2 - Collision detection: problem formulation

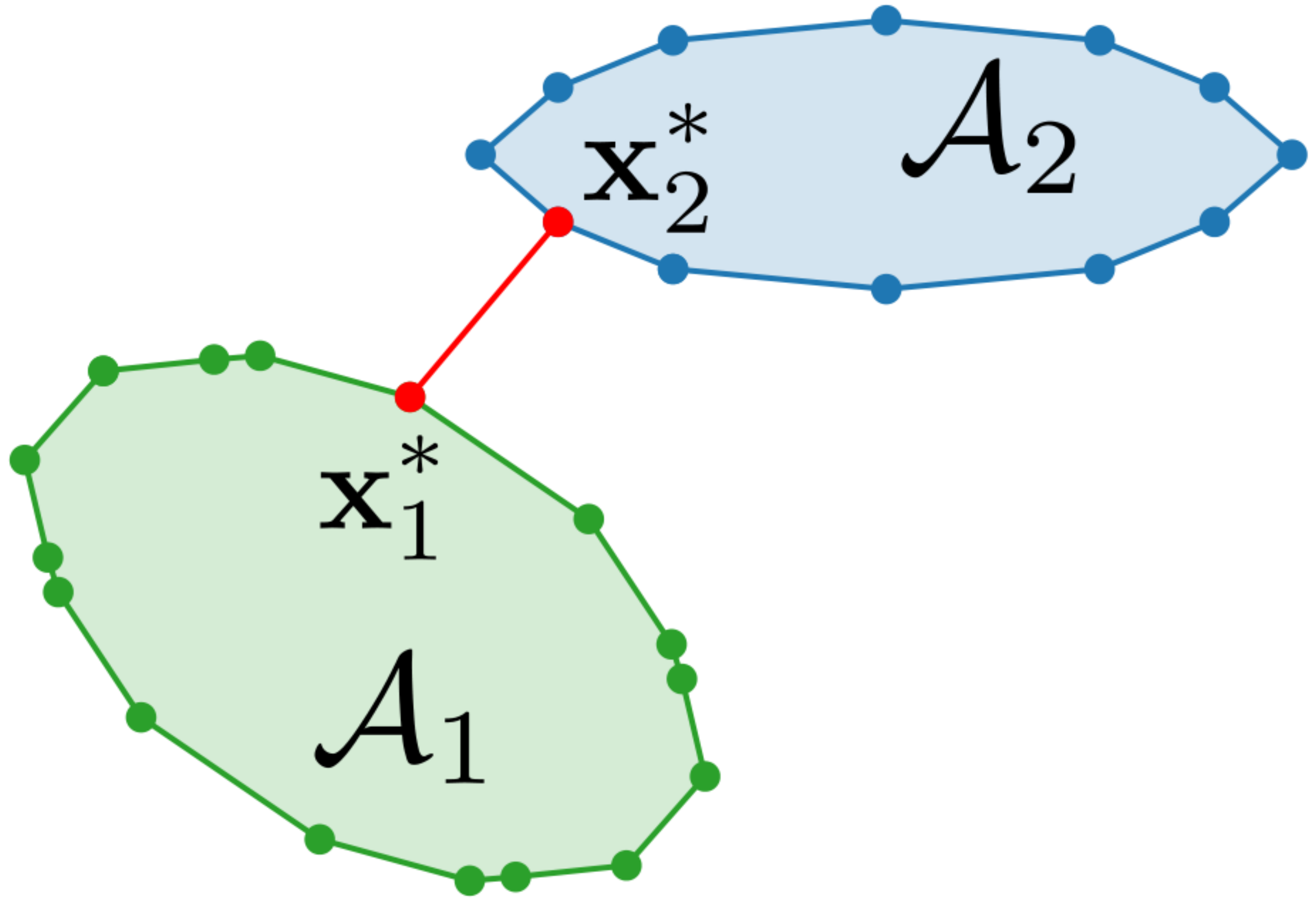
$$\min_{x_1 \in \mathcal{A}_1, x_2 \in \mathcal{A}_2} \frac{1}{2} ||x_1 - x_2||^2$$

**If the shapes are meshes**

$$\min_{x_1, x_2} \frac{1}{2} ||x_1 - x_2||^2$$

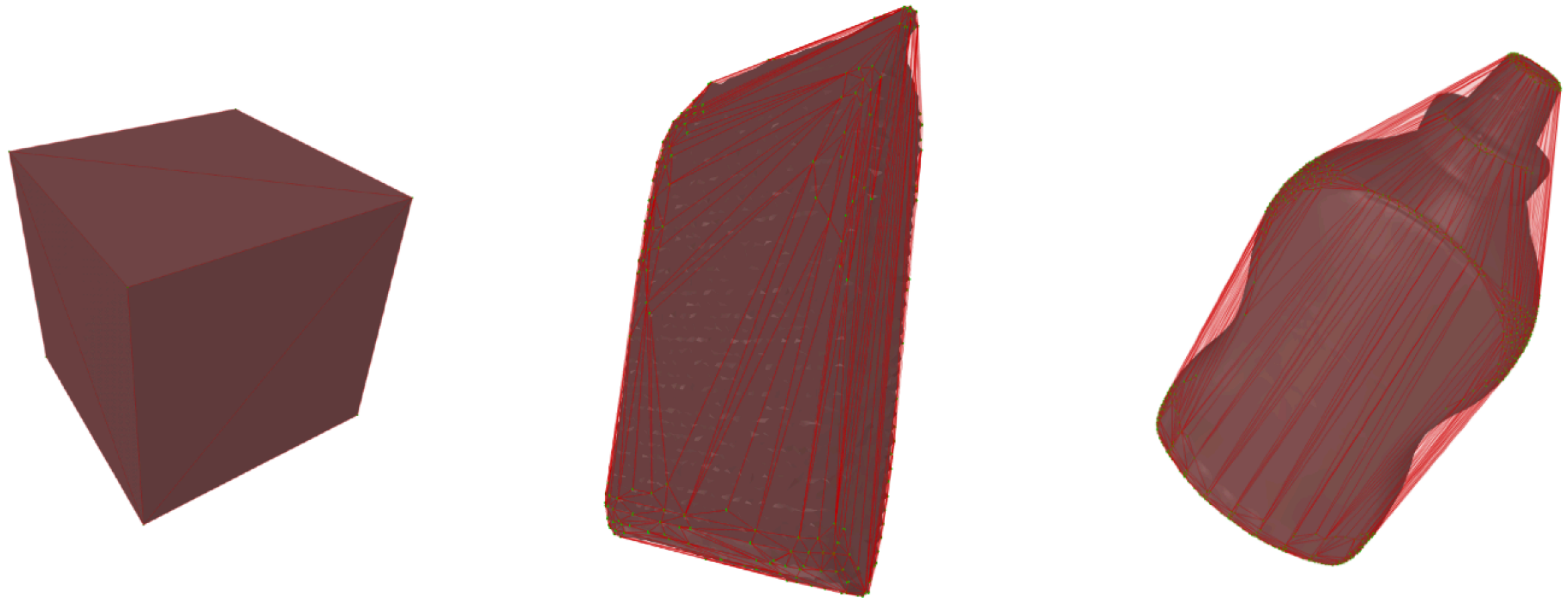
s.t. $A_1 x_1 \leq b_1$

$A_2 x_2 \leq b_2$



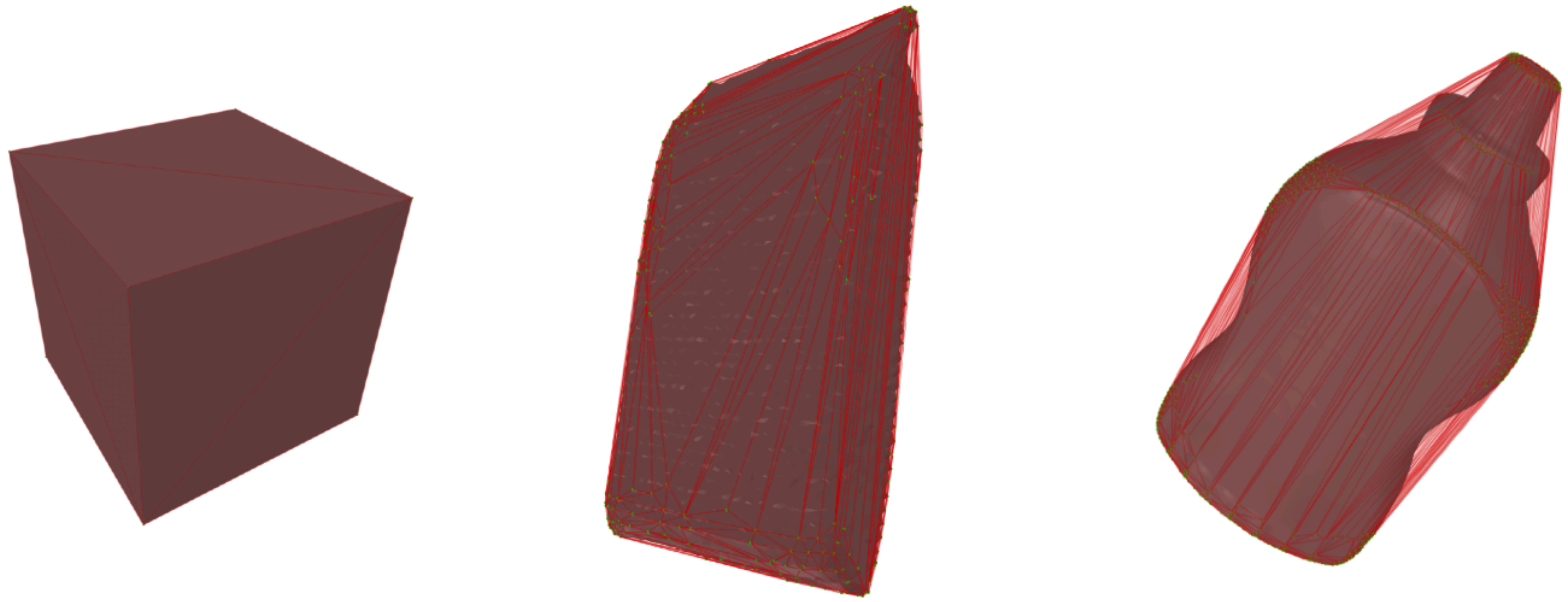

**As many constraints as the number of faces in each polytope!**

# 2 - Collision detection: problem formulation



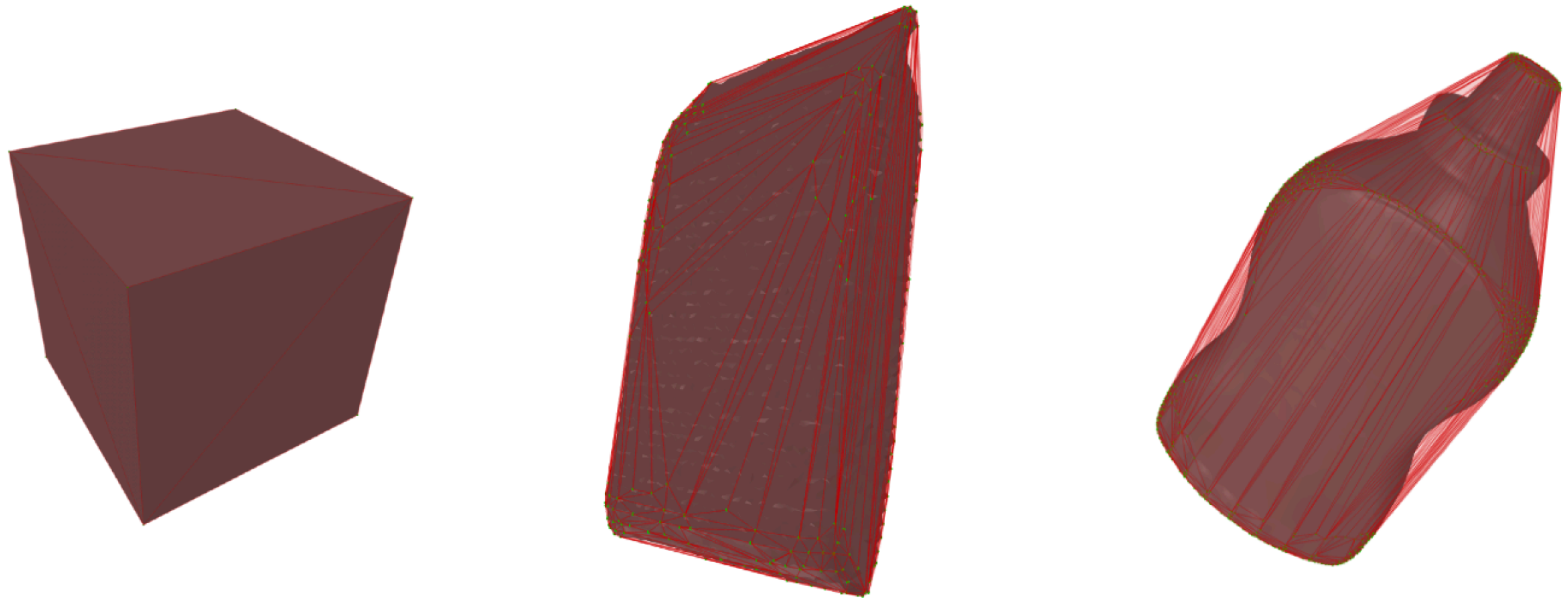
| | $N_v = 8$, $N_f = 6$ | $N_v = 250$, $N_f = 496$ | $N_v = 940$, $N_f = 1876$ |
|---|---|---|---|
| ProxQP | $5.3 \pm 2.7\ \mu s$ | $(2 \pm 0.6) \cdot 10^3\ \mu s$ | $(20 \pm 14) \cdot 10^3\ \mu s$ |

# 2 - Collision detection: problem formulation



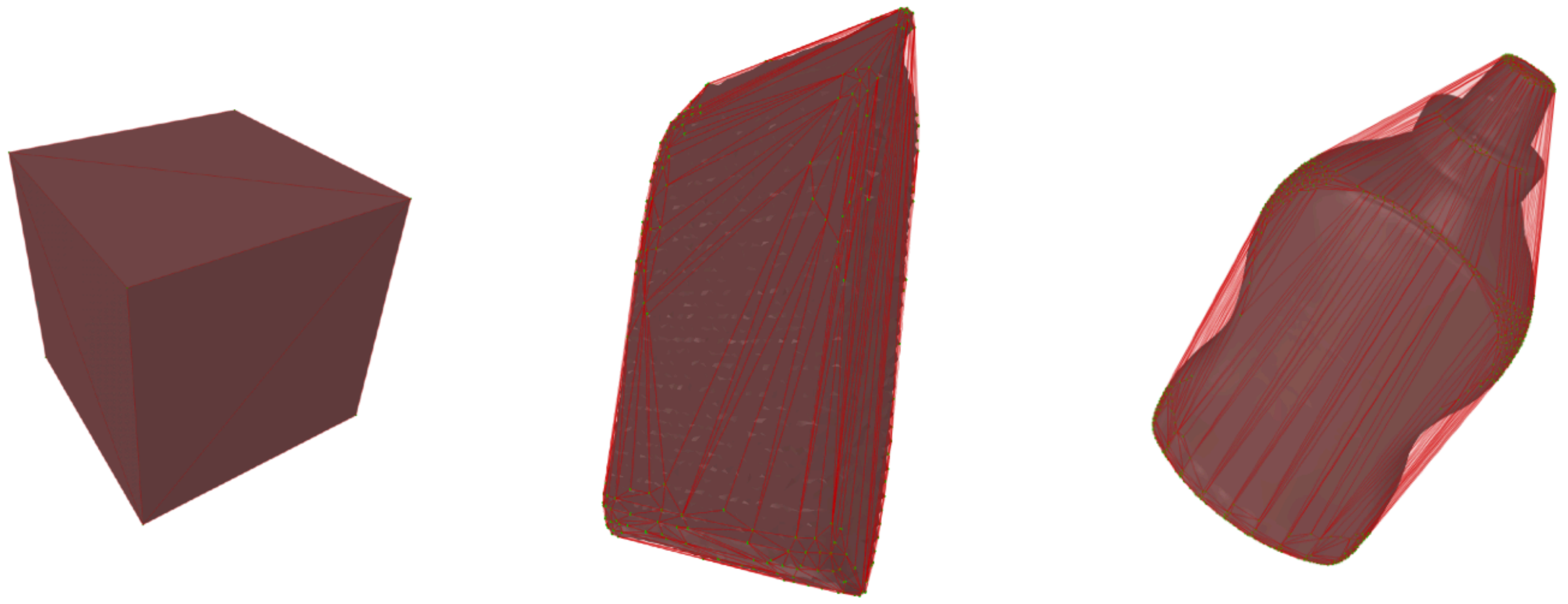
| | $N_v = 8$<br>$N_f = 6$ | $N_v = 250$<br>$N_f = 496$ | $N_v = 940$<br>$N_f = 1876$ |
|---|---|---|---|
| ProxQP | $5.3 \pm 2.7\ \mu s$ | $(2 \pm 0.6) \cdot 10^3\ \mu s$ | $(20 \pm 14) \cdot 10^3\ \mu s$ |
| GJK | $\mathbf{0.2 \pm 0.03}\ \mu s$ | $0.8 \pm 0.3\ \mu s$ | $2.1 \pm 0.5\ \mu s$ |

# 2 - Collision detection: problem formulation



| | $N_v = 8$<br>$N_f = 6$ | $N_v = 250$<br>$N_f = 496$ | $N_v = 940$<br>$N_f = 1876$ |
|---|---|---|---|
| ProxQP | $5.3 \pm 2.7\ \mu s$ | $(2 \pm 0.6) \cdot 10^3\ \mu s$ | $(20 \pm 14) \cdot 10^3\ \mu s$ |
| GJK | $\mathbf{0.2 \pm 0.03}\ \mu s$ | $0.8 \pm 0.3\ \mu s$ | $2.1 \pm 0.5\ \mu s$ |
| Ours | $\mathbf{0.2 \pm 0.05}\ \mu s$ | $\mathbf{0.7 \pm 0.2}\ \mu s$ | $\mathbf{1.4 \pm 0.3}\ \mu s$ |

# 2 - Collision detection: problem formulation

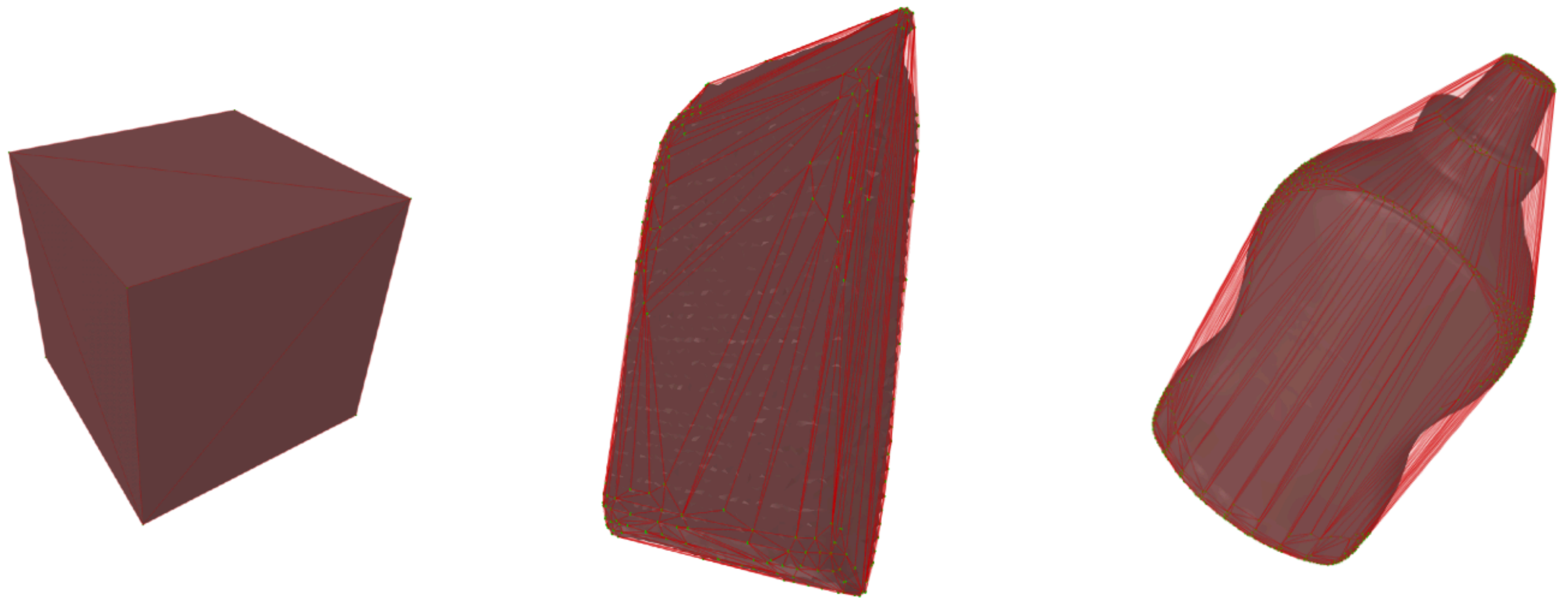


| | $N_v = 8$, $N_f = 6$ | $N_v = 250$, $N_f = 496$ | $N_v = 940$, $N_f = 1876$ |
|---|---|---|---|
| ProxQP | $5.3 \pm 2.7\ \mu s$ | $(2 \pm 0.6) \cdot 10^3\ \mu s$ | $(20 \pm 14) \cdot 10^3\ \mu s$ |
| GJK | $\mathbf{0.2 \pm 0.03}\ \mu s$ | $0.8 \pm 0.3\ \mu s$ | $2.1 \pm 0.5\ \mu s$ |
| Ours | $\mathbf{0.2 \pm 0.05}\ \mu s$ | $\mathbf{0.7 \pm 0.2}\ \mu s$ | $\mathbf{1.4 \pm 0.3}\ \mu s$ |

**What is GJK?**
**Why is it so fast?**

**GJK = Acceleration of Frank-Wolfe applied to a Minimum Norm Point problem (MNP)**

# 2 - Collision detection: problem formulation



| | $N_v = 8$ <br> $N_f = 6$ | $N_v = 250$ <br> $N_f = 496$ | $N_v = 940$ <br> $N_f = 1876$ |
|---|---|---|---|
| ProxQP | $5.3 \pm 2.7$ $\mu$s | $(2 \pm 0.6) \cdot 10^3$ $\mu$s | $(20 \pm 14) \cdot 10^3$ $\mu$s |
| GJK | $\mathbf{0.2 \pm 0.03}$ $\mu$s | $0.8 \pm 0.3$ $\mu$s | $2.1 \pm 0.5$ $\mu$s |
| Ours | $\mathbf{0.2 \pm 0.05}$ $\mu$s | $\mathbf{0.7 \pm 0.2}$ $\mu$s | $\mathbf{1.4 \pm 0.3}$ $\mu$s |

**What is GJK?**
**Why is it so fast?**

**GJK = Acceleration of Frank-Wolfe applied to a Minimum Norm Point problem (MNP)**

- **MNP?**
- **Frank-Wolfe?**
- **Acceleration?**

## Step 1 - What is collision detection?

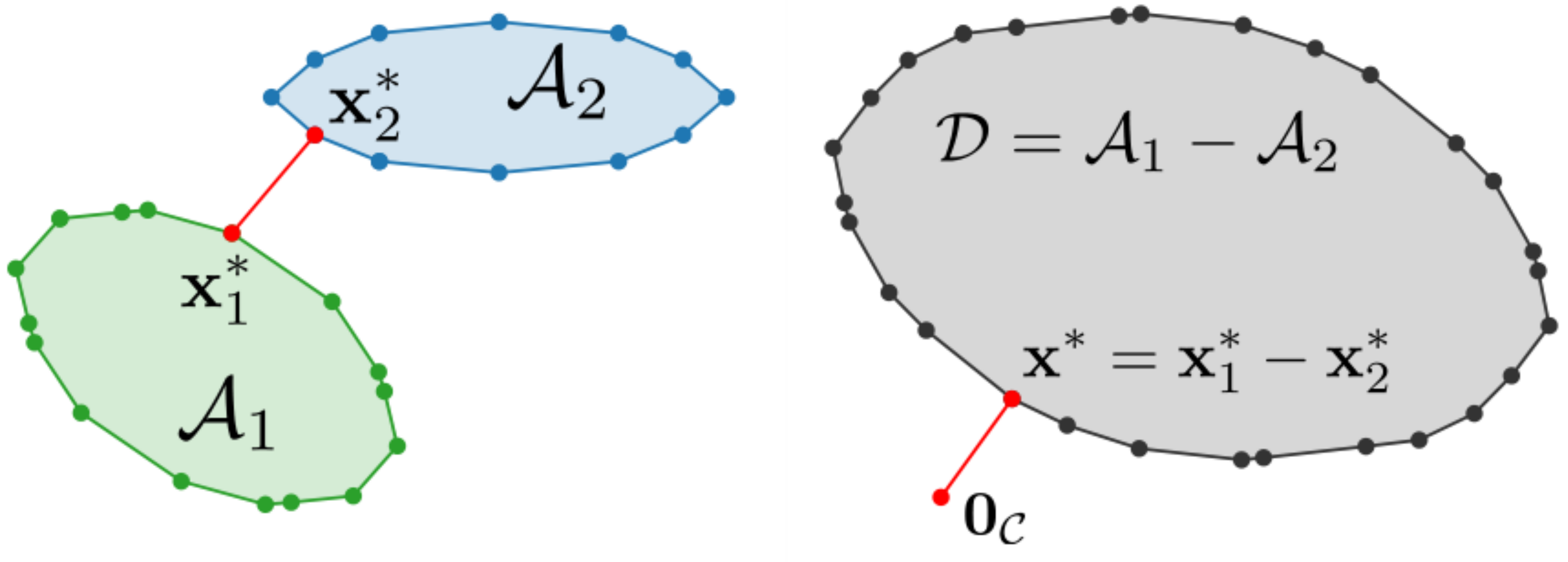
## Step 2 - How to formulate a collision detection problem

## Step 3 - Solving a collision detection problem with Frank-Wolfe

# 3 - Recasting the collision problem to a MNP

**The Minkowski difference:**

$$\mathscr{D} = \mathscr{A}_1 - \mathscr{A}_2 = \{x = x_1 - x_2,\ x_1 \in \mathscr{A}_1\ x_2 \in \mathscr{A}_2\}$$

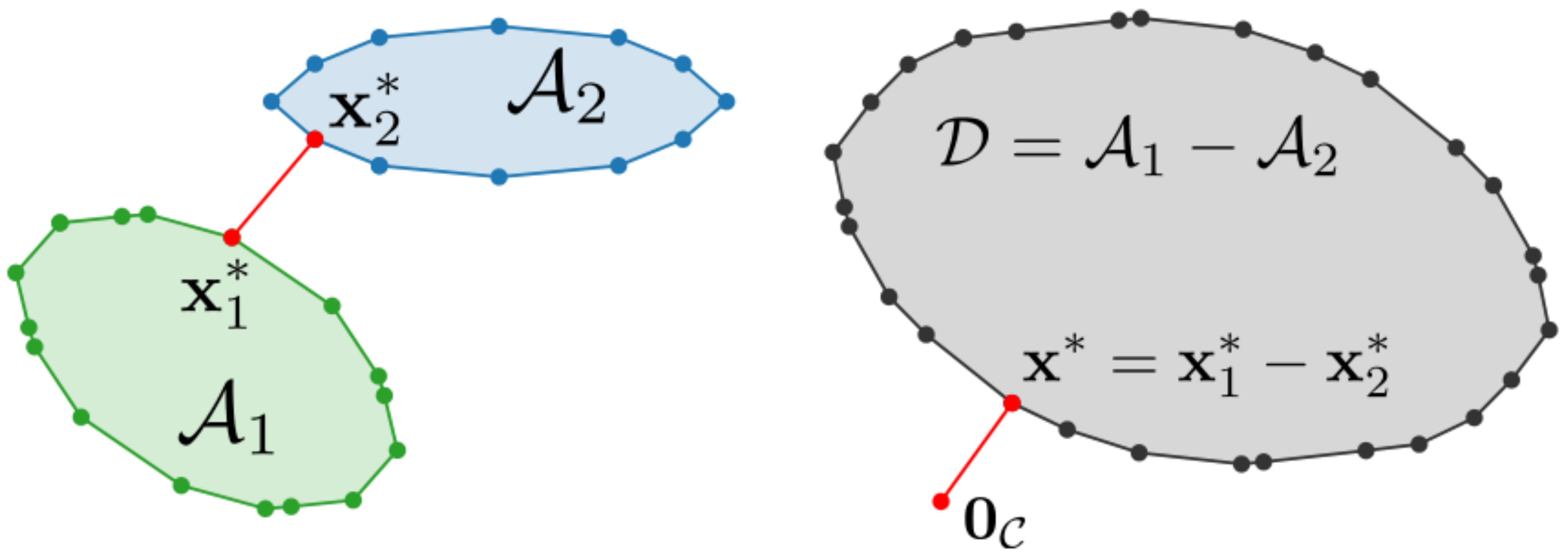



No collision

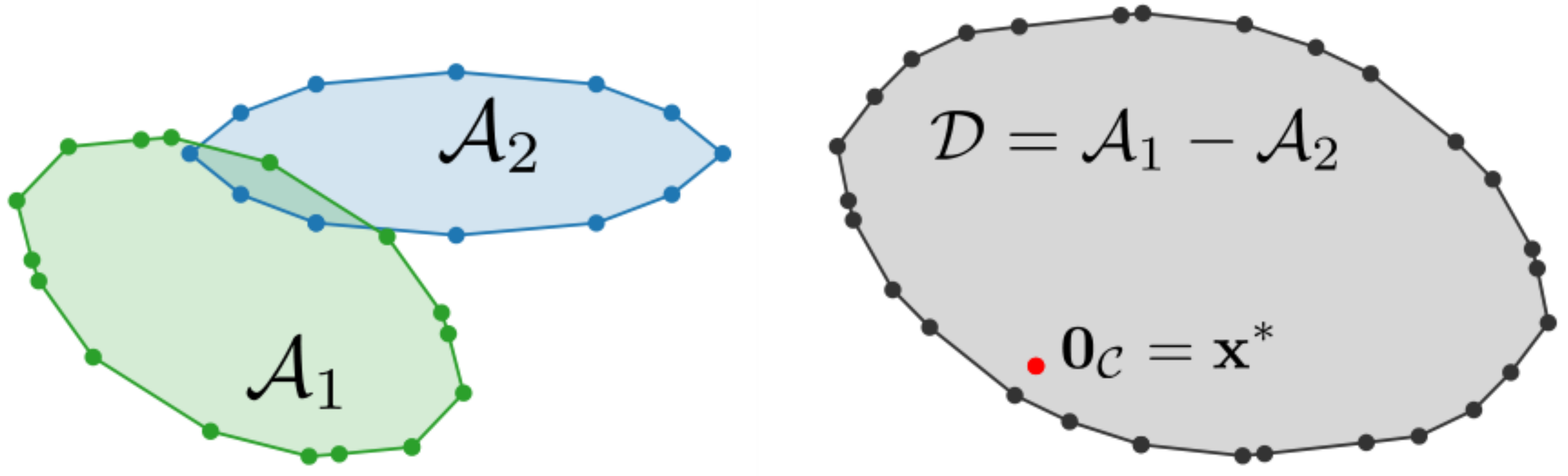
# 3 - Recasting the collision problem to a MNP

## The Minkowski difference:

$$\mathscr{D} = \mathscr{A}_1 - \mathscr{A}_2 = \{x = x_1 - x_2,\ x_1 \in \mathscr{A}_1\ x_2 \in \mathscr{A}_2\}$$



No collision

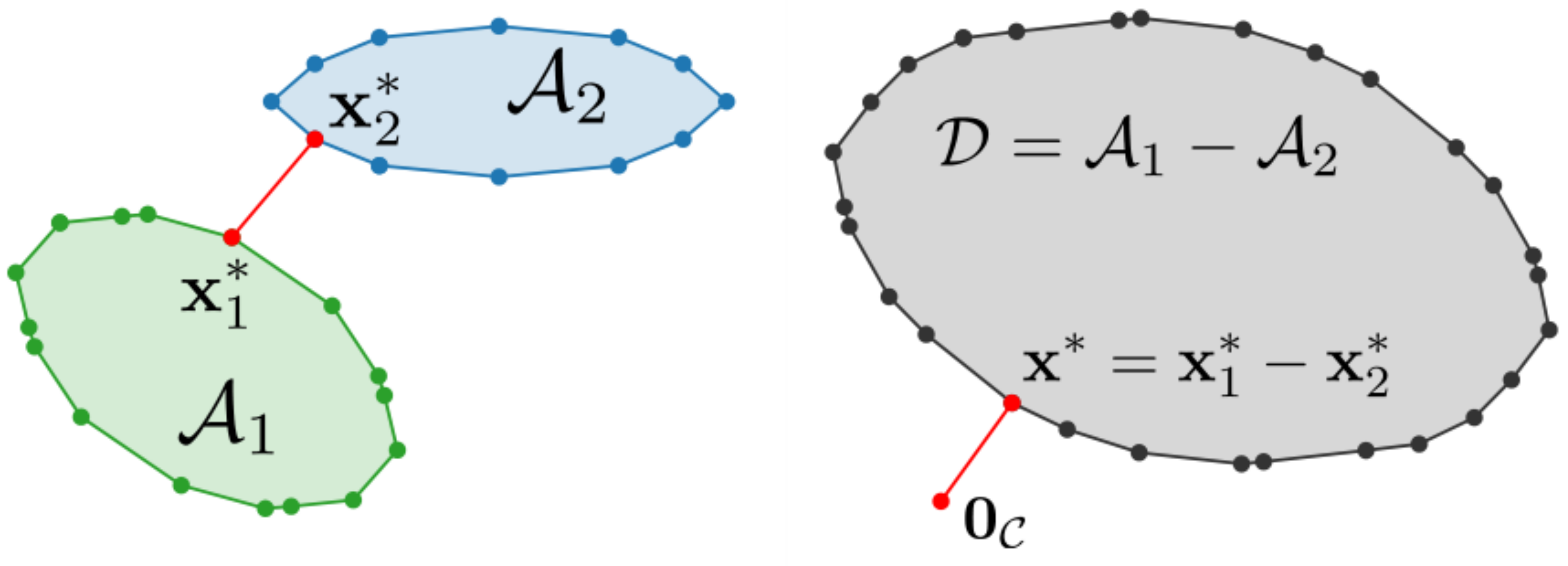


Collision

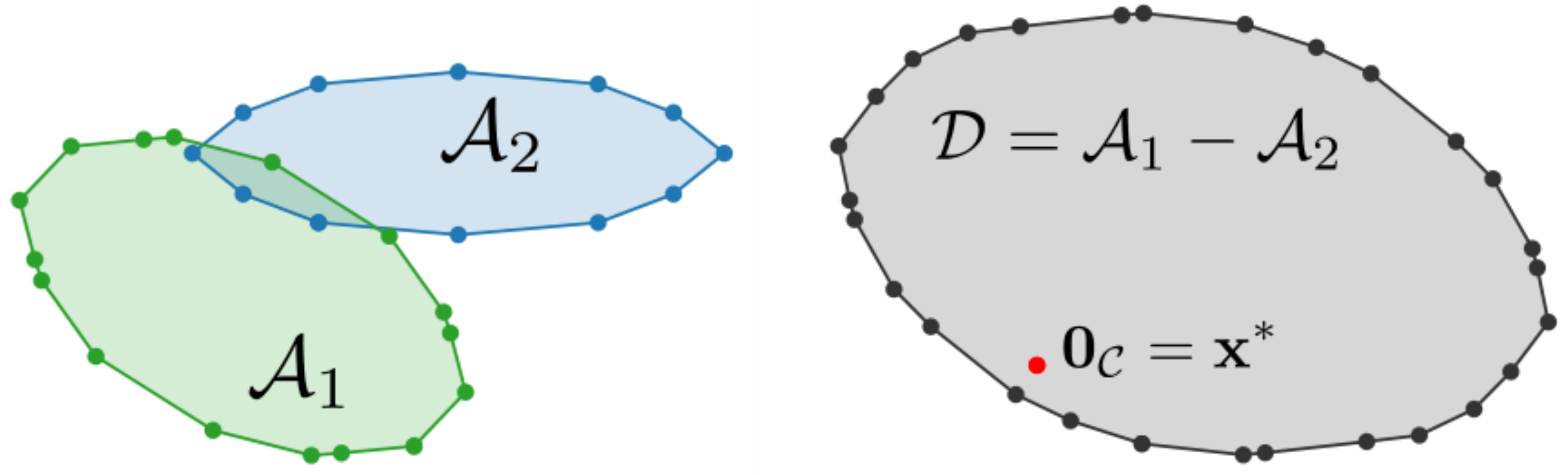
# 3 - Recasting the collision problem to a MNP

## The Minkowski difference:

$$\mathcal{D} = \mathcal{A}_1 - \mathcal{A}_2 = \{x = x_1 - x_2,\ x_1 \in \mathcal{A}_1\ x_2 \in \mathcal{A}_2\}$$



No collision



Collision

$$\min_{x_1 \in \mathcal{A}_1, x_2 \in \mathcal{A}_2} \frac{1}{2} ||x_1 - x_2||^2 \longrightarrow \boxed{\min_{x \in \mathcal{D}} \frac{1}{2} ||x||^2}$$

**MNP**

# 3 - Recasting the collision problem to a MNP

The Minkowski difference:

$\mathcal{D} = \mathcal{A}_1 - \mathcal{A}_2 = \{x = x_1 - x_2,\ x_1 \in \mathcal{A}_1\ x_2 \in \mathcal{A}_2\}$

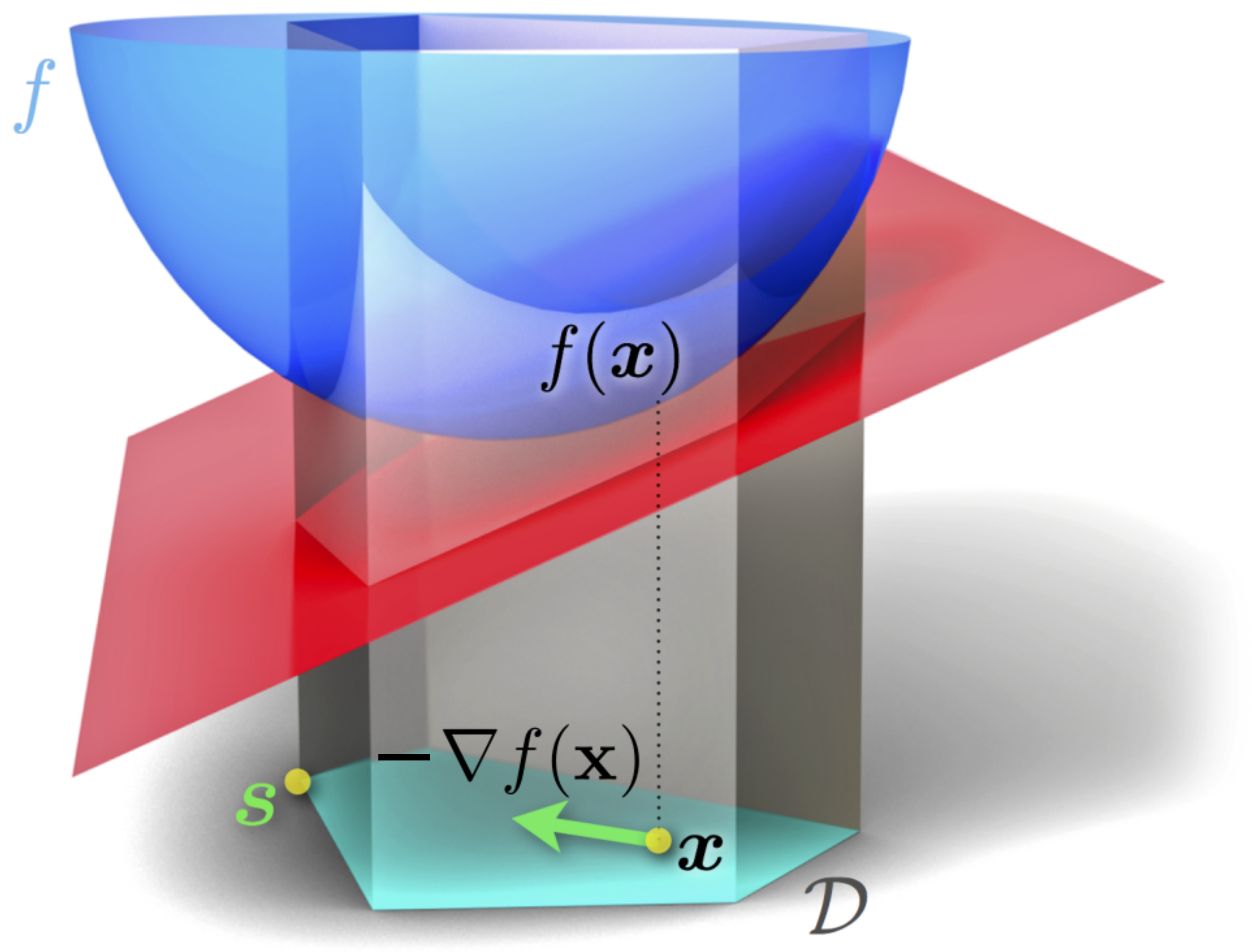
**Problem:** the Minkowski difference is intractable.
**Solution:** work implicitly with the Minkowski difference
**Algorithm:** Frank-Wolfe

$$\min_{x_1 \in \mathcal{A}_1, x_2 \in \mathcal{A}_2} \frac{1}{2} ||x_1 - x_2||^2 \longrightarrow \min_{x \in \mathcal{D}} \frac{1}{2} ||x||^2$$

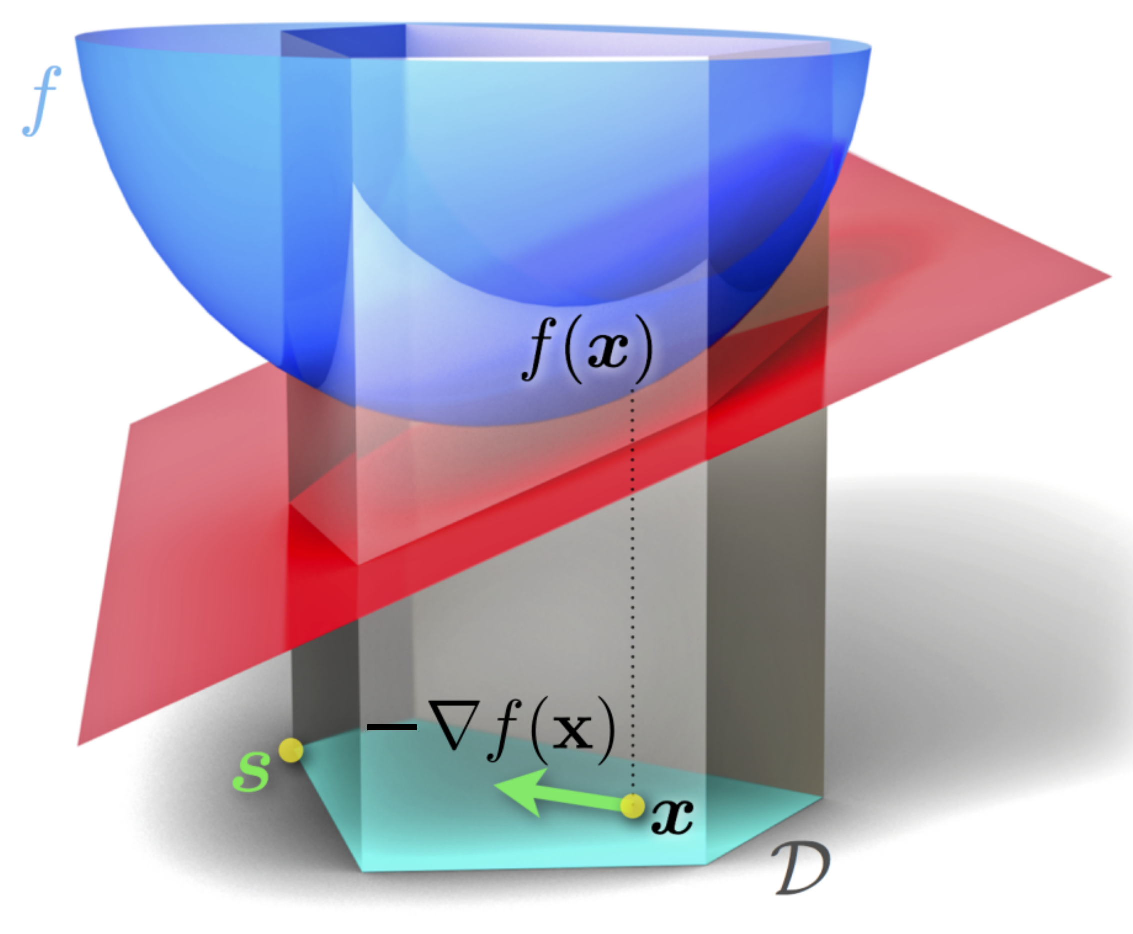
MNP

# 3 - The Frank-Wolfe algorithm

$\min_{x \in \mathscr{D}} f(x)$ $f$ **convex,** $\mathscr{D}$ **convex**



# 3 - The Frank-Wolfe algorithm

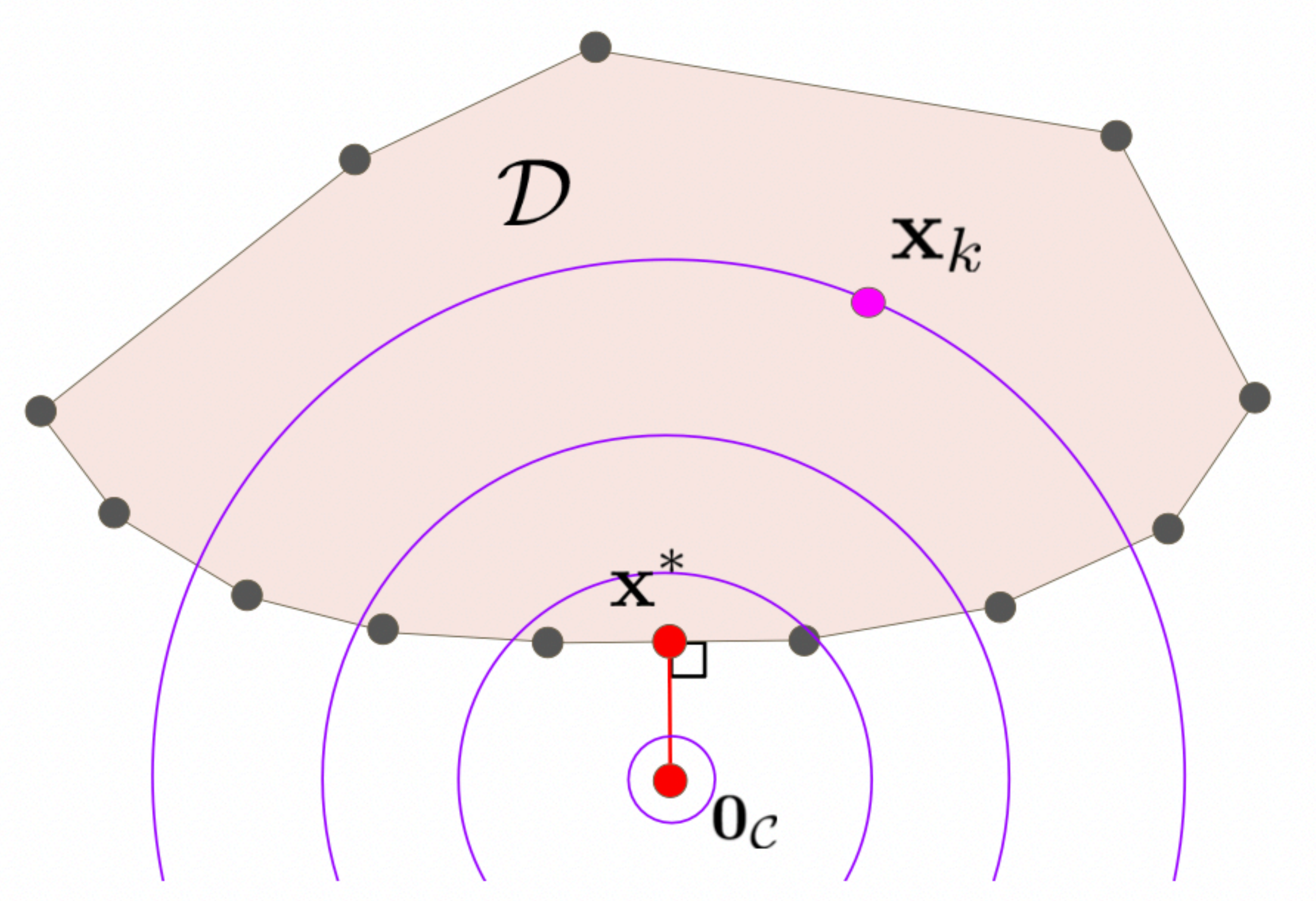



$$\min_{x \in \mathcal{D}} f(x)$$ $f$ convex, $\mathcal{D}$ convex

**Collision detection:**

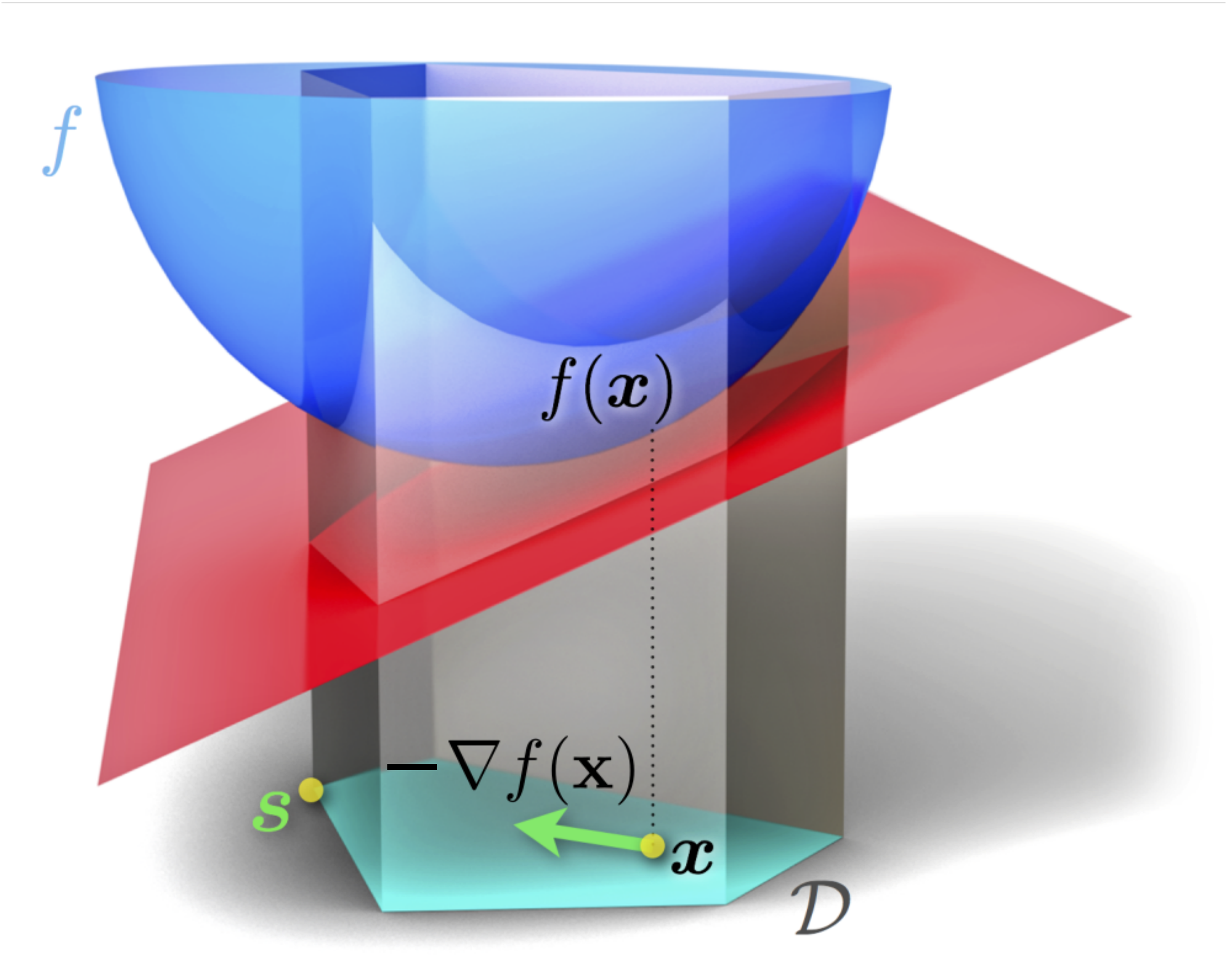
$$f(x) = \frac{1}{2} ||x||^2$$

$\mathcal{D}$ Minkowski difference of two shapes

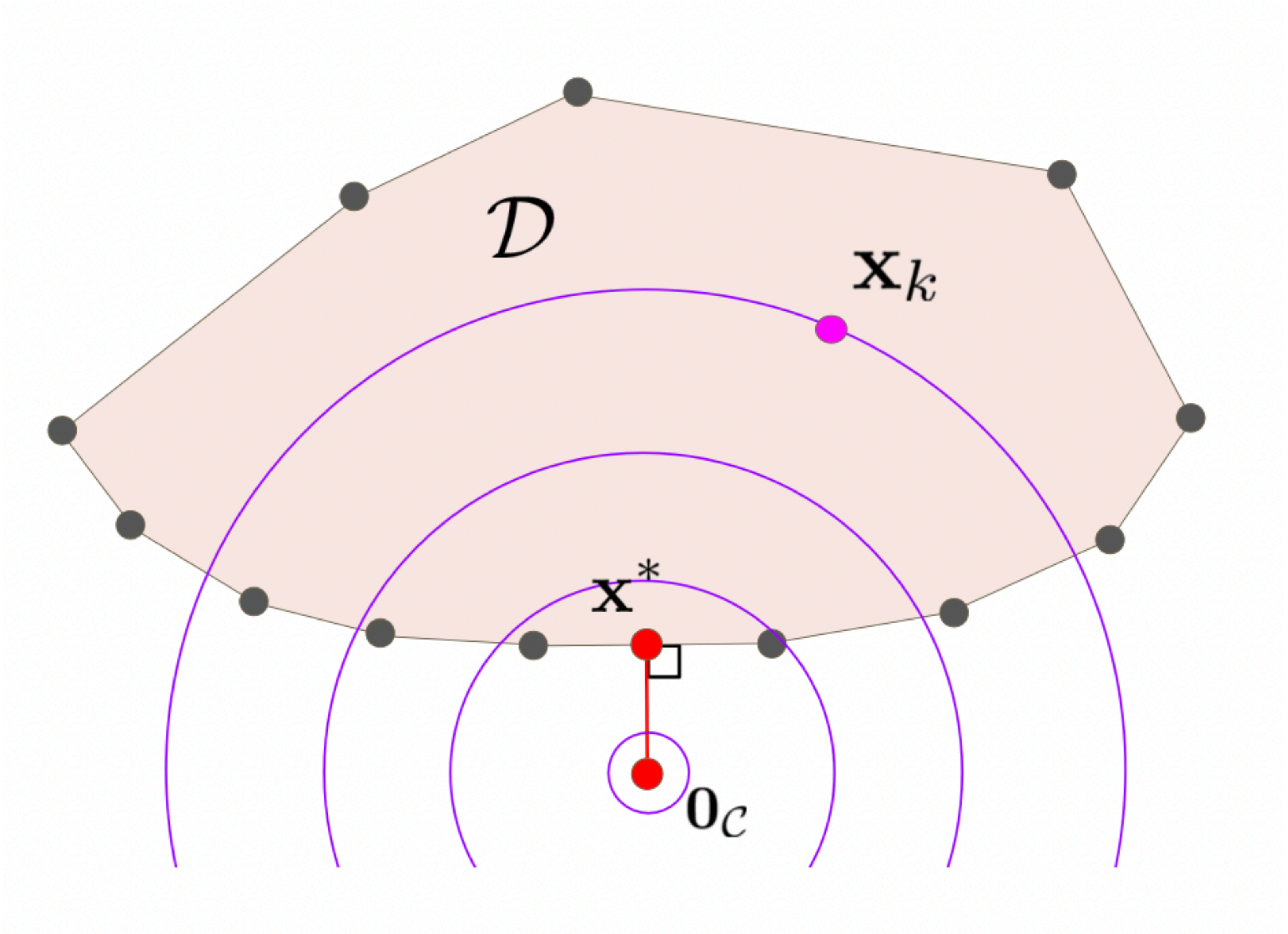


# 3 - The Frank-Wolfe algorithm

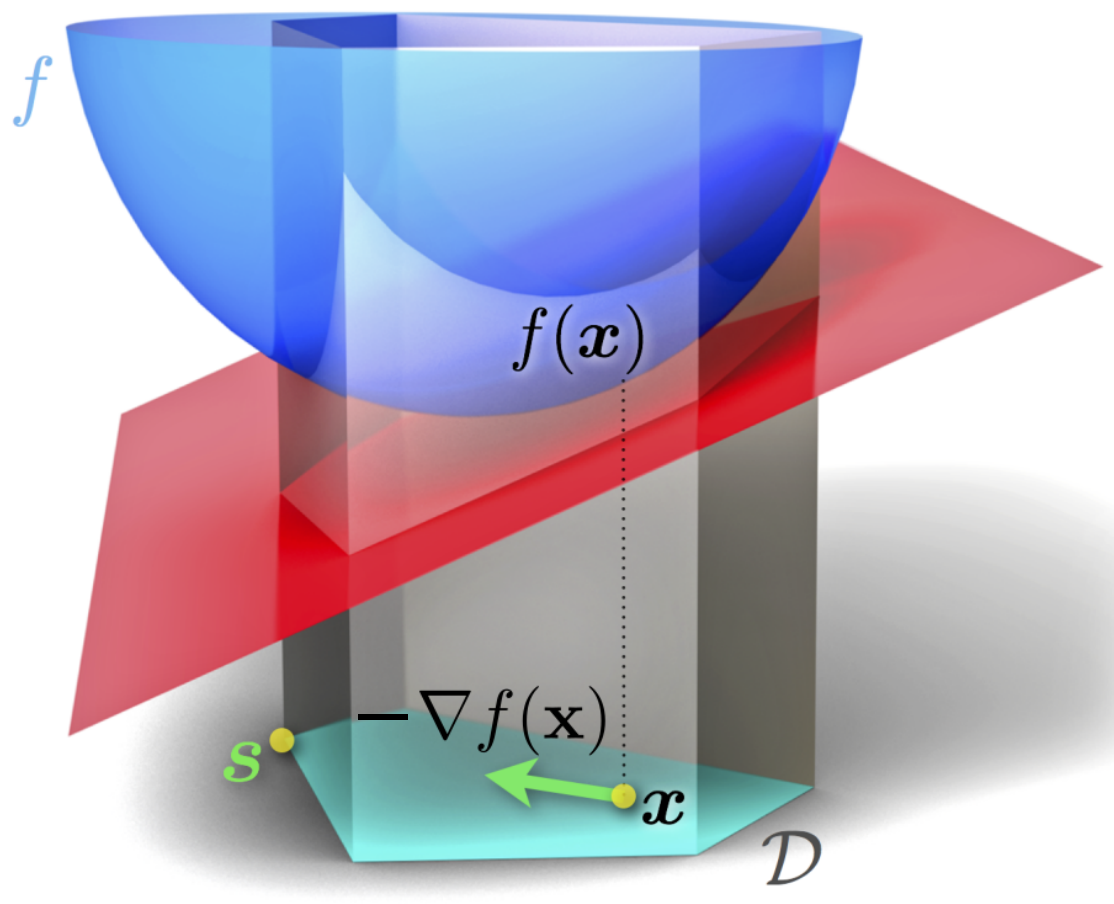
$\min_{x \in \mathscr{D}} f(x)$ $f$ **convex,** $\mathscr{D}$ **convex**




**Frank-Wolfe = "constrained gradient descent"**



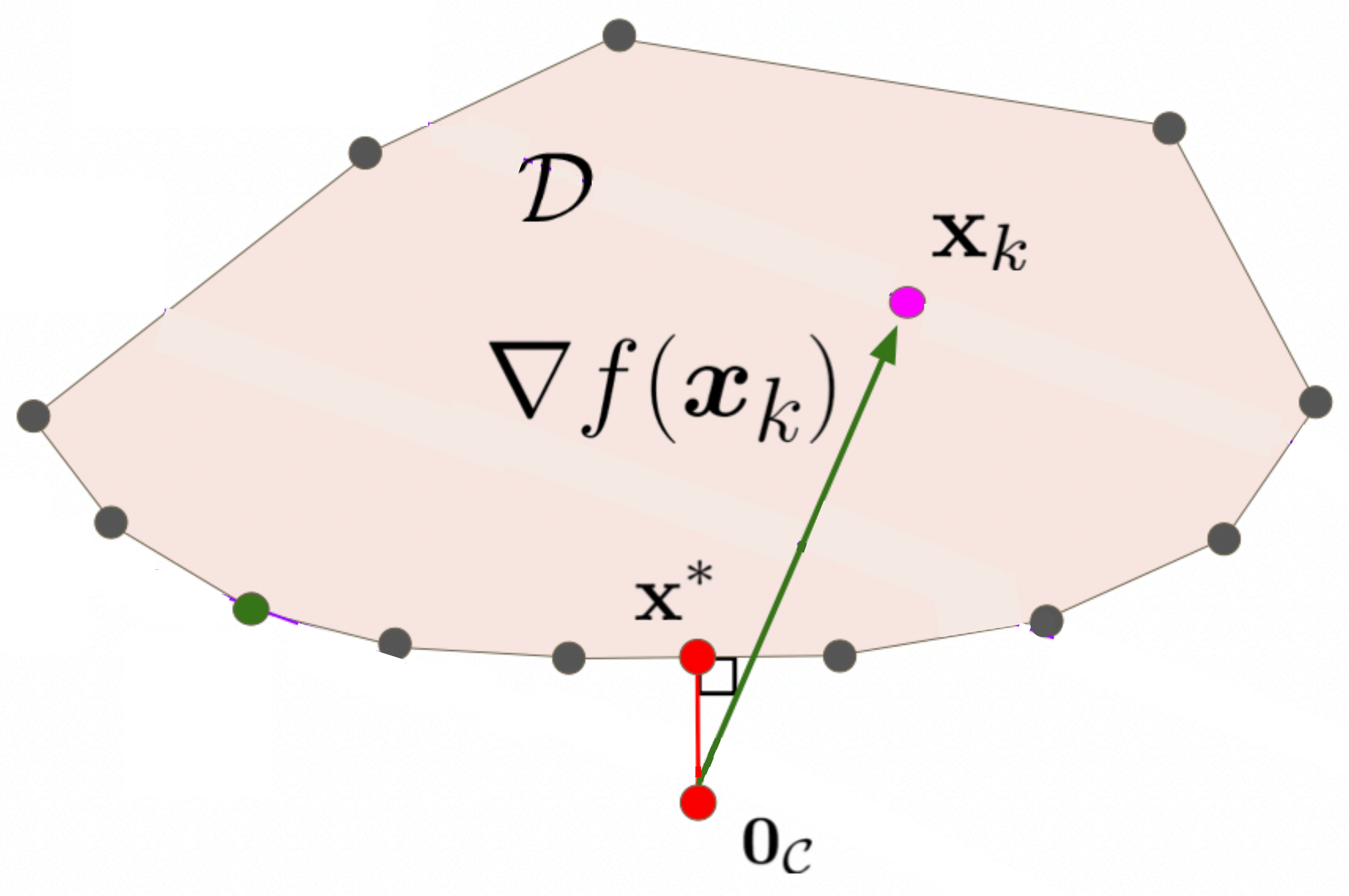
# 3 - The Frank-Wolfe algorithm



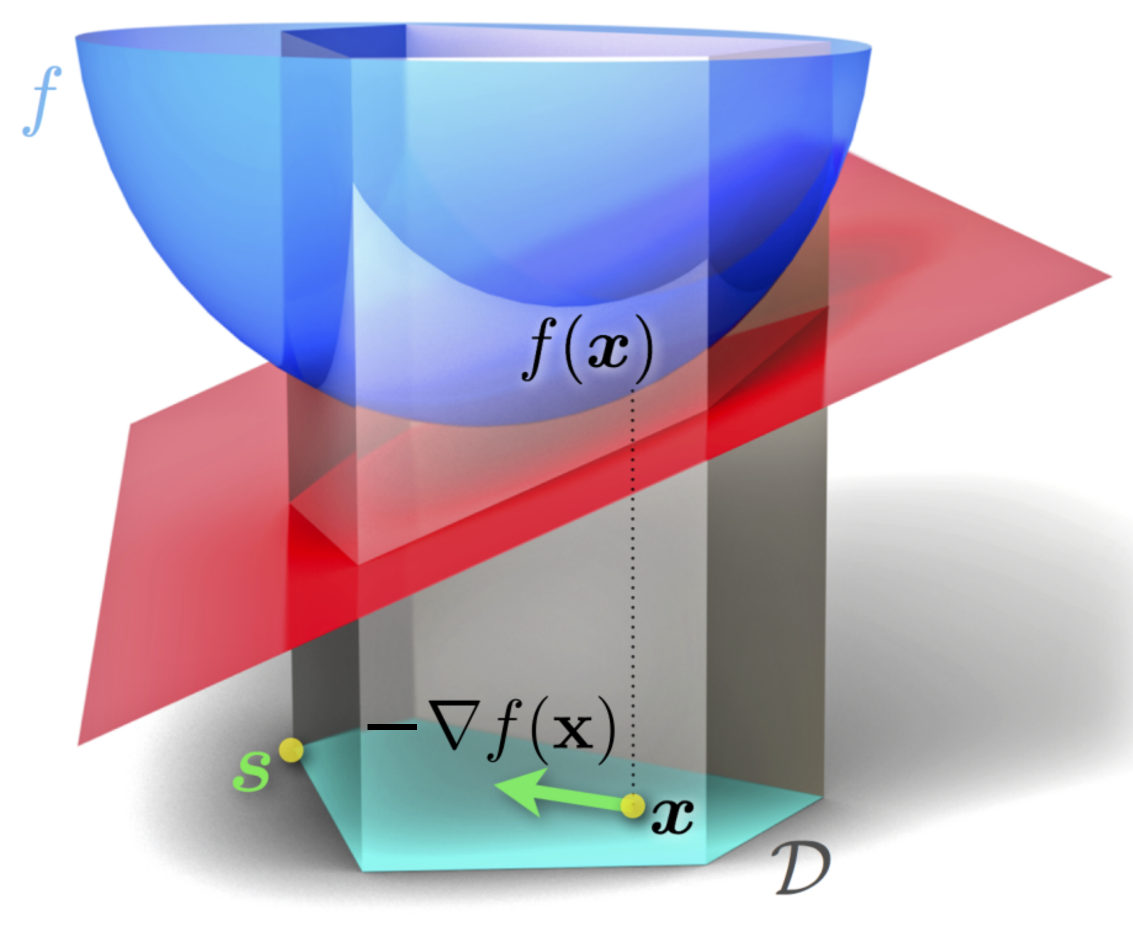

$$\min_{x \in \mathscr{D}} f(x) \quad f \text{ convex, } \mathscr{D} \text{ convex}$$

**Frank-Wolfe = "constrained gradient descent":**

<u>Step 1:</u> Compute gradient $\nabla f(x_k)$ at current iterate $x_k$



# 3 - The Frank-Wolfe algorithm



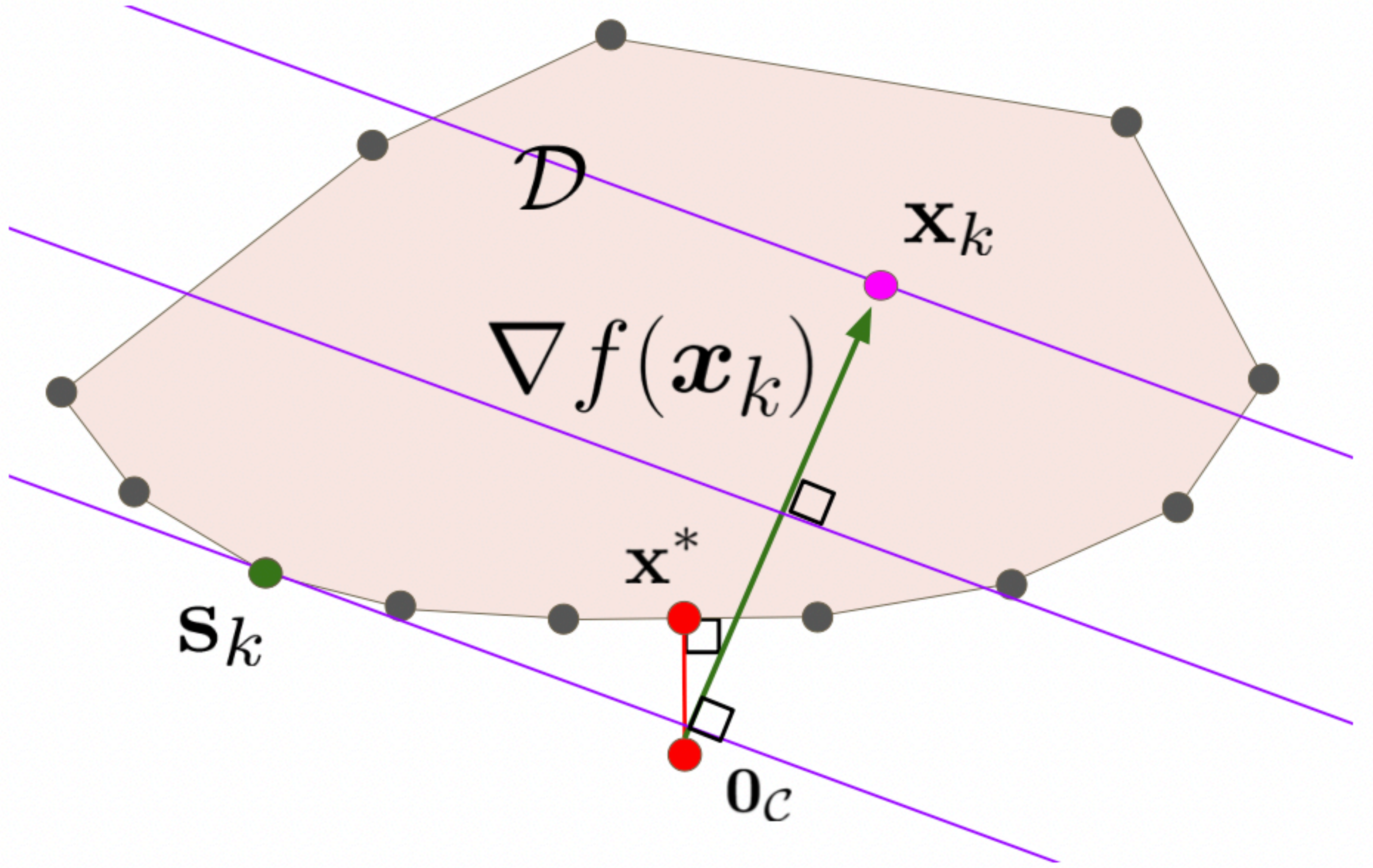

$$\min_{x \in \mathscr{D}} f(x)$$ $f$ **convex,** $\mathscr{D}$ **convex**

**Frank-Wolfe = "constrained gradient descent":**

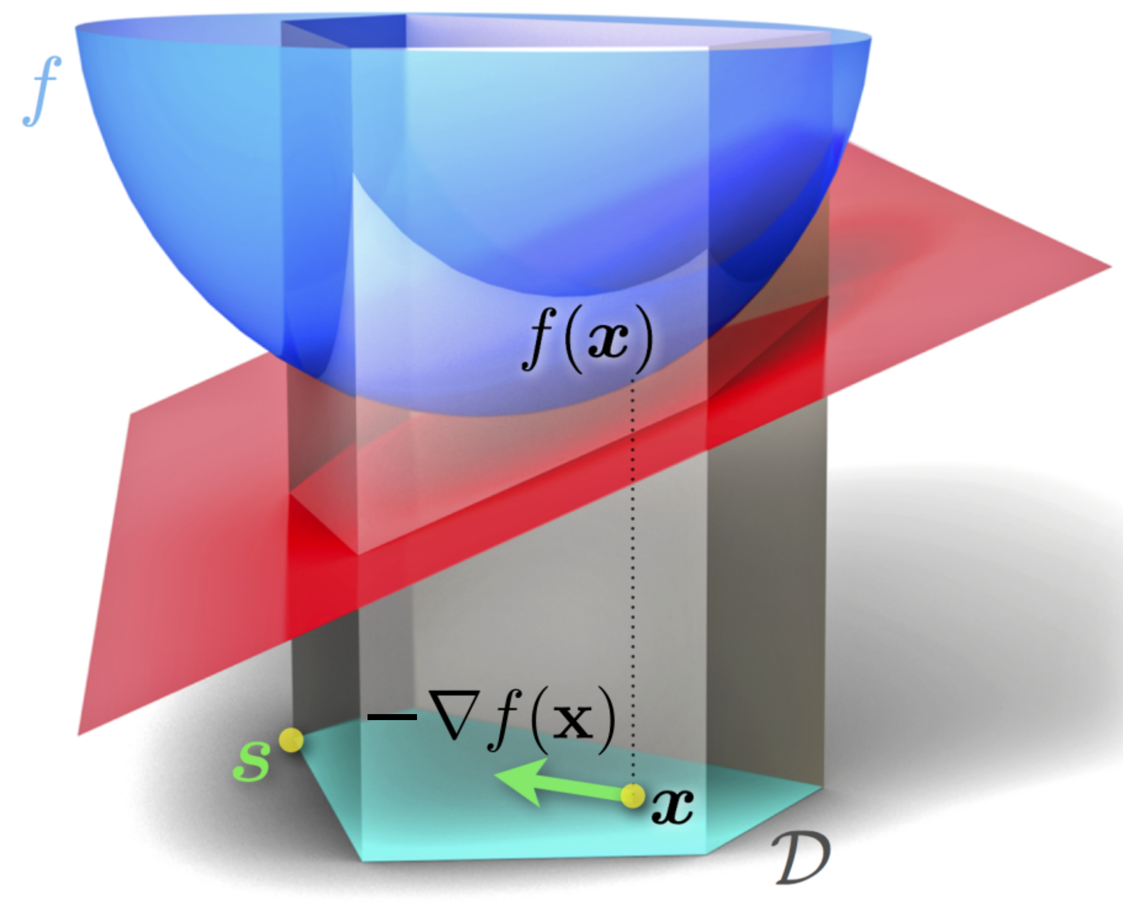
**Step 1: Compute gradient $\nabla f(x_k)$ at current iterate $x_k$**

**Step 2: Compute point $s_k \in \mathscr{D}$ "most" in direction $-\nabla f(x_k)$**

**-> support point** $s_k = \text{argmin}_{y \in \mathscr{D}} < y, \nabla f(x_k) >$



# 3 - The Frank-Wolfe algorithm




$\min_{x \in \mathscr{D}} f(x)$ $\quad f$ convex, $\mathscr{D}$ convex

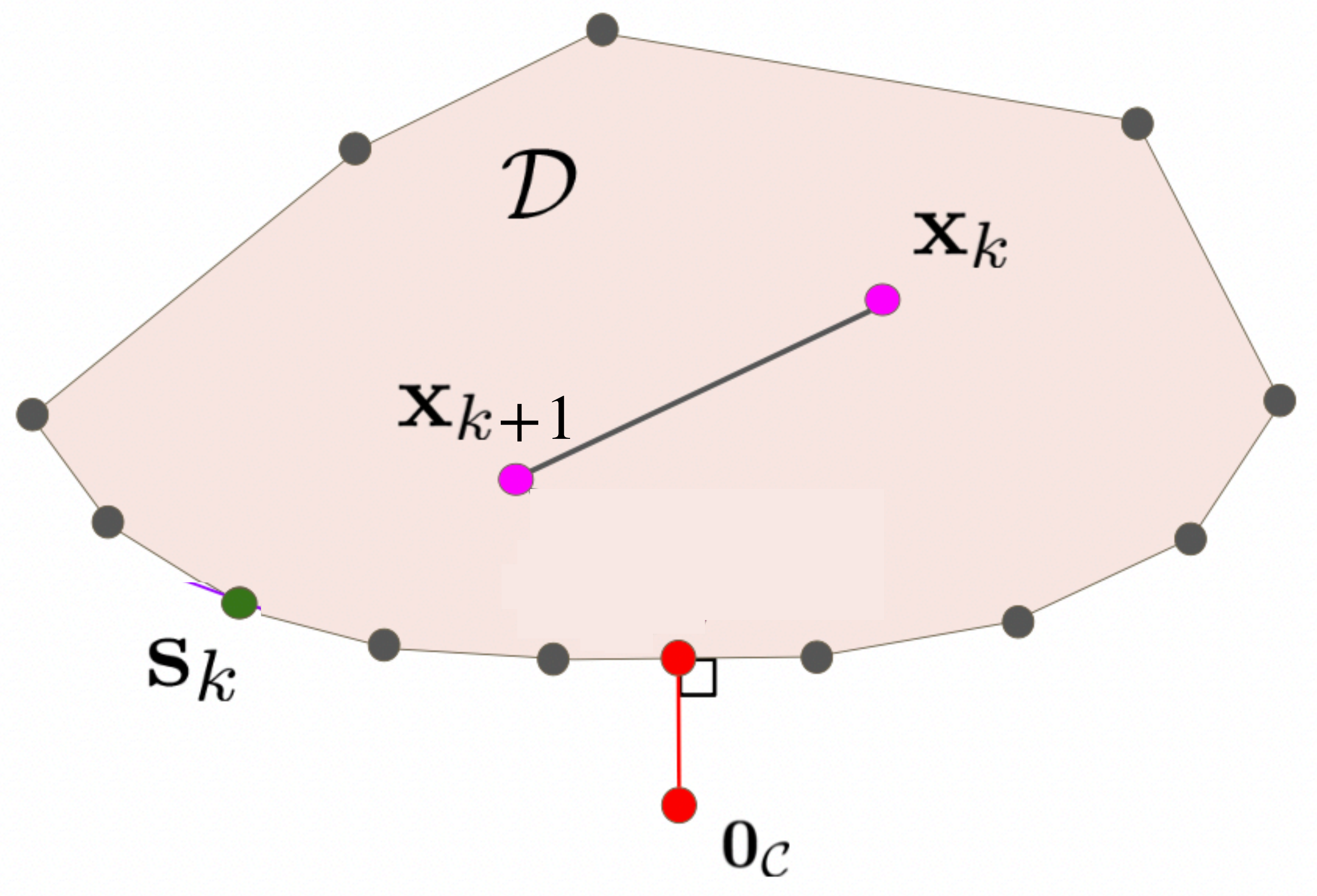
Frank-Wolfe = "constrained gradient descent":

Step 1: Compute gradient $\nabla f(x_k)$ at current iterate $x_k$

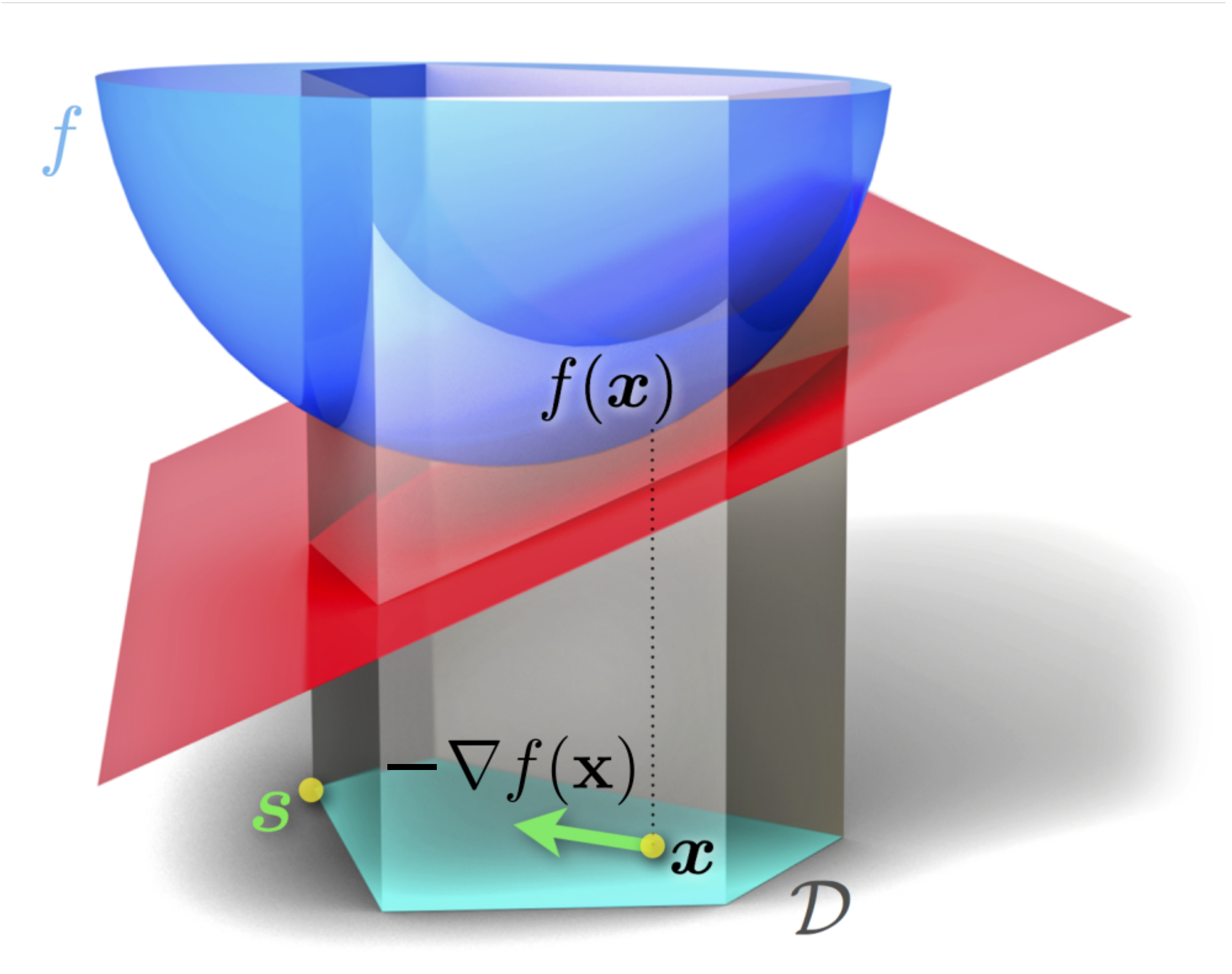
Step 2: Compute point $s_k \in \mathscr{D}$ "most" in direction $-\nabla f(x_k)$

-> support point $s_k = \text{argmin}_{y \in \mathscr{D}} < y, \nabla f(x_k) >$

Step 3: Move towards $s_k$ and repeat steps 1-3



# 3 - The Frank-Wolfe algorithm




$\min_{x \in \mathscr{D}} f(x)$ $\quad f$ convex, $\mathscr{D}$ convex

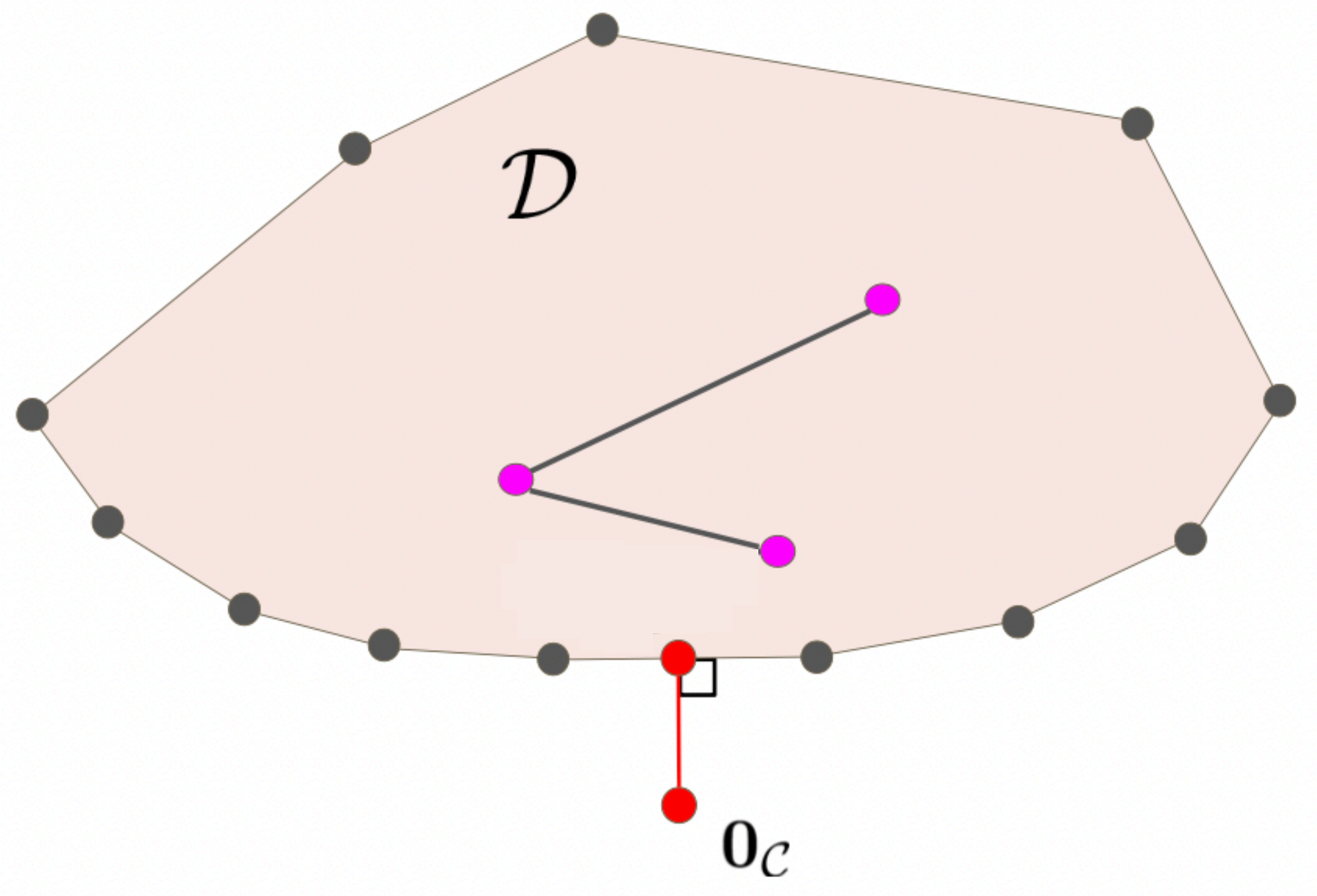
Frank-Wolfe = “constrained gradient descent”:

**Step 1:** Compute gradient $\nabla f(x_k)$ at current iterate $x_k$

**Step 2:** Compute point $s_k \in \mathscr{D}$ “most” in direction $-\nabla f(x_k)$

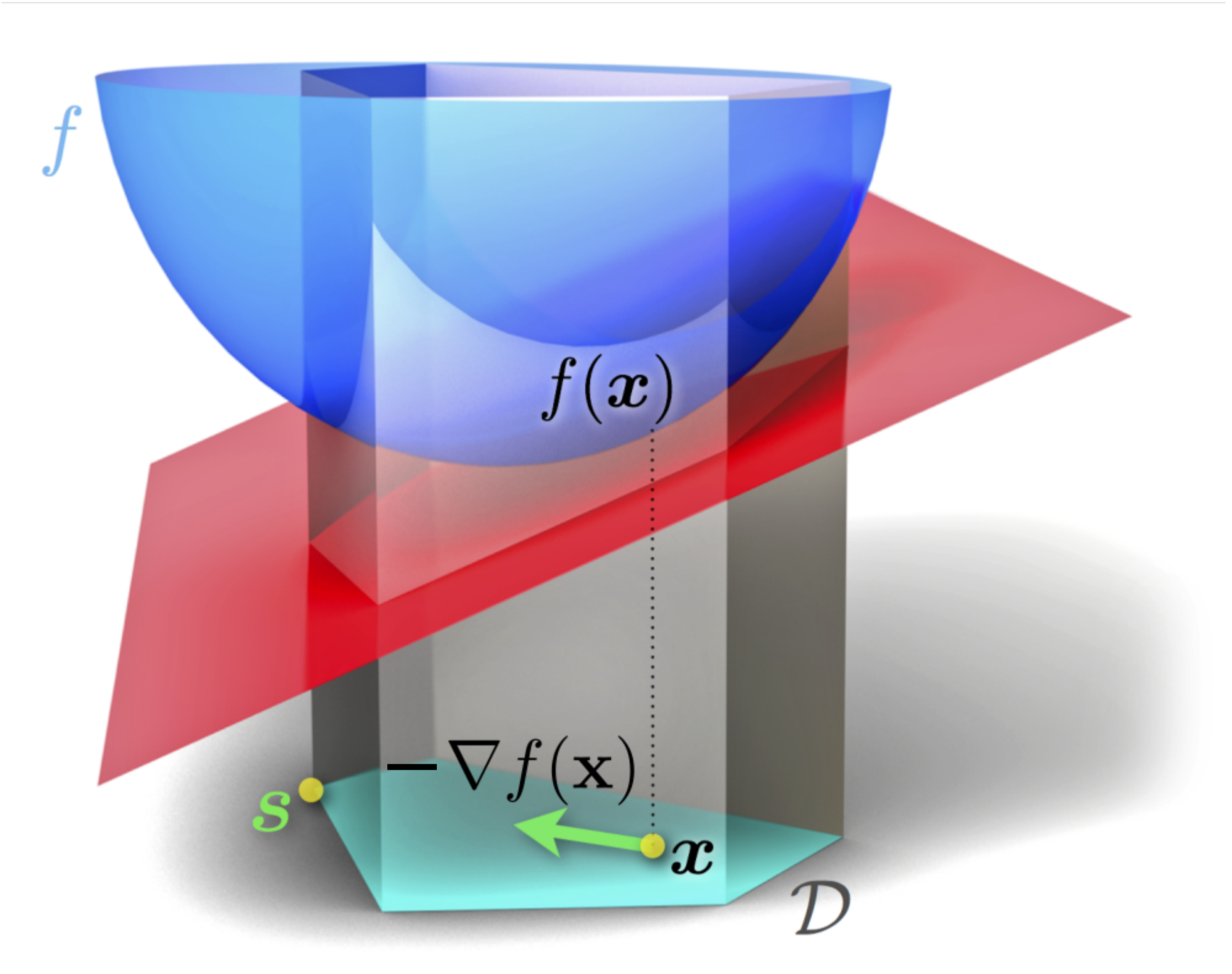
-> support point $s_k = \text{argmin}_{y \in \mathscr{D}} < y, \nabla f(x_k) >$

**Step 3:** Move towards $s_k$ and repeat steps 1-3



# 3 - The Frank-Wolfe algorithm

$\min_{x \in \mathcal{D}} f(x)$ $f$ convex, $\mathcal{D}$ convex



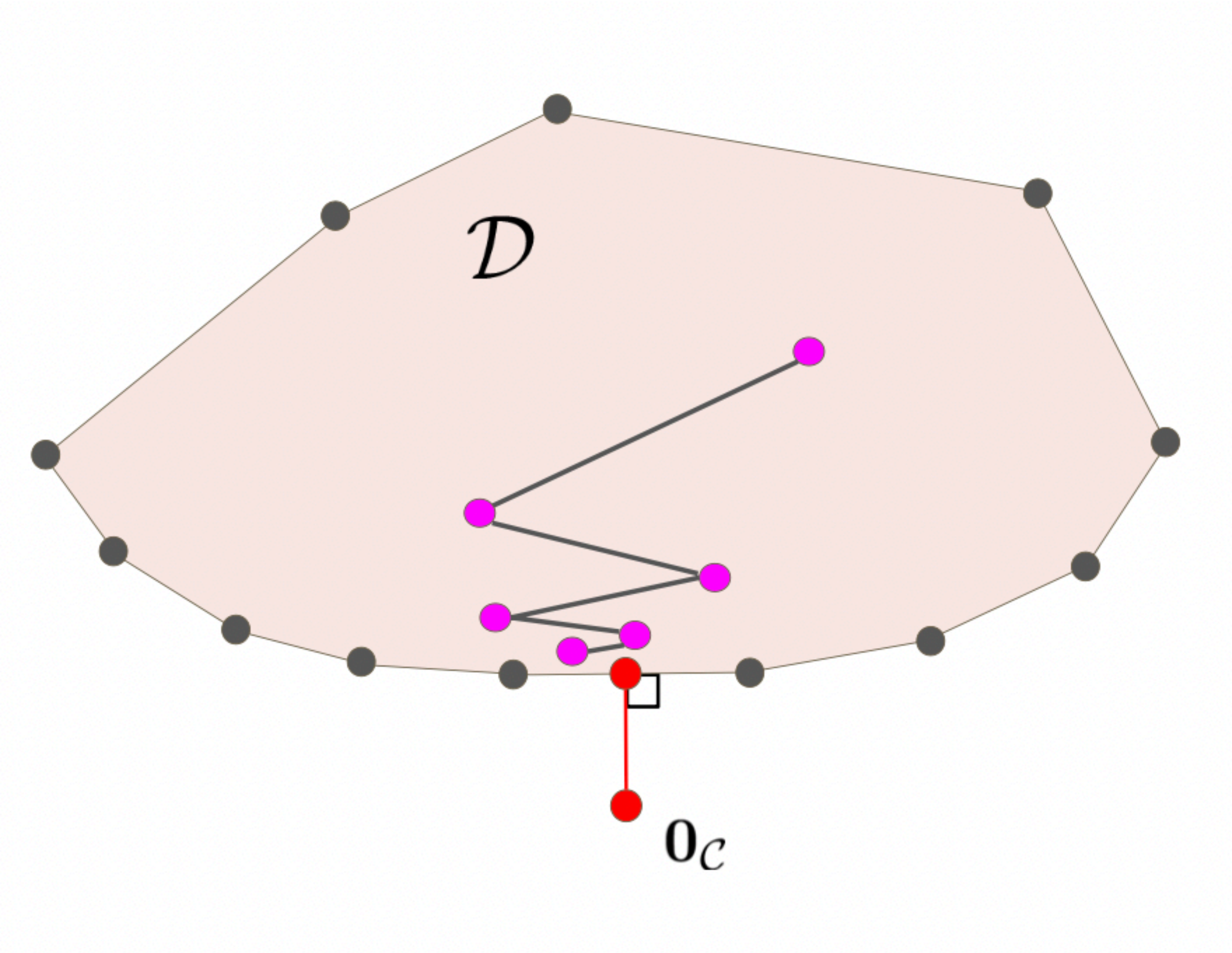

Frank-Wolfe = “constrained gradient descent”:

Step 1: Compute gradient $\nabla f(x_k)$ at current iterate $x_k$

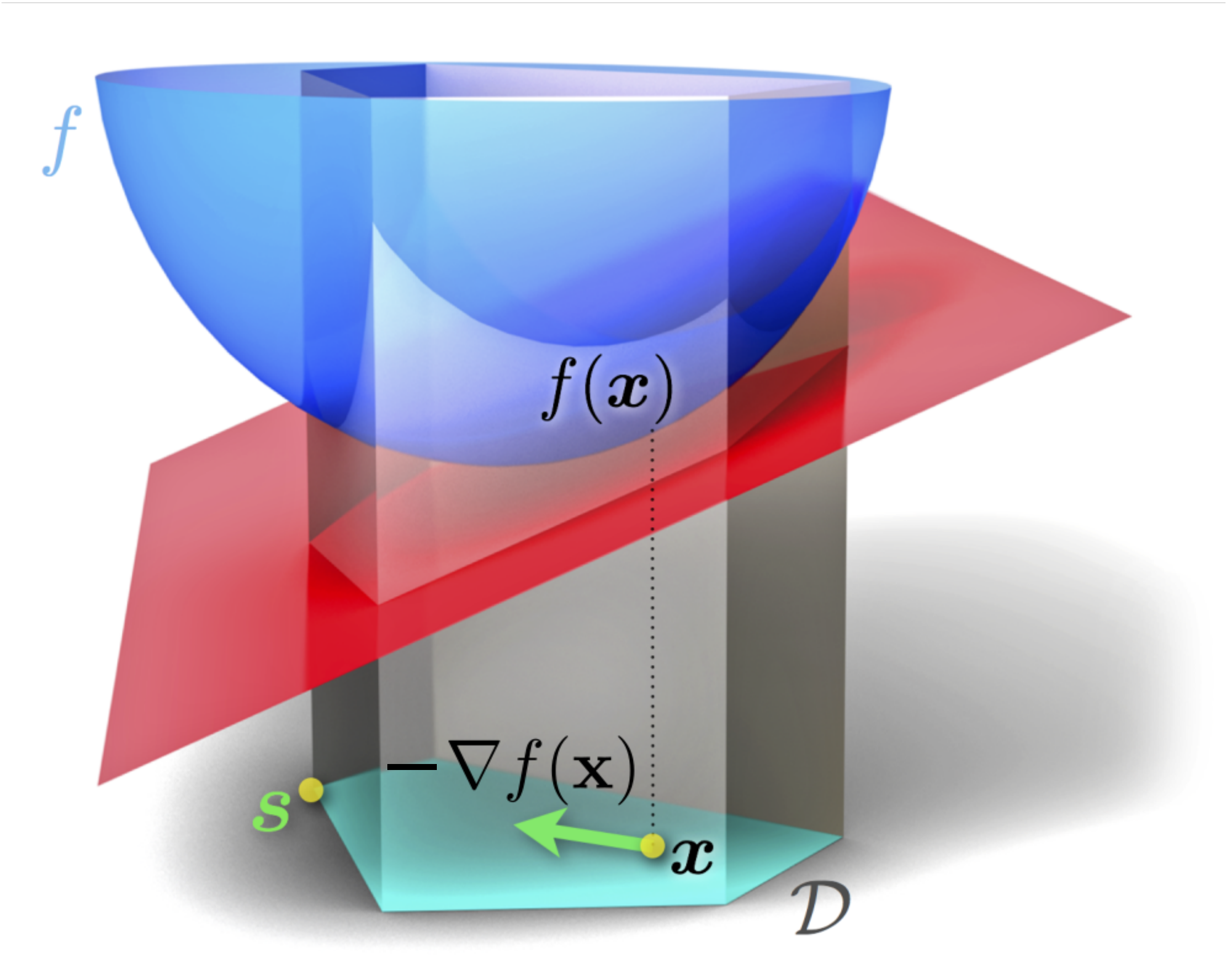
Step 2: Compute point $s_k \in \mathcal{D}$ “most” in direction $-\nabla f(x_k)$

-> support point $s_k = \text{argmin}_{y \in \mathcal{D}} < y, \nabla f(x_k) >$

Step 3: Move towards $s_k$ and repeat steps 1-3



# 3 - The Frank-Wolfe algorithm




$$\min_{x \in \mathcal{D}} f(x)$$ $f$ convex, $\mathcal{D}$ convex

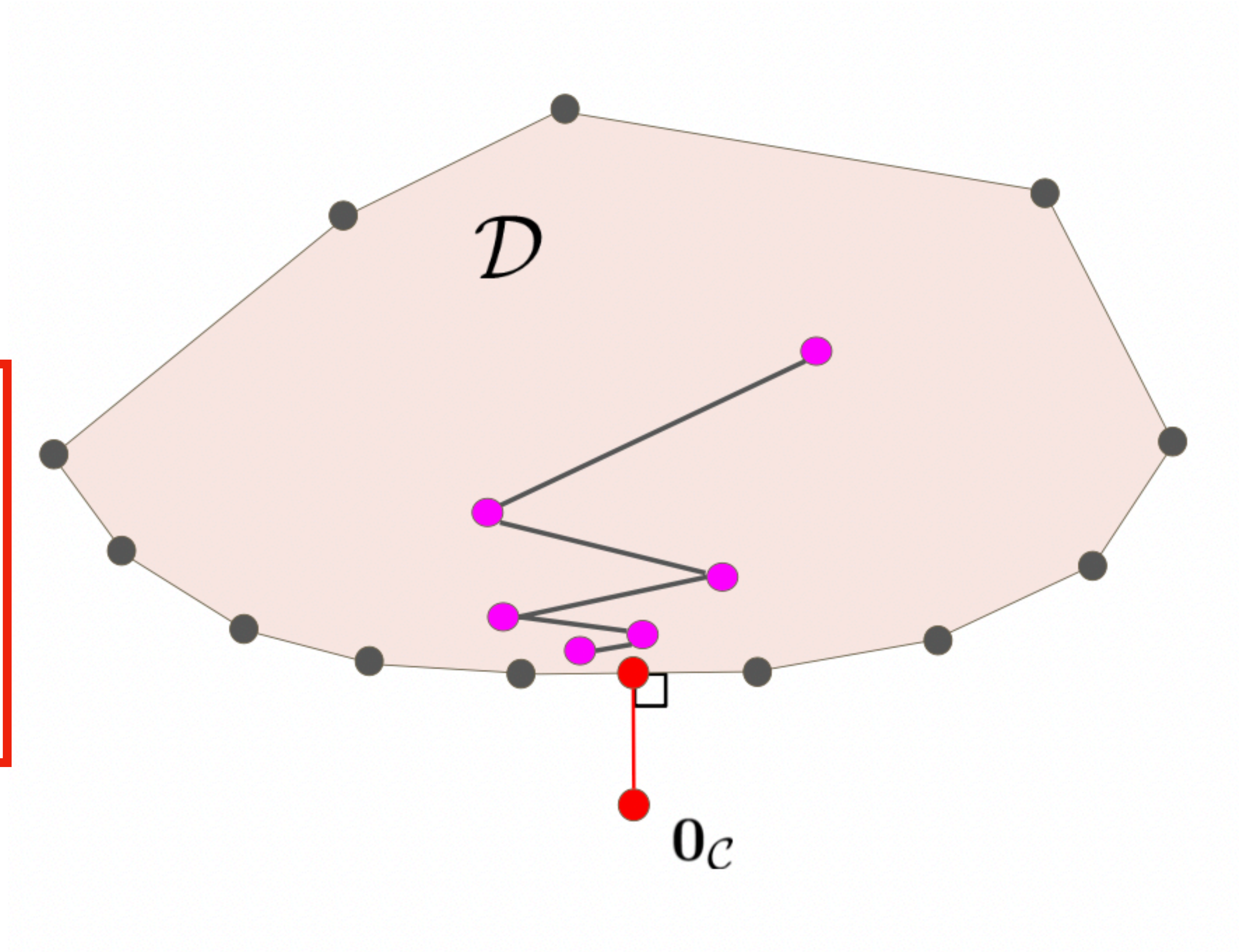
**Frank-Wolfe = "constrained gradient descent":**

**Step 1:** Compute gradient $\nabla f(x_k)$ at current iterate $x_k$

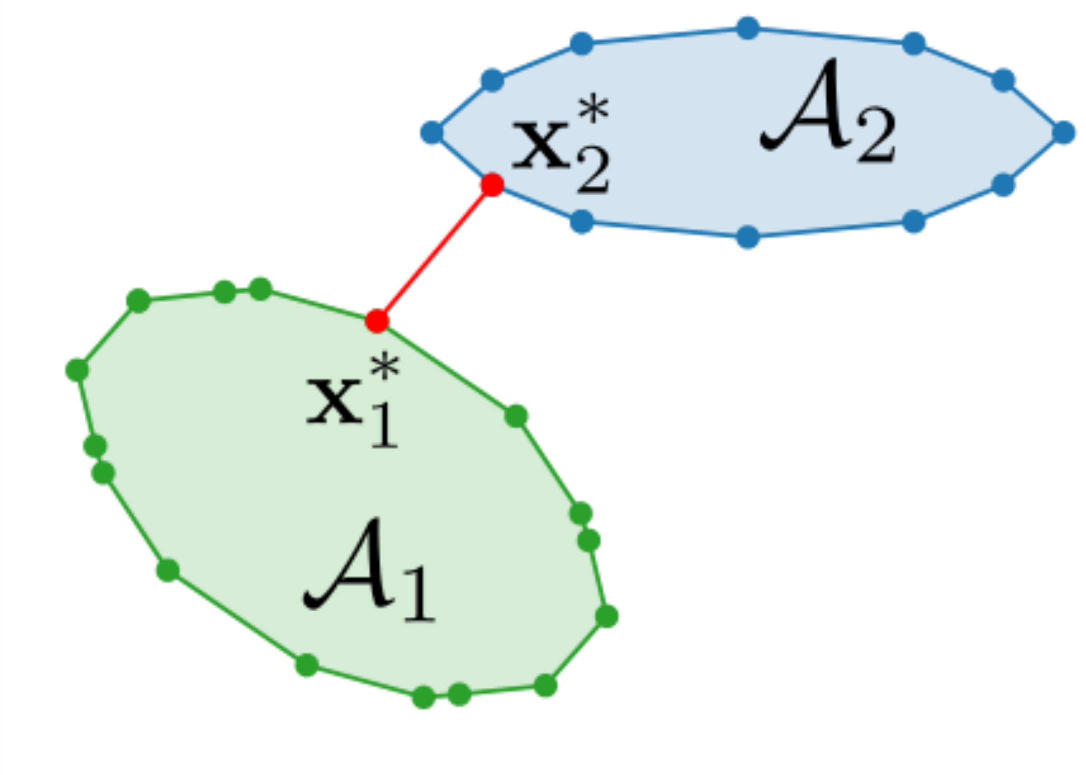
**Step 2:** Compute point $s_k \in \mathcal{D}$ "most" in direction $-\nabla f(x_k)$

-> support point $s_k = \text{argmin}_{y \in \mathcal{D}} < y, \nabla f(x_k) >$

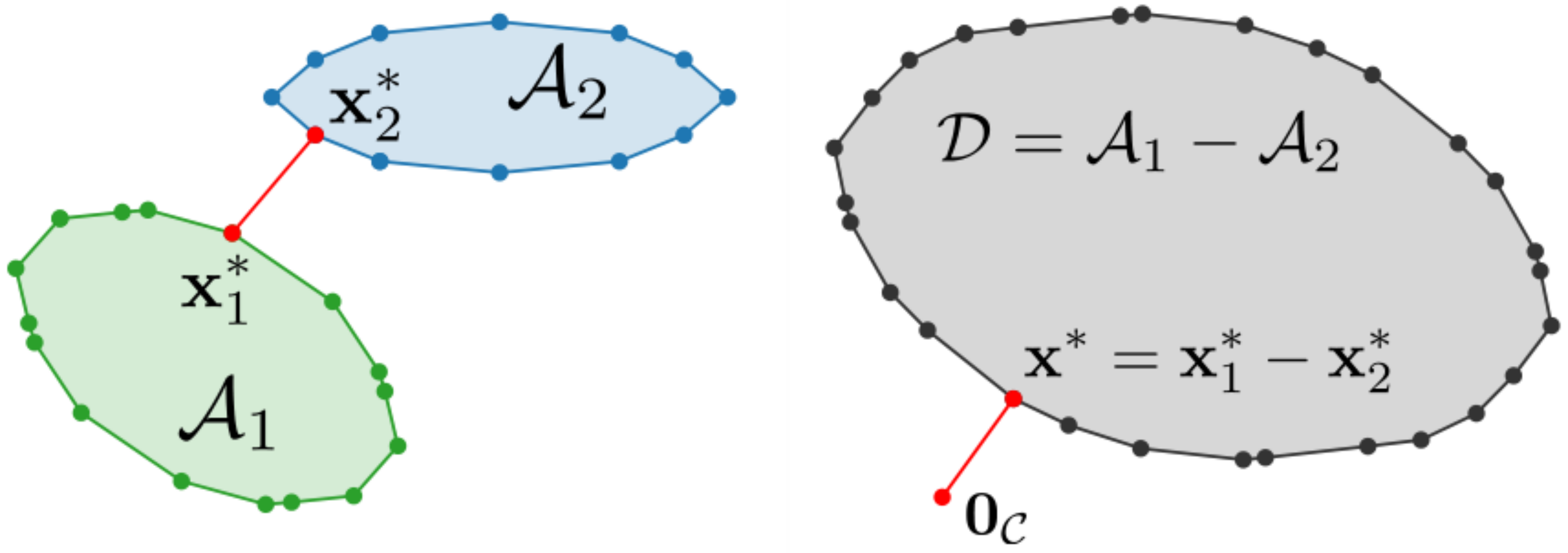
**Step 3:** Move towards $s_k$ and repeat steps 1-3 **?**



# 3 - Recap of collision detection with Frank-Wolfe



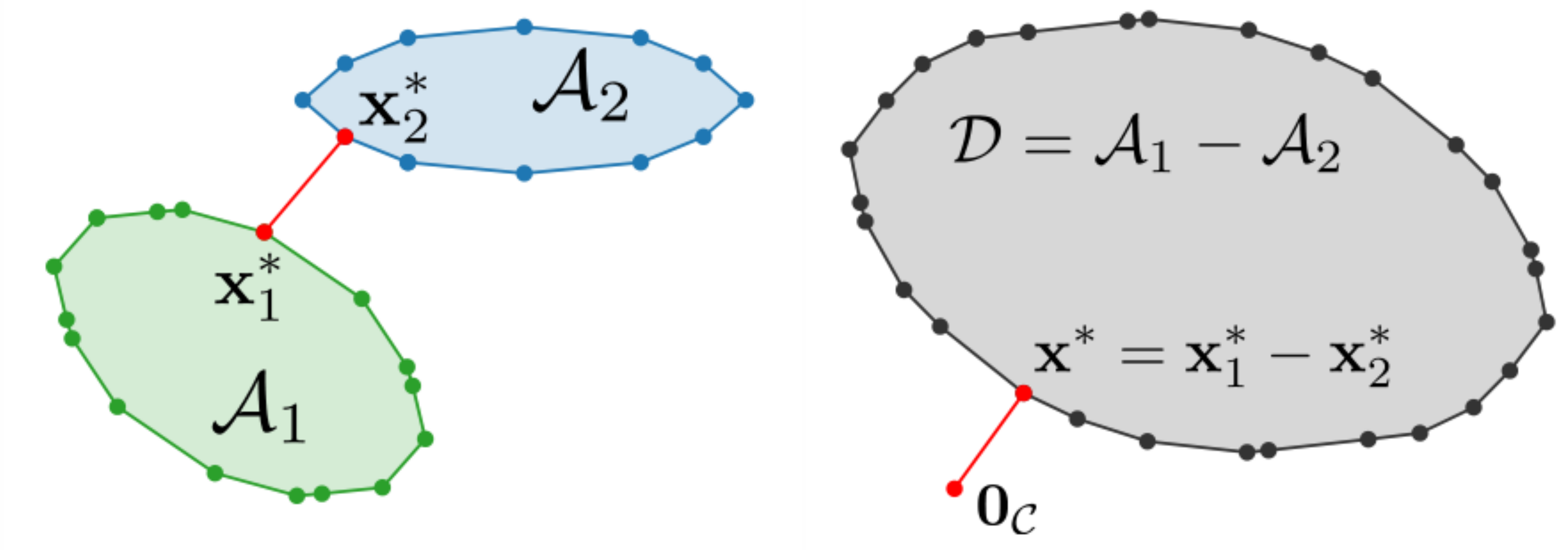
# 3 - Recap of collision detection with Frank-Wolfe




$$\min_{x_1 \in \mathcal{A}_1, x_2 \in \mathcal{A}_2} \frac{1}{2} || x_1 - x_2 ||^2 \longrightarrow \min_{x \in \mathcal{D}} \frac{1}{2} || x ||^2$$

**MNP**

# 3 - Recap of collision detection with Frank-Wolfe




$$\min_{x_1 \in \mathscr{A}_1, x_2 \in \mathscr{A}_2} \frac{1}{2} ||x_1 - x_2||^2 \longrightarrow \boxed{\min_{x \in \mathscr{D}} \frac{1}{2} ||x||^2}$$

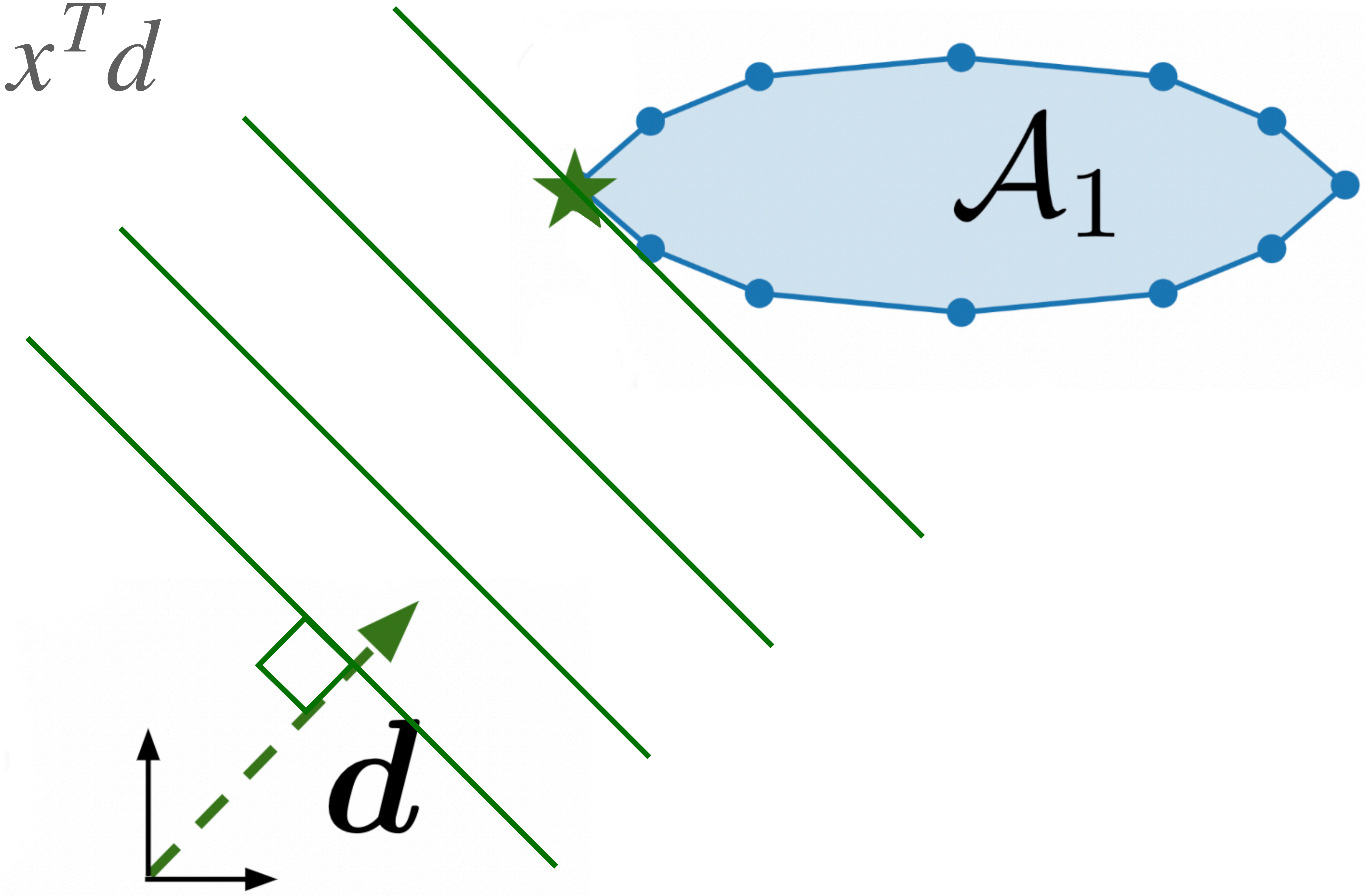
**MNP**

**Frank-Wolfe = "constrained gradient descent", needs to compute support points:**

$$\boxed{s = \text{argmin}_{y \in \mathscr{D}} < y, \nabla f(x) >}$$

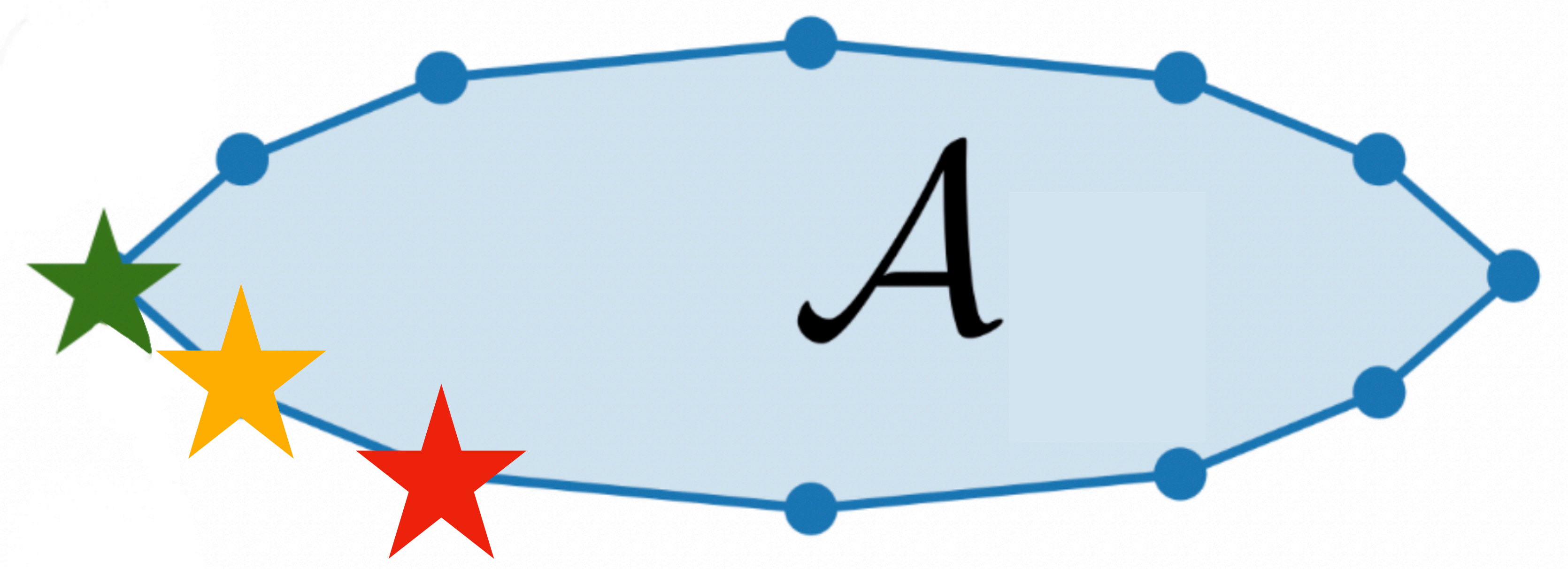
# 3 - Computing support points on shapes

$$S_{\mathscr{A}}(d) = \underset{x \in \mathscr{A}}{\text{argmin}}\ x^T d$$

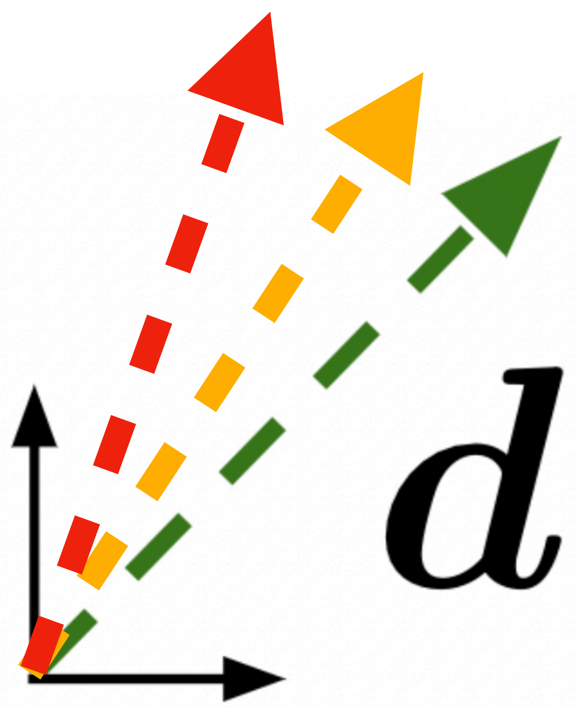


# 3 - Computing support points on shapes

$$S_{\mathscr{A}}(d) = \underset{x \in \mathscr{A}}{\text{argmin}}\ x^T d$$



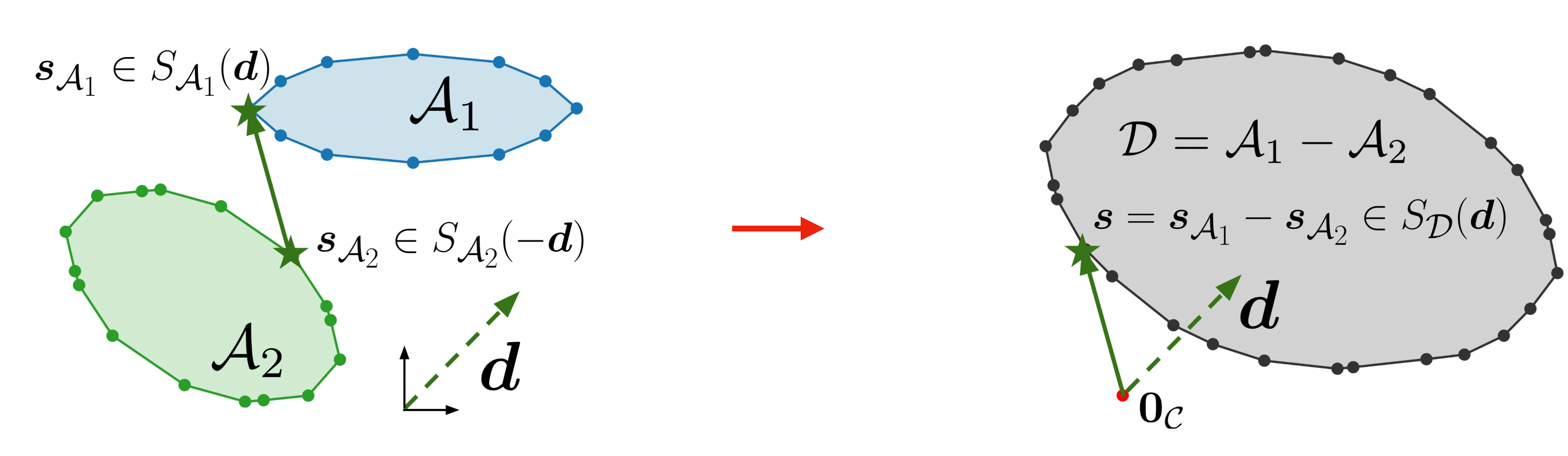

**Can be computed very efficiently for most shapes**



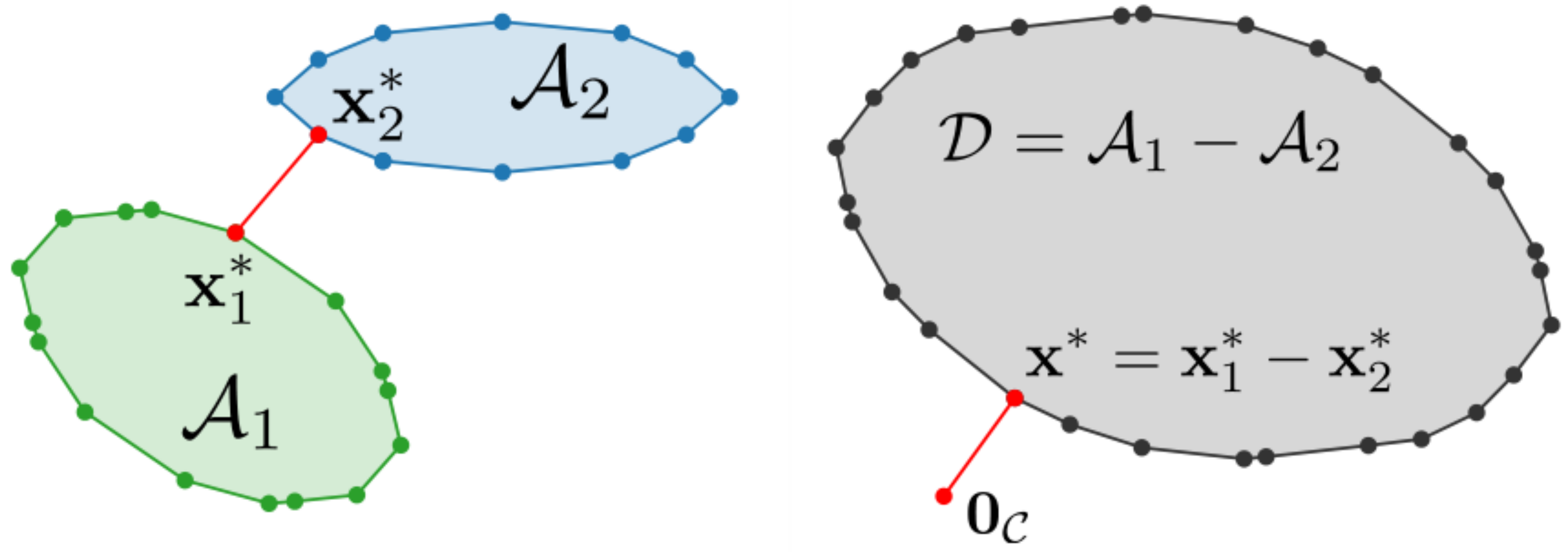
# 3 - Computing support points on a Minkowski difference

$$S_{\mathscr{A}}(d) = \underset{x \in \mathscr{A}}{\text{argmin}}\ x^T d$$

$s_1 \in S_{\mathscr{A}_1}(d)$
$s_2 \in S_{\mathscr{A}_2}(-d)$
→ $s = s_1 - s_2 \in S_{\mathscr{D}}(d)$



# 3 - Recap of collision detection with Frank-Wolfe




$$\min_{x_1 \in \mathcal{A}_1, x_2 \in \mathcal{A}_2} \frac{1}{2} || x_1 - x_2 ||^2 \longrightarrow \min_{x \in \mathcal{D}} \frac{1}{2} || x ||^2$$

**MNP**

**Frank-Wolfe = "constrained gradient descent", needs to compute support points:**

$$s = \text{argmin}_{y \in \mathcal{D}} < y, \nabla f(x) >$$

## Step 1 - What is collision detection?

## Step 2 - How to formulate a collision detection problem

## Step 3 - Solving a collision detection problem with Frank-Wolfe

## Step 4 - Accelerating Frank-Wolfe: the GJK algorithm

# 4 - Frank-Wolfe zigzags

**Algorithm** Frank-Wolfe

**Let** $\boldsymbol{x}_0 \in \mathcal{D}$, $\epsilon > 0$

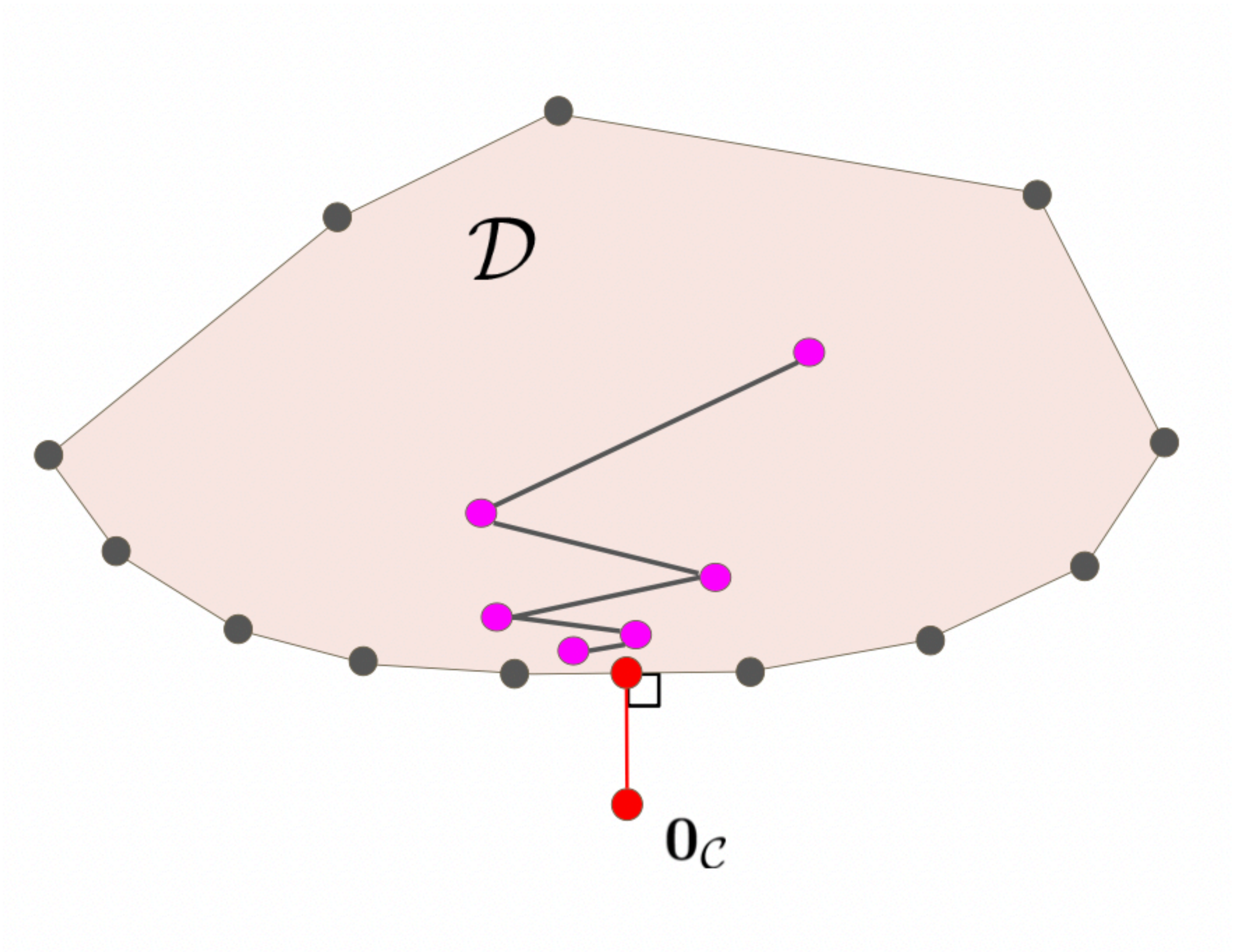
**For** `k=0, 1, ...` **do**

$\boldsymbol{s}_k \in \arg\min_{\boldsymbol{s} \in \mathcal{D}} \langle \boldsymbol{\nabla} f(\boldsymbol{x}_k), \boldsymbol{s} \rangle$ ▷ Support

**If** $g_{FW}(\boldsymbol{x}_k) \leq \epsilon$, **return** $f(\boldsymbol{x}_k)$ ▷ Duality gap

$\gamma_k = \arg\min_{\gamma \in [0,1]} f(\gamma \boldsymbol{x}_k + (1-\gamma)\boldsymbol{s}_k)$ ▷ Linesearch

$\boldsymbol{x}_{k+1} = \gamma_k \boldsymbol{x}_k + (1-\gamma_k)\boldsymbol{s}_k$ ▷ Update iterate



# 4 - From Frank-Wolfe to GJK

**Algorithm** Frank-Wolfe

**Let** $x_0 \in \mathcal{D}$, $\epsilon > 0$

**For** `k=0, 1, ...` **do**

1: $\boldsymbol{s}_k \in \arg\min_{\boldsymbol{s} \in \mathcal{D}} \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{s} \rangle$ ▷ Support

2: **If** $g_{FW}(\boldsymbol{x}_k) \leq \epsilon$, **return** $f(\boldsymbol{x}_k)$ ▷ Duality gap

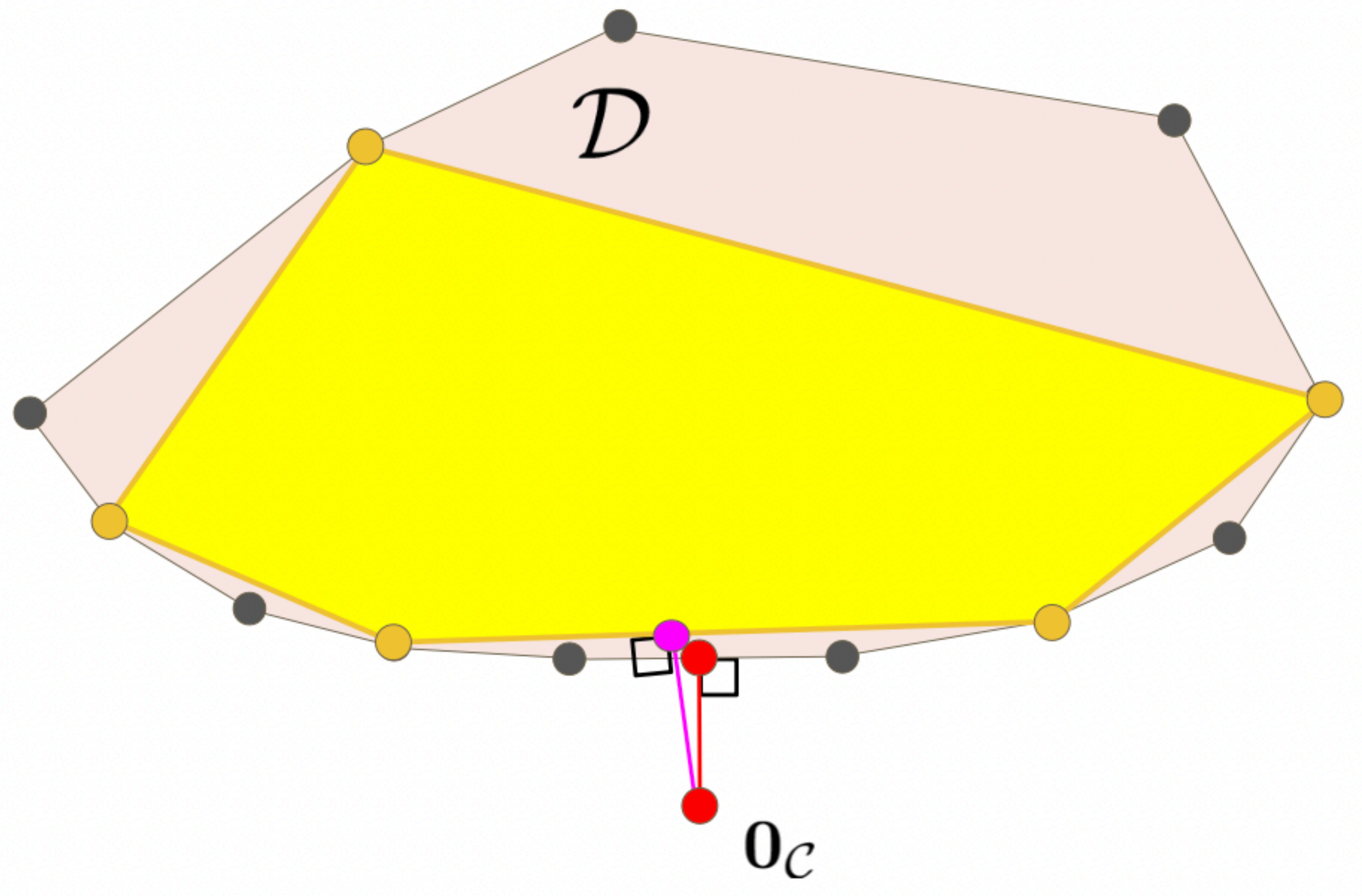
3: $\gamma_k = \arg\min_{\gamma \in [0,1]} f(\gamma \boldsymbol{x}_k + (1-\gamma)\boldsymbol{s}_k)$ ▷ Linesearch

4: $\boldsymbol{x}_{k+1} = \gamma_k \boldsymbol{x}_k + (1-\gamma_k)\boldsymbol{s}_k$ ▷ Update iterate

**Algorithm** Fully-Corrective Frank-Wolfe

*In Frank-Wolfe, replace line 3 and 4 by:*

1: $\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x} \in \text{conv}(\boldsymbol{s}_0, \dots, \boldsymbol{s}_{k-1})} f(\boldsymbol{x})$



# 4 - Nesterov accelerated Frank-Wolfe (or GJK)

**Algorithm** Frank-Wolfe

**Let** $\boldsymbol{x}_0 \in \mathcal{D}$, $\epsilon > 0$

**For** `k=0, 1, ...` **do**

1: $\boldsymbol{s}_k \in \arg\min_{\boldsymbol{s} \in \mathcal{D}} \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{s} \rangle$ ▷ Support

2: **If** $g_{FW}(\boldsymbol{x}_k) \leq \epsilon$, **return** $f(\boldsymbol{x}_k)$ ▷ Duality gap

3: $\gamma_k = \arg\min_{\gamma \in [0,1]} f(\gamma \boldsymbol{x}_k + (1-\gamma)\boldsymbol{s}_k)$ ▷ Linesearch

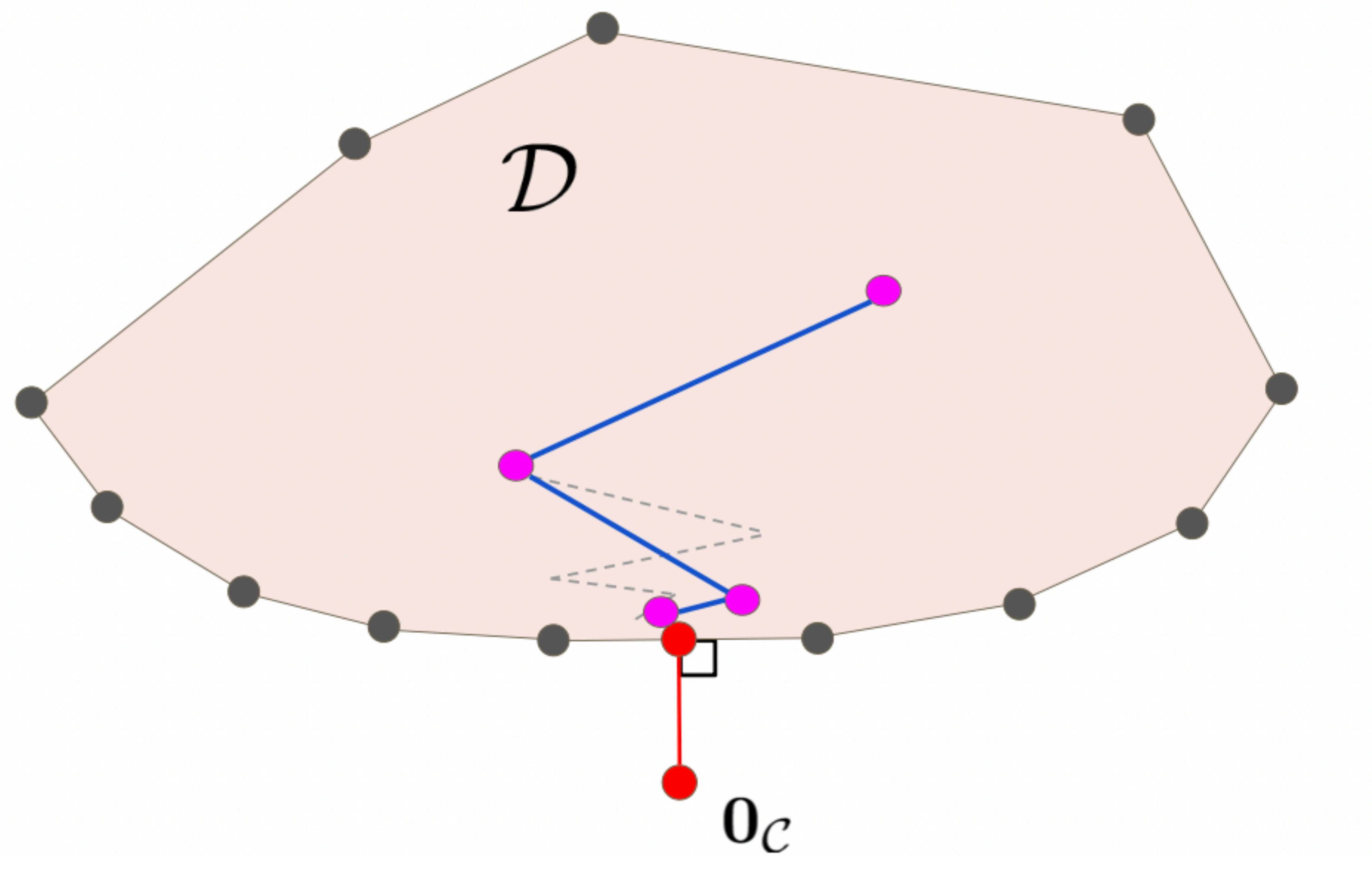
4: $\boldsymbol{x}_{k+1} = \gamma_k \boldsymbol{x}_k + (1-\gamma_k)\boldsymbol{s}_k$ ▷ Update iterate

**Algorithm** Nesterov-accelerated Frank-Wolfe

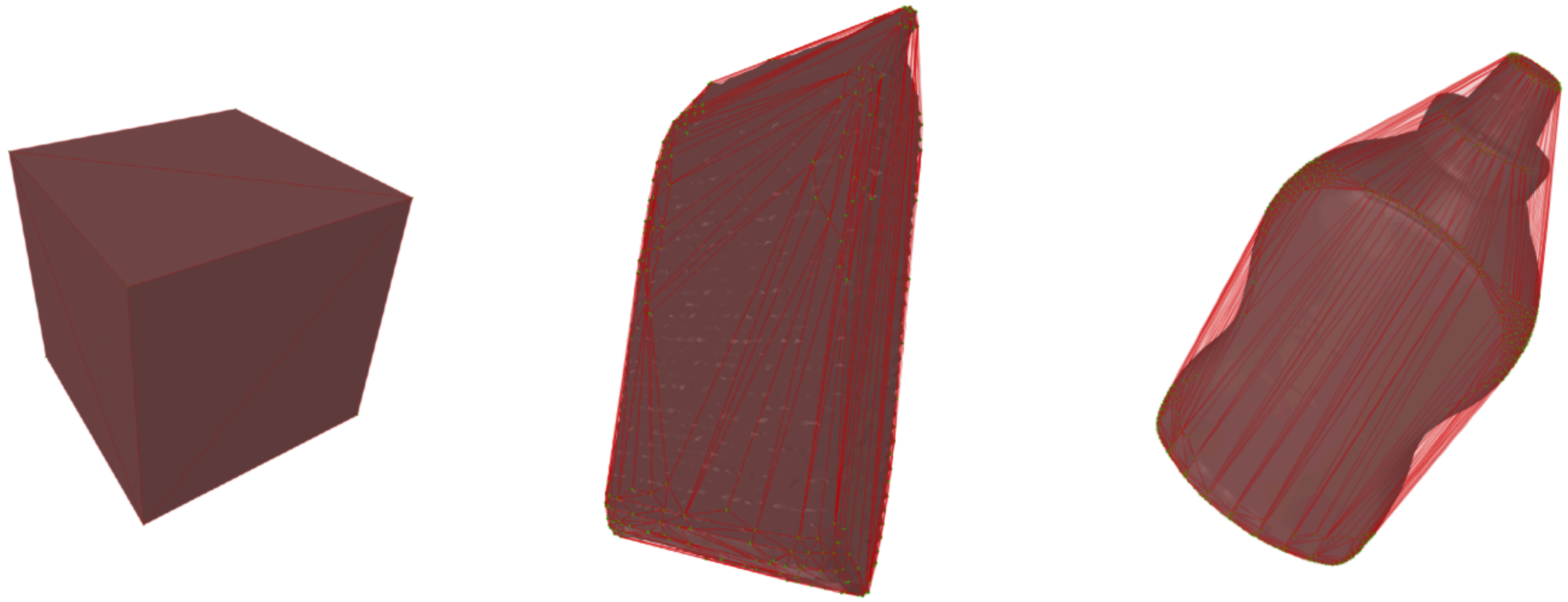
*In Frank-Wolfe, let* $\boldsymbol{d}_{-1} = \boldsymbol{s}_{-1} = \boldsymbol{x}_0$, $\delta_k = \frac{k+1}{k+3}$ *and replace line 1 by:*

1: $\boldsymbol{y}_k = \delta_k \boldsymbol{x}_k + (1-\delta_k)\boldsymbol{s}_{k-1}$

2: $\boldsymbol{d}_k = \delta_k \boldsymbol{d}_{k-1} + (1-\delta_k)\nabla f(\boldsymbol{y}_k)$

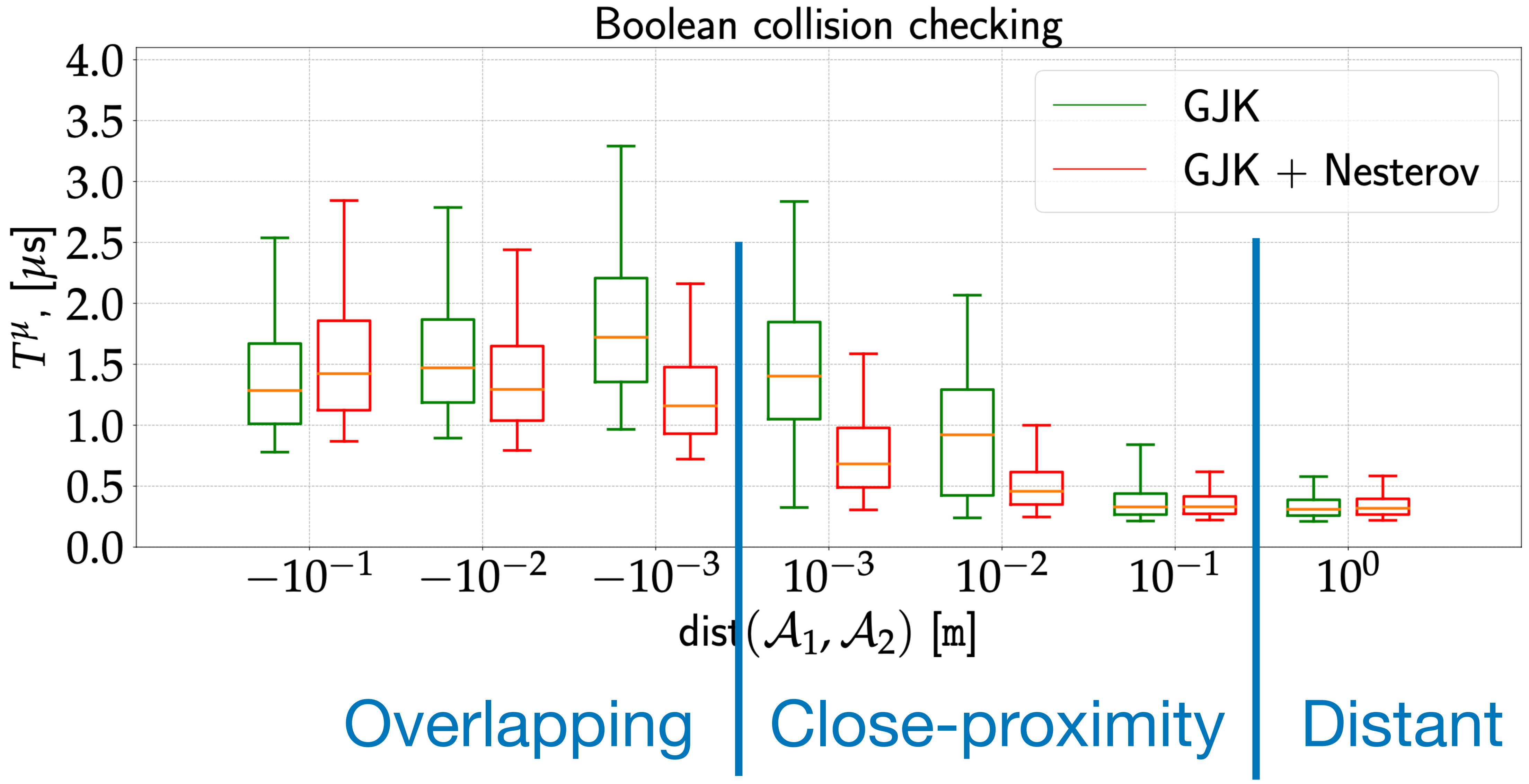


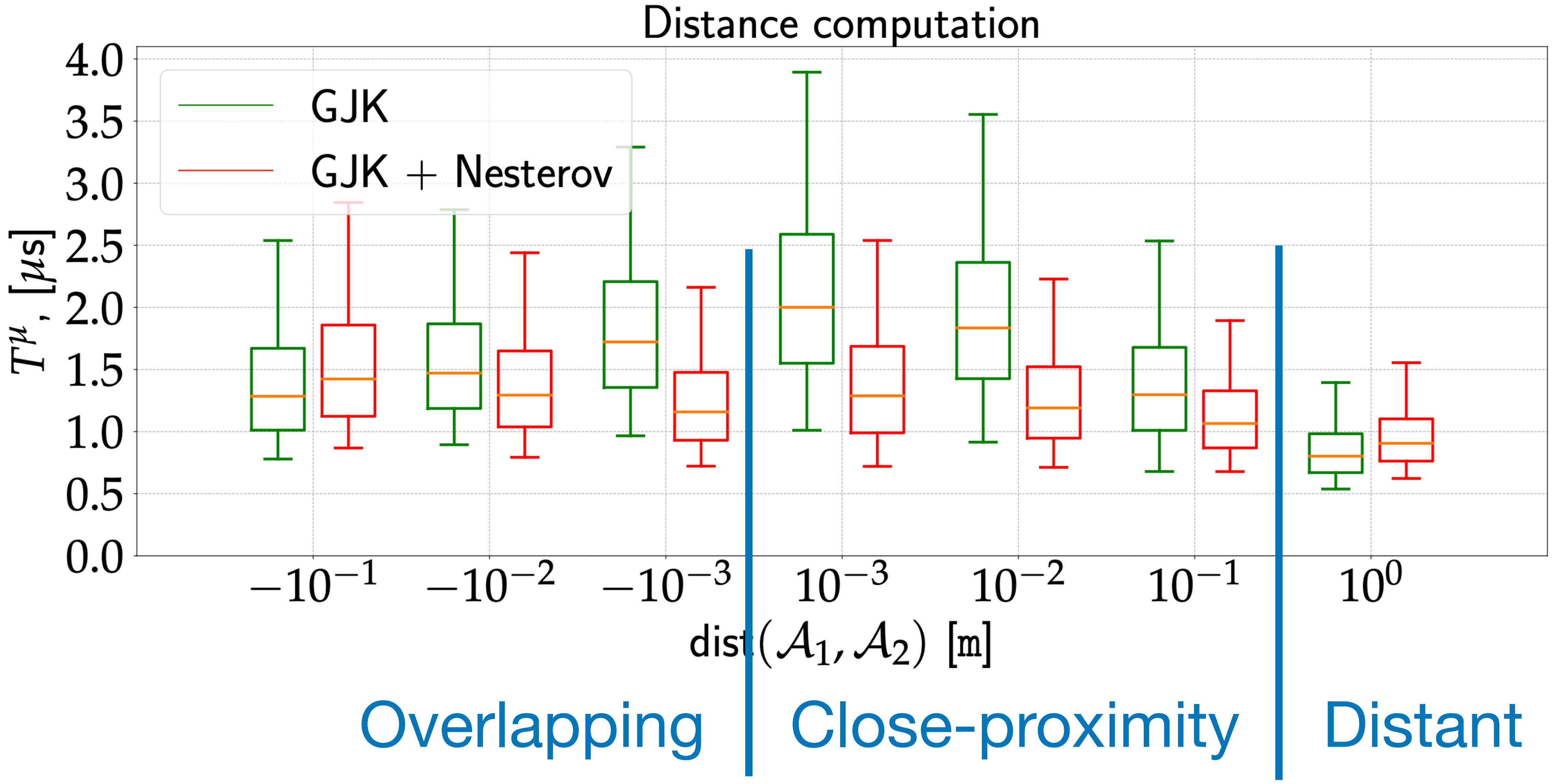
# 4 - Nesterov accelerated Frank-Wolfe (or GJK)

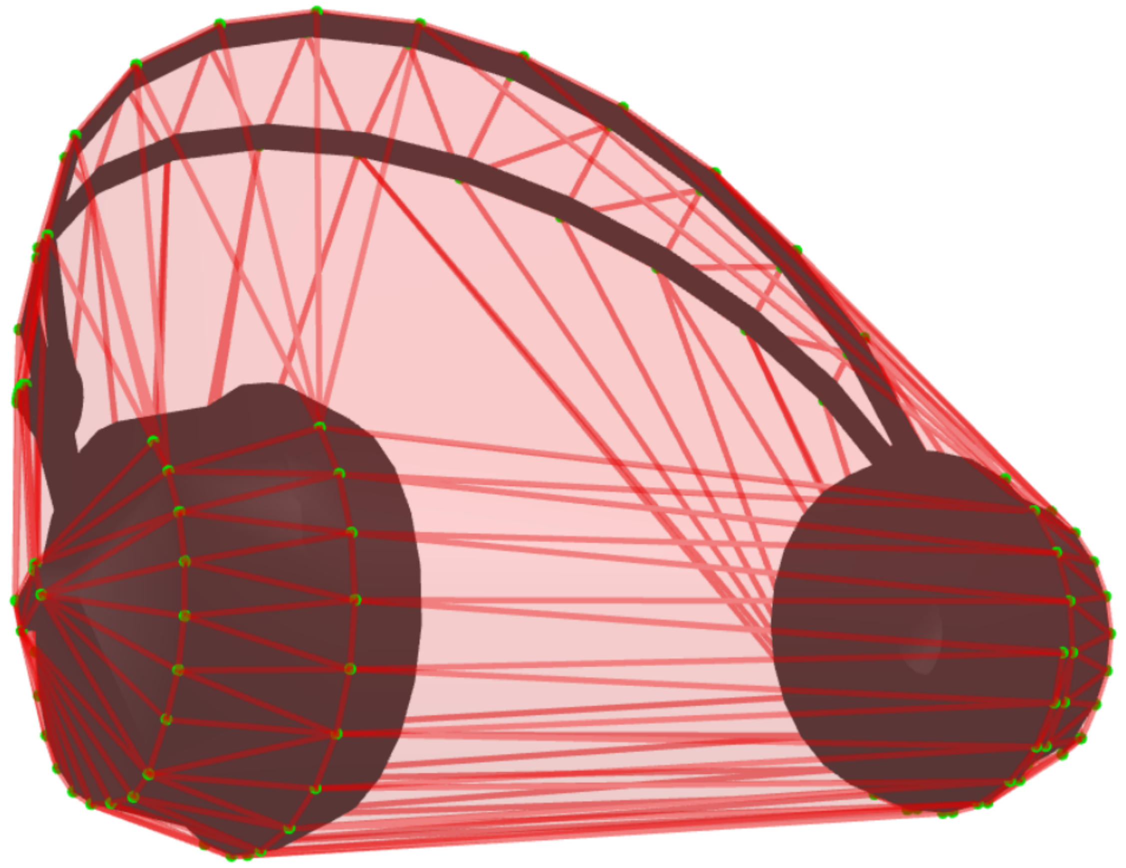


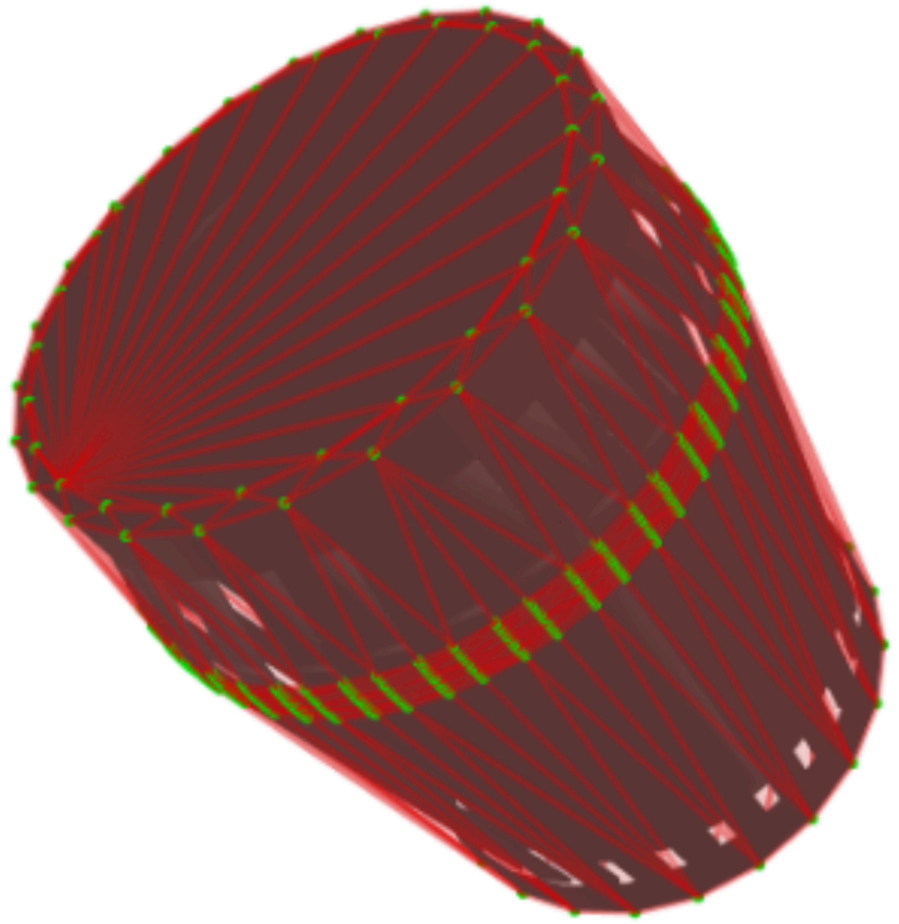
| | $N_v = 8$, $N_f = 6$ | $N_v = 250$, $N_f = 496$ | $N_v = 940$, $N_f = 1876$ |
|---|---|---|---|
| ProxQP | $5.3 \pm 2.7$ $\mu$s | $(2 \pm 0.6) \cdot 10^3$ $\mu$s | $(20 \pm 14) \cdot 10^3$ $\mu$s |
| GJK | $\mathbf{0.2 \pm 0.03}$ $\mu$s | $0.8 \pm 0.3$ $\mu$s | $2.1 \pm 0.5$ $\mu$s |
| Ours | $\mathbf{0.2 \pm 0.05}$ $\mu$s | $\mathbf{0.7 \pm 0.2}$ $\mu$s | $\mathbf{1.4 \pm 0.3}$ $\mu$s |

# 4 - Nesterov accelerated Frank-Wolfe (or GJK)







# Conclusion