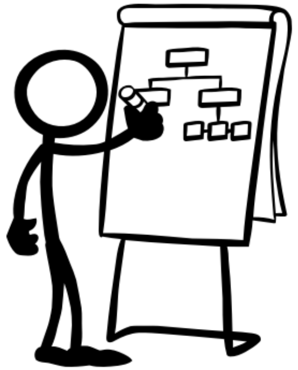
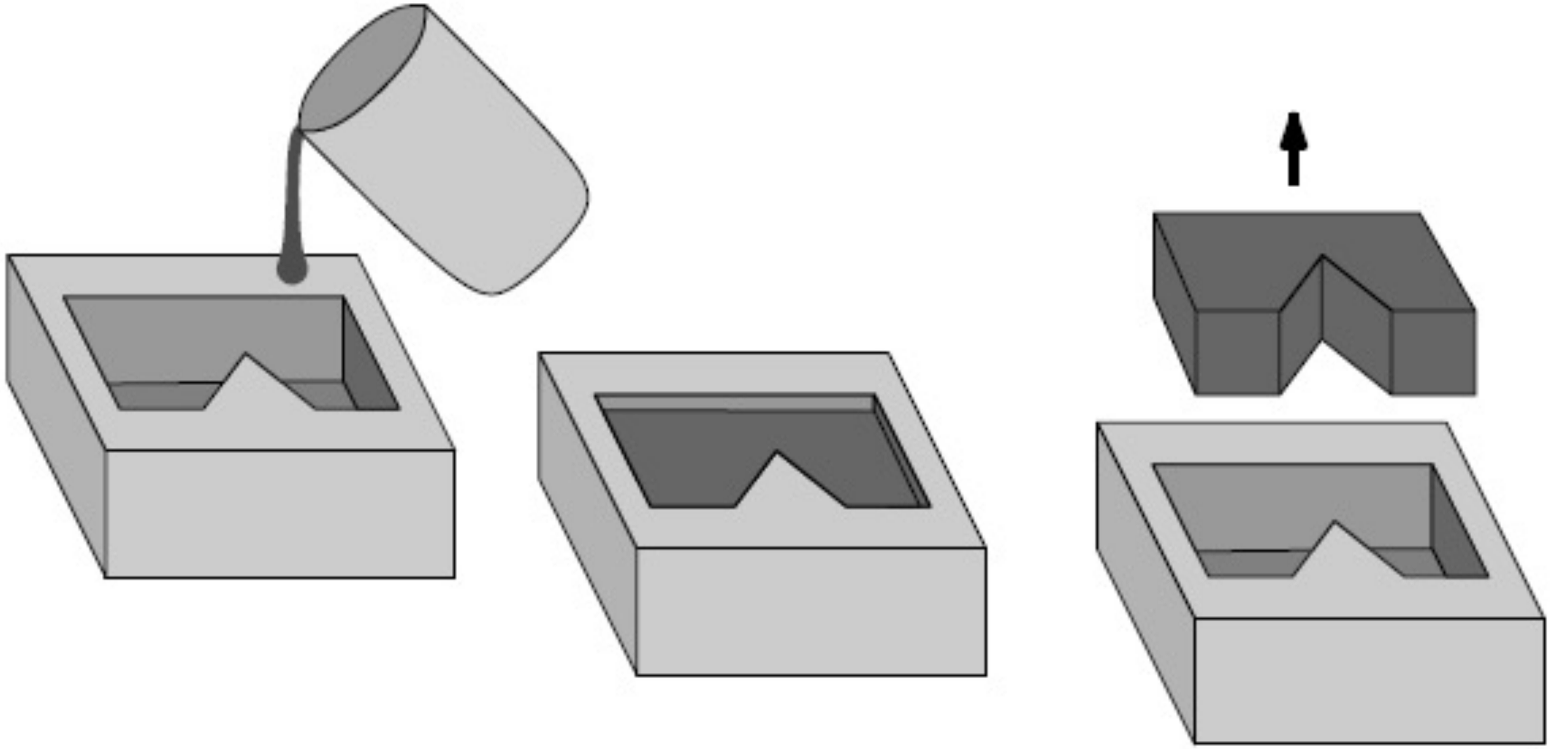
Polyhedron Casting and Backward Analysis

Computational Geometry – Recitation 5

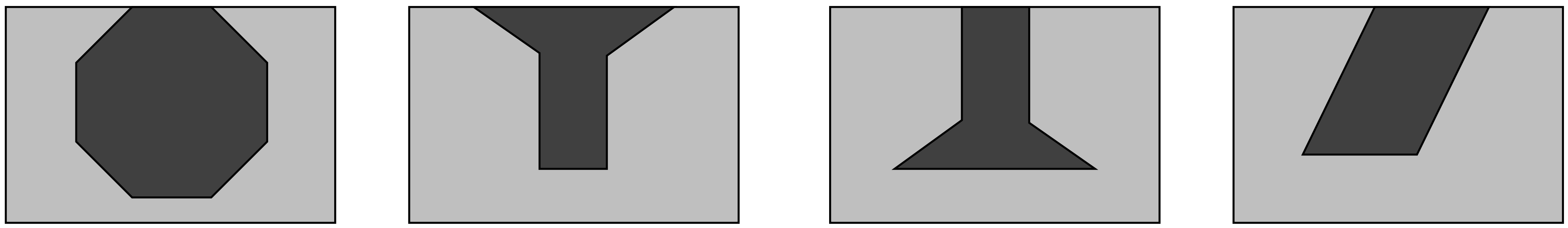


Casting



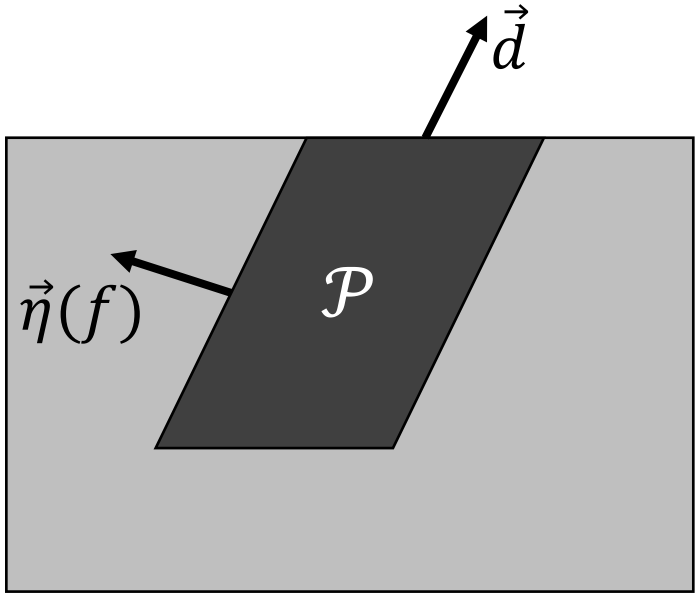
Casting

- Can we create a mold for any polyhedron?
- If a polyhedron is castable, does any mold fit?
- Given a legal mold, in what direction we need to translate? Upwards?



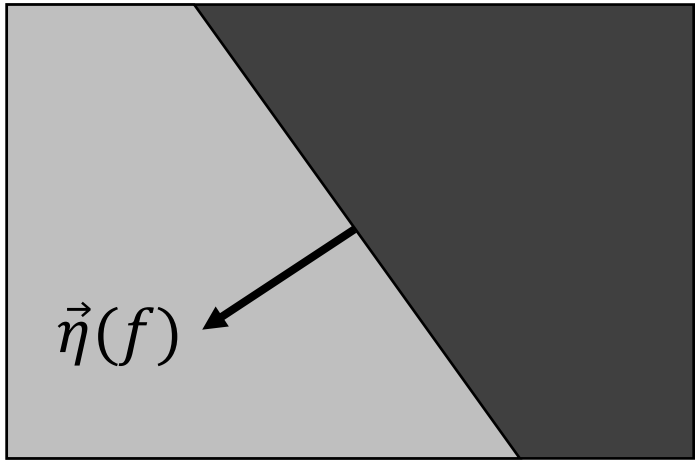
Definitions

- A polyhedron to cast - $\mathcal{P}$
- Each face of $\mathcal{P}$, f, have a corresponding face in the mold $\hat{f}$.
- The (outward) normal of f - $\vec{\eta}(f)$.
- Direction of translation - $\vec{d}$.



Castable polyhedrons

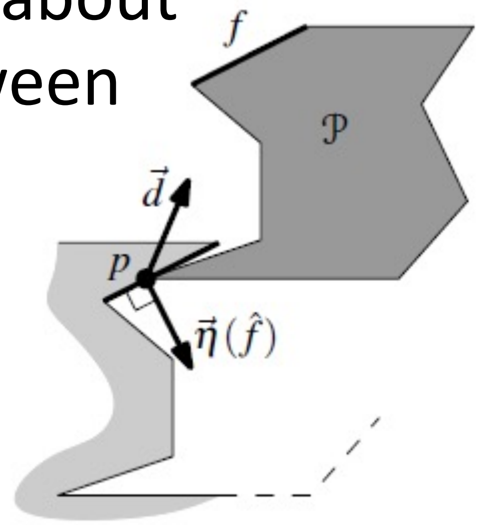
- Which directions $\vec{d}$ are valid?



- We want the angle between $\vec{d}$ and $\vec{\eta}(f)$ to be at least 90°.
- For each face!

Castable polyhedrons

- Lemma: The polyhedron $\mathcal{P}$ can be removed from its mold by a translation in direction $\vec{d}$ if and only if $\vec{d}$ makes an angle of at least $90°$ with $\vec{\eta}(f)$ for all f.
- Only if – We have already seen.
- If – The same reasoning holds for any collision, if $\mathcal{P}$ is about to collide with the mold at face $\hat{f}$ then the angle between $\vec{d}$ and $\vec{\eta}(f)$ is less than $90°$.



Castable polyhedrons

- Let us write the directions as vectors –

$$\vec{d} = (d_x, d_y, 1)$$

Why 1?

$$\vec{\eta} = (\eta_x, \eta_y, \eta_z)$$

- Recall that $\vec{d} \cdot \vec{\eta} = |\vec{d}| \cdot |\vec{\eta}| \cdot \cos(\theta)$
- We want θ to be greater than $90°$ for all faces, that is:

$$\vec{d} \cdot \vec{\eta} \leq 0 \Rightarrow$$

$$d_x \eta_x + d_y \eta_y + \eta_z \leq 0$$

- How do we solve it for all faces?

Castable polyhedrons

- We want θ to be greater than $90°$ for all faces, that is:
$$\vec{d} \cdot \vec{\eta} \leq 0 \Rightarrow$$
$$d_x\eta_x + d_y\eta_y + \eta_z \leq 0$$
- How do we solve it for all faces?
- Linear programing!
- Corollary: we can decide if a polyhedron is castable in $O(n^2)$ expected time.
    - Why $O(n^2)$?

Backward analysis

- What is the worst-case time complexity of the following algorithm?
- And the expected time?

Algorithm PARANOIDMAXIMUM(A)

if $\text{card}(A) = 1$
    then return the unique element $x \in A$
    else Pick a random element x from A.
        $x' \leftarrow$ PARANOIDMAXIMUM($A \setminus \{x\}$)
        if $x \leqslant x'$
            then return x'
            else Now we suspect that x is the maximum, but to be absolutely sure, we compare x with all $\text{card}(A) - 1$ other elements of A.
                return x

Backward analysis

- Worst case:
- $T(n) = T(n-1) + O(n) = O(n^2)$

Algorithm PARANOIDMAXIMUM(A)

```
if card(A) = 1
  then return the unique element x ∈ A
  else Pick a random element x from A.
       x' ← PARANOIDMAXIMUM(A \ {x})
       if x ≤ x'
         then return x'
         else Now we suspect that x is the maximum, but to be
              absolutely sure, we compare x with all card(A) − 1
              other elements of A.
              return x
```

Backward analysis

- Expected time:
- $\mathrm{E}\big(T(n)\big) = \mathrm{E}\big(T(n-1)\big) + \mathrm{E}(f(n))$

$$= \mathrm{E}\big(T(n-2)\big) + \mathrm{E}\big(f(n)\big) + E\big(f(n-1)\big)$$

$$\dots = \sum_{i=1}^{n} f(i)$$

$$= \frac{i-1}{i} O(1) + \frac{1}{i} O(i)$$

Algorithm ParanoidMaximum(A)

1. **if** $\text{card}(A) = 1$
2.   **then return** the unique element $x \in A$
3.   **else** Pick a random element x from A.
4. $x' \leftarrow$ParanoidMaximum$(A \setminus \{x\})$
5.   **if** $x \leqslant x'$
6.   **then return** x'
7.   **else** Now we suspect that x is the maximum, but to be absolutely sure, we compare x with all $\text{card}(A) - 1$ other elements of A.
8.   **return** x