# Risolvere collisioni tra poligoni convessi

Elia Calligaris

12 settembre 2015

# Indice

1. Fondamenti
   - Descrivere la posizione dei poligoni
   - Somme di Minkowski
2. Rilevare le collisioni
   - Una soluzione lineare
   - Ombre
   - Test in tempo logaritmico
3. Risolvere le collisioni
   - Adattare l'algoritmo
   - Casi particolari

# Enunciato del problema

Dati:

- due poligoni convessi A e B, tali che $A \cap B \neq \emptyset$
- una retta orientata $d$ (direzione)

Ci si propone di:

- trovare la minima distanza $\sigma_d(A, B)$ necessaria a separare i due poligoni muovendo A lungo $d$;
- risolvere tale problema partendo dall'analisi di uno più semplice;
- arrivare ad una soluzione a costo logaritmico.

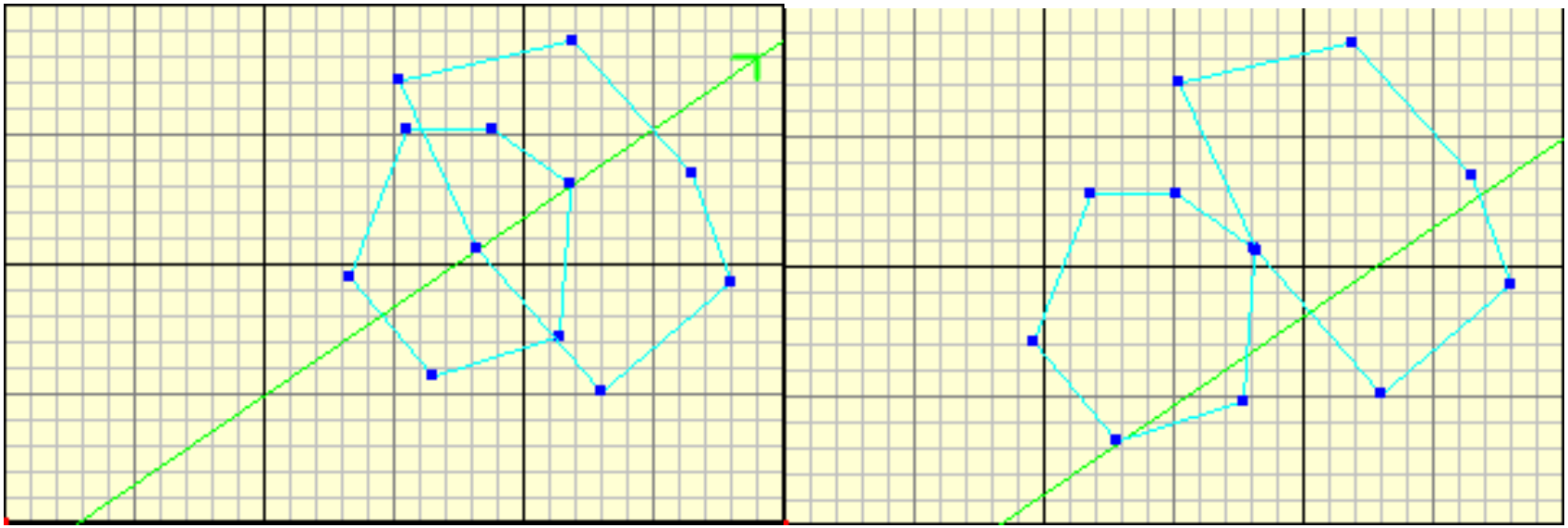


Figura 1: Esempio di risoluzione di collisione fra poligoni convessi.

Sezione 1

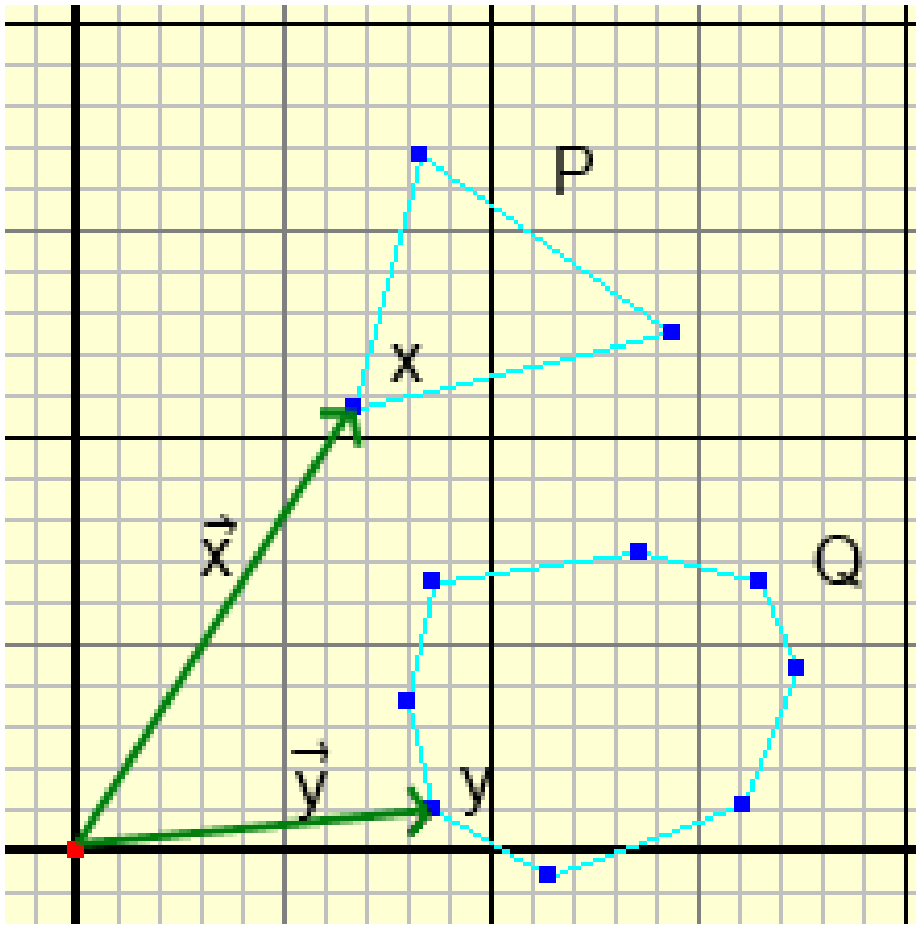
# Fondamenti

# Descrivere la posizione dei poligoni

- Per ogni poligono P si prende un punto di riferimento $x$ arbitrario;
- La posizione di P è descritta dal vettore $\vec{x}$.
- In realtà non si prende un punto a caso...

## Notazione

- Con $P^{\vec{x}}$ si indica il poligono P spostato di $\vec{x}$ rispetto all'origine;
- Con solamente $P$ si intende che il poligono ha il suo punto di riferimento nell'origine;
- Con $-P$ si intende il poligono capovolto rispetto all'origine, cioè $-P = \{-p : p \in P\}$.


P
x
$\vec{x}$
Q
$\vec{y}$
y

# Somme di Minkowski
## Definizione

Nozione su cui si basano gli algoritmi a seguire.

### Definizione 1

Dati due poligoni A e B, la loro *somma di Minkowski* è definita come: $A \oplus B = \{a + b : a \in A, b \in B\}$

### Notazione

Per compattezza, "*Somme di Minkowski*" verrà abbreviato in *sdM*

Che significato hanno le *sdM*?

# Somme di Minkowski
## Definizione

Nozione su cui si basano gli algoritmi a seguire.

### Definizione 1

Dati due poligoni A e B, la loro *somma di Minkowski* è definita come: $A \oplus B = \{a + b : a \in A, b \in B\}$

### Notazione

Per compattezza, "*Somme di Minkowski*" verrà abbreviato in *sdM*

Che significato hanno le *sdM*?

# Somme di Minkowski
## Significato

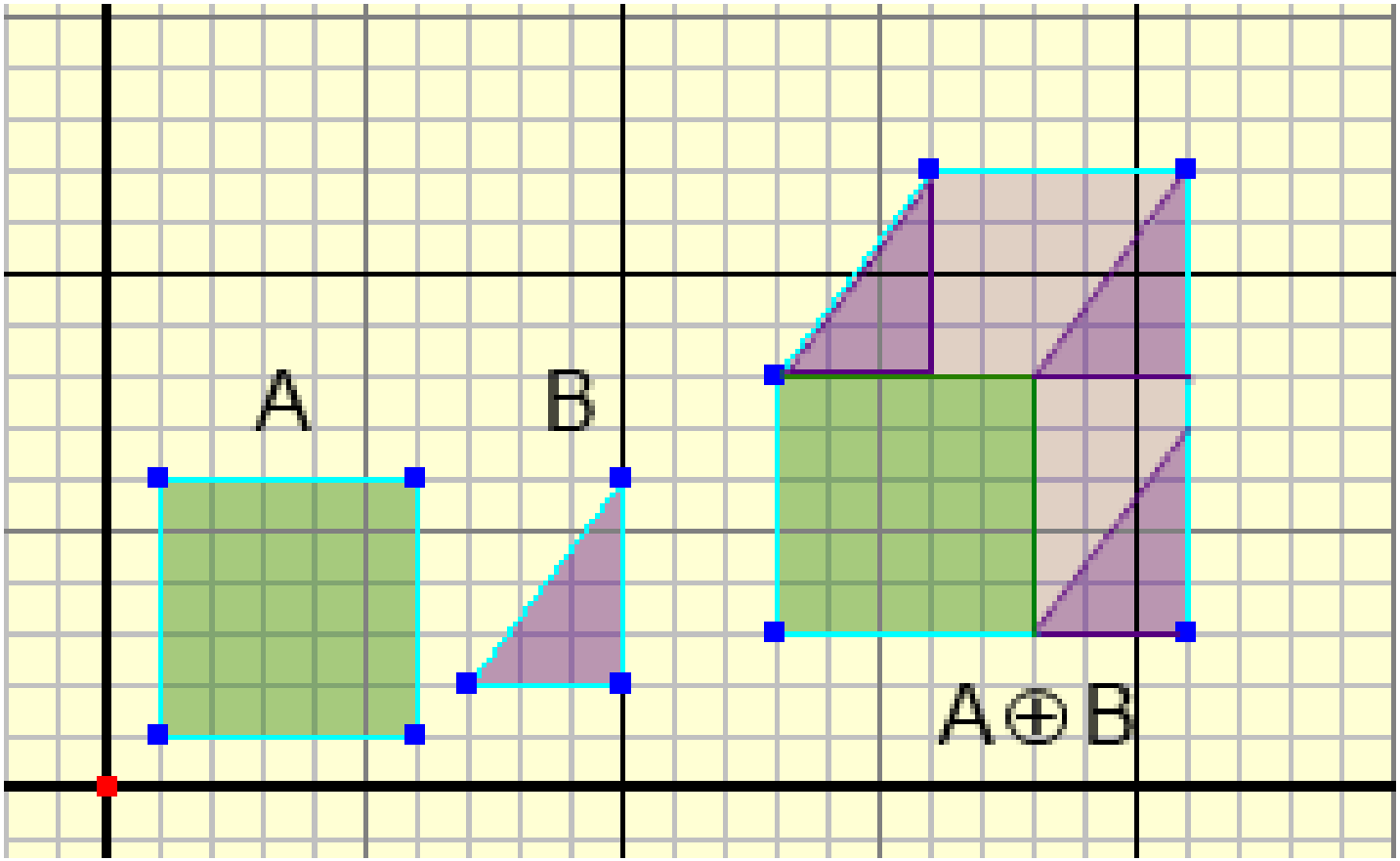



Figura 2: Visualizzazione delle somme di Minkowski.

# Somme di Minkowski
## Proprietà

**Teorema 1**

*La somma di Minkowski di due poligoni convessi è, a sua volta, un poligono convesso.*

# Somme di Minkowski
## Analisi dei costi

Siano $A$ e $B$ due poligoni convessi, con $n$ ed $m$ punti rispettivamente:

- algoritmo a forza bruta: costo $O(n \times m)$.
  - Calcolare la somma di tutte le coppie di punti;
  - Costruire il *convex hull* dei punti ottenuti;
  - $\Rightarrow$ *Vengono calcolati punti superflui!*
- algoritmo per poligoni convessi: costo $O(n + m)$.
  - Lati dei poligoni ordinati per angolo di rotazione;
  - Assemblare i lati di entrambi i poligoni in ordine di angolo crescente.

# Algoritmo per poligoni convessi

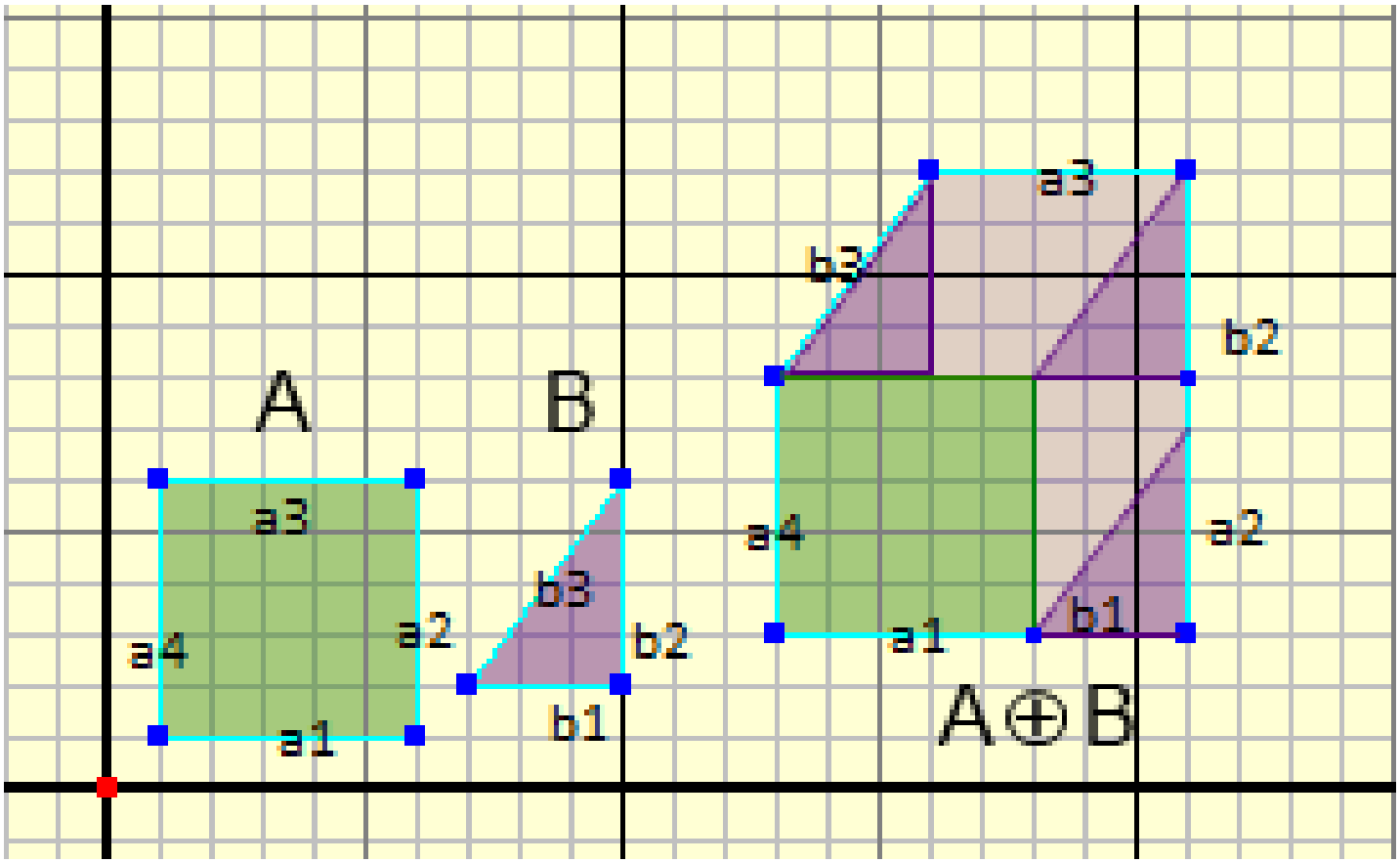



Figura 3: Visualizzazione della costruzione della *sdM* di due poligoni convessi.

Sezione 2

# Rilevare le collisioni

# Rilevare le collisioni
## Collision detection

- Problema più semplice della risoluzione delle collisioni;
- Determinare se due poligoni collidono (i.e. $A \cap B \neq \emptyset$);
- Algoritmo a costo $O(\log(n))$, dove $n = \max(|A|, |B|)$, proposto da Guibas e Stolfi e basato sulle *sdM*;
- Tale algoritmo verrà adattato per risolvere le collisioni.

# Una prima soluzione...

## ... a costo lineare

**Teorema 2**

*Dati due poligoni* $A^{\vec{x}}$ *e* $B^{\vec{y}}$*:*
$A^{\vec{x}} \cap B^{\vec{y}} \neq \emptyset \iff \vec{x} - \vec{y} \in -A \oplus B \vee \vec{y} - \vec{x} \in A \oplus -B.$

Tuttavia, come si è visto, calcolare la *sdM* ha costo almeno lineare.

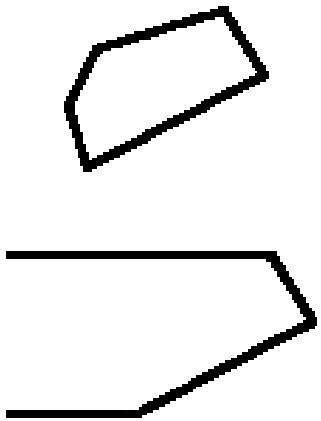
$\Rightarrow$ *Non serve costruire tutta la sdM*!

# Ombre

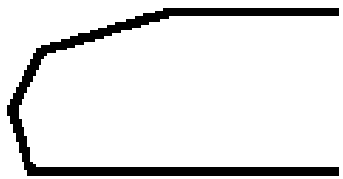
Un'*ombra* destra/sinistra di un poligono P (denotata $P_R/P_L$) è ciò che si ottiene "trascinando" P verso destra/sinistra, All'infinito. Da questa definizione si ottengono alcuni risultati interessanti.

**Teorema 3**

$$P_L \cap P_R = P$$



(b)
Ombra sinistra



(c)
Ombra destra

**Teorema 4**

$A \cap B \neq \emptyset \Leftrightarrow A_R \cap B_L \neq \emptyset \wedge A_L \cap B_R \neq \emptyset.$

Di conseguenza, riformulando il teorema 2, si può:

- Prendere un'ombra sinistra $L^{\vec{u}}$ ed una destra $R^{\vec{v}}$;
- Testare se $\vec{w} = \vec{v} - \vec{u}$ sta in $L \oplus -R$.
  $\Rightarrow$ Con una tecnica particolare!

**Osservazione**

Si noti che -R è un'ombra sinistra!

## Teorema 4

$A \cap B \neq \emptyset \Leftrightarrow A_R \cap B_L \neq \emptyset \wedge A_L \cap B_R \neq \emptyset.$

Di conseguenza, riformulando il teorema 2, si può:

- Prendere un'ombra sinistra $L^{\vec{u}}$ ed una destra $R^{\vec{v}}$;
- Testare se $\vec{w} = \vec{v} - \vec{u}$ sta in $L \oplus -R$.
  $\Rightarrow$ Con una tecnica particolare!

## Osservazione

Si noti che -R è un'ombra sinistra!

# Test in tempo logaritmico
## sfruttando le ombre

- Siano $l'$ ed $r'$ i punti iniziali delle catene $L$ ed $-R$, rispettivamente;
- Siano $l''$ ed $r''$ i loro punti finali;
- Allora $L \oplus -R$ sarà una catena convessa (verso destra) che va da $l' + r'$ a $l'' + r''$; in particolare, sarà la catena che sta più a destra di tutte le altre.

- Siano $f$ e $g$ i lati mediani di $L$ ed $-R$;
- Si avrà allora che le ombre saranno da essi divise in una parte superiore ed una inferiore, indicate con $L_H/L_L$ e $-R_H/-R_L$;
- Senza perdita di generalità, si supponga che $f$ venga prima di $g$ in ordine di angolo: allora si può definire la catena $D$, composta da $L_L \oplus -R_L, f, g$ e $L_H \oplus -R_H$;
- $D$ sarà una catena convessa verso destra, ma non è detto che corrisponda a $L \oplus -R$

Con riferimento alla figura a fianco, $\vec{w}$ può cadere in una delle seguenti aree:

- LEFT: $\vec{w}$ è sicuramente dentro $L \oplus -R$, quindi c'è una collisione;
- ABOVE: $\vec{w}$ è sopra $g$, quindi $f$ e $L_L$ possono essere scartati in quanto irrilevanti;
- BELOW: $\vec{w}$ è sopra $f$, quindi $g$ e $-R_H$ possono essere scartati in quanto irrilevanti;

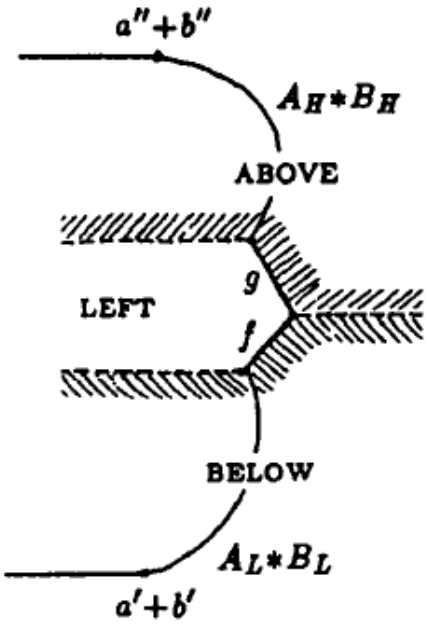



Figura 4:
Immagine tratta da [2]

- Se $\vec{w}$ non cade in LEFT, dopo un certo numero di passi una delle due ombre (ipotizziamo sia $L$) sarà ridotta ad un singolo vertice $\vec{x}$;
- A questo punto basta vedere se $\vec{w} - \vec{x}$ sta dentro $-R$ tramite una semplice ricerca binaria (a costo $O(\log(n))$).
- $\Rightarrow$ **Quindi abbiamo risolto il problema ad un costo complessivo di** $O(\log(n))$.

**Osservazione**

Costruire le ombre dei poligoni ha costo lineare, quindi devono pre-computate.

Sezione 3

# Risolvere le collisioni

# Risolvere le collisioni
## Penetration Depth

A questo punto bisogna adattare l'algoritmo descritto da Guibas e Stolfi per determinare $\sigma_d$, ovvero la profondità di penetrazione:

- L'algoritmo precedente trova il lato di $L \oplus -R$ che interseca il raggio orizzontale (verso destra) da $\vec{w}$;
- Adesso invece si vuole trovare il lato $e$ che interseca il raggio $r$ che parte da $\vec{w}$ in direzione $d$;
- Se $z$ è il punto di intersezione tra $e$ ed $r$, allora $\sigma_d = \vec{z} - \vec{w}$;
- Come annunciato, questa procedura manterrà il costo logaritmico.

# Adattare l'algoritmo

- Si costruisce la catena $D$, come in precedenza;
- Dati i lati mediani $f$ e $g$:
  - se $r$ interseca o sta sotto f, si procede come in BELOW;
  - se interseca o sta sopra g, si procede come in ABOVE.
- Si itera fino a ridurre un'ombra ad un singolo punto $\vec{x}$;
- Quindi $\vec{w} = \vec{w} - \vec{x}$, e $r$ va modificato affinché passi per $\vec{w}$;
- Si esegue una ricerca binaria per trovare un'intersezione tra $r$ ed un lato dell'ombra rimanente:
  - Se viene trovata, abbiamo il punto $\vec{z}$ grazie al quale possiamo determinare la profondità della collisione;
  - Altrimenti significa che non c'è collisione alcuna.

A questo punto abbiamo trovato $\sigma_d$ ed è possibile spostare il primo poligono per risolvere la collisione!

# Casi particolari da gestire

- È possibile che $r$ sia parallelo sia ad $f$ che $g$: guardare il coefficiente angolare di $r$;
- Complicazioni dovute al modo di rappresentare una retta orientata (raggio).

# Riferimenti bibliografici


D. Dobkin, J. Hershberger, D. Kirkpatrick, S. Suri
*Computing the Intersection-Depth of Polyhedra*


L.J. Guibas and J. Stolfi
*Ruler, Compass and Computer: The Design and Analysis of Geometric Algorithms*

# Grazie per l'attenzione!