

# Nueva propuesta para detección de contacto entre poliedros a gran escala

Yerko Zec



**Universidad  
Andrés Bello®**  
Conectar • Innovar • Liderar

September 27, 2019

# Contenido

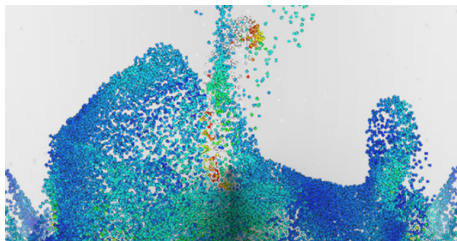
- 1 Motivación
- 2 Historia
- 3 Problema
- 4 Objetivo
- 5 Metodología
- 6 Conclusion
- 7 Referencias

# Motivación

- Detección de contacto entre poliedros.
- Simulación de grandes movimientos de rocas propuesto por Cundall [1].
- Nueva propuesta para detección de contacto entre poliedros.
- Problema de detección aun no se encuentra resuelto en su totalidad.

# Historia

- En 1971 Cundall [1] elaboro un método llamado Discrete Element Method (DEM).
- DEM está dividido en 3 fases: neighbor search, contact detection y force calculations.
- Esta propuesta simula grandes movimiento de partículas.
- Hasta la fecha existen más de 15 algoritmos para la detección de contacto entre partículas.

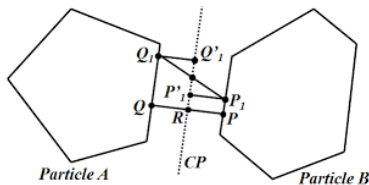


# Common-Plane

- En 1988 Cundall[1] propuso un algoritmo llamado Common-Plane (CP)
- Mencionando algunos algoritmos utilizados a lo largo de la historia son: CP, FCP, MR, SLM, Multi-grid y MSC entre otros.
- Estos algoritmos empezaron desde la misma base que ocupa el algoritmo Common-Plane.

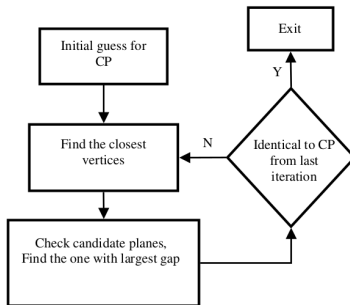
# Common-Plane

- Common-Plane(CP), fue en primera instancia creado por Cundall en 1988.
- El principio de este algoritmo, es localizar un plano entre dos figuras e ir calculando la distancias.



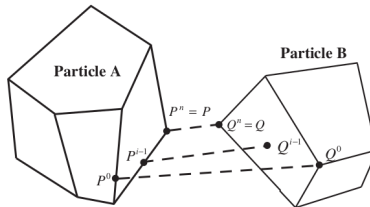
# Fast Common-Plane

- En 2004 Nezami [2] propuso una nueva propuesta para el calculo del CP y se llamo Fast Common-Plane (FCP).
- Con esta nueva propuesta mejoro el orden del algoritmo por ello la eficiencia.



# Shortest Link Method

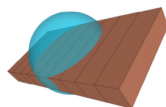
- Nezami en 2006 [3] propuso otro algoritmo el cual ocupa como base FCP.
- En base a resultados expuestos por Nezami [3] SLM es 17 veces mas rápido que otros algoritmos convencionales.



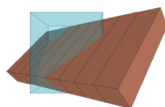


# Multi-shell Contact Detection

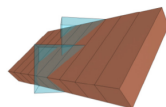
- Fue desarrollado por Zhuang el 2014 [4].
- La detección de contacto y la búsqueda de vecindario.
- MSC como método tanto de detección de vecindario como detección de contacto es uno de los mas eficiente, según concluye Zhuang.



(d) 3D NBS



(e) 3D DESS



(f) 3D MSC

# Problema

- Detección de contacto computacionalmente costoso.
- Análisis de gran cantidad de partículas.

# Objetivo

## Objetivo General

- Detectar de forma eficiente y eficaz el contacto entre poliedros.

## Objetivo Especifico

- Generar algoritmo de detección de contacto mediante método propuesto.

# Metodología

- Se realizó un estudio sobre el tema propuesto, donde se recopilaban los datos pertinentes.
- Se generó una línea de tiempo donde se señalaron los trabajos más relevantes al tema.

- Principales framework de trabajo son github, pycharm, kile y overleaf.



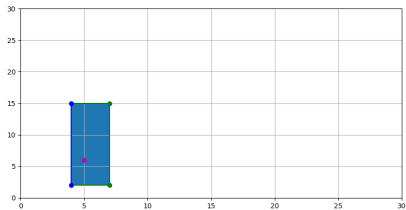
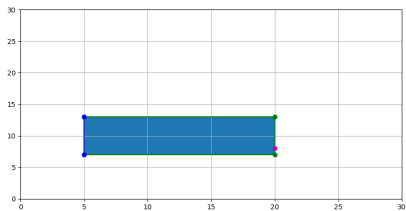
## Algoritmo

```
7 def genfig():
8
9     vectorx = random.sample(range(30), 2)
10    vectory = random.sample(range(30), 2)
11    '''
12    vectorx = np.array([5, 20])
13    vectory = np.array([7, 13])
14    '''
15    return (vectorx, vectory)
16
17 def genpunto():
18     punto = random.sample(range(20), 2)
19     #punto = np.array([5, 13])
20     return punto
```

## Algoritmo

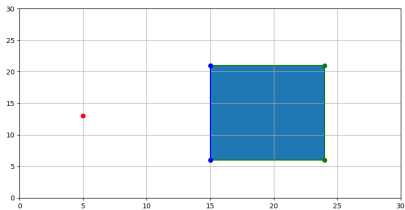
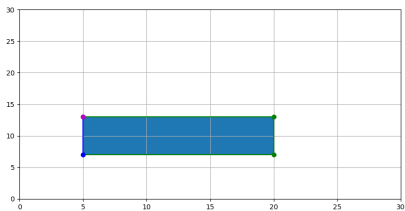
```
22 def pointdetection(figura,punto):
23     vertex = np.array(getvertex(figura))
24     U0 = np.array(vertex[0])
25     U1 = np.array(vertex[1])
26     U2 = np.array(vertex[2])
27     U3 = np.array(vertex[3])
28     #-----
29     b1 = U1 - U0
30     b3 = U3 - U0
31     print("\n b1: \n", b1, "\n b3:\n", b3)
32
33     normb1 = np.linalg.norm(b1, ord=2)
34     normb3 = np.linalg.norm(b3, ord=2)
35
36     print("\n norma1: \n", normb1, "\n norma0: \n", normb3)
37
38     b1v = (b1/ normb1)
39     b3v = (b3/ normb3)
40
41     B = np.array([[b1v[0], b1v[1]],
42                  [b3v[0], b3v[1]]])
43
44     print("\n B: \n", B)
45     V0 = punto[0]-U0[0]
46     V1 = punto[1]-U0[1]
47
48     V = np.array([[V0], [V1]])
49     print("\n V: \n", V)
50     alpha = np.linalg.solve(B, V)
51     print("\n Alfa: \n", alpha)
52     resultx,resulty = False, False
53     if(0 <= alpha[0] <= max(b1)):
54         resultx = True
55     if(0 <= alpha[1] <= max(b3)):
56         resulty = True
57     return resultx and resulty
```

## Algoritmo









## Algoritmo



- Para el control de versiones del código se utilizó un repositorio en github.

 <b>yerkozec</b> modified: diapos reuniones/hito1/hito1.tex ...		Latest commit 2d61ebe 10 hours ago
 ptgit	modified: diapos reuniones/hito1/hito1.tex	10 hours ago
 .gitignore	Initial commit	13 hours ago
 README.md	Update README.md	13 hours ago

# Conclusion

- Cabe mencionar que existen varias soluciones al problema propuesto, pero aun así no esta completamente resuelto.
- Se busca realizar la detección de contacto de la forma más eficiente posible ocupando la menor cantidad de recursos computacionales posibles.

# Referencias



Cundall P. A.

Formulation of a three-dimensional distinct element model-part i. a scheme to detect and represent contacts in a system composed of many polyhedral blocks.

1988.



Nezami G. Erfan.

A fast contact detection algorithm for 3-d discrete element method.

2004.



Nezami G. Erfan.

Shortest link method for contact detection in discrete element method.

2006.



Zhuang Z.

A multi-shell cover algorithm for contact detection in the

# Nueva propuesta para detección de contacto entre poliedros a gran escala

Yerko Zec



**Universidad  
Andrés Bello®**  
Conectar • Innovar • Liderar

September 27, 2019