Nueva propuesta para detección de contacto entre poliedros a gran escala

Yerko Zec

Universidad Andrés Bello®
Conectar • Innovar • Liderar

January 27, 2020

Contenido

Motivación

Figure 1: Derrumbe de masa tierra

Motivación

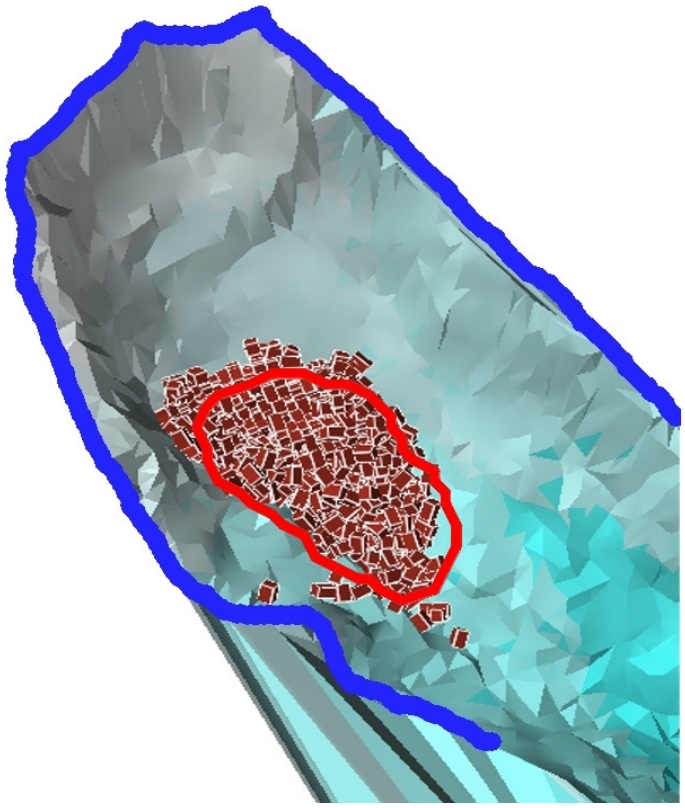


Figure 2: Detección de contacto

Pregunta de investigación

¿Se podrá realizar?

Nueva propuesta para la detección de contacto entre cuerpos rígidos.

Marco teórico

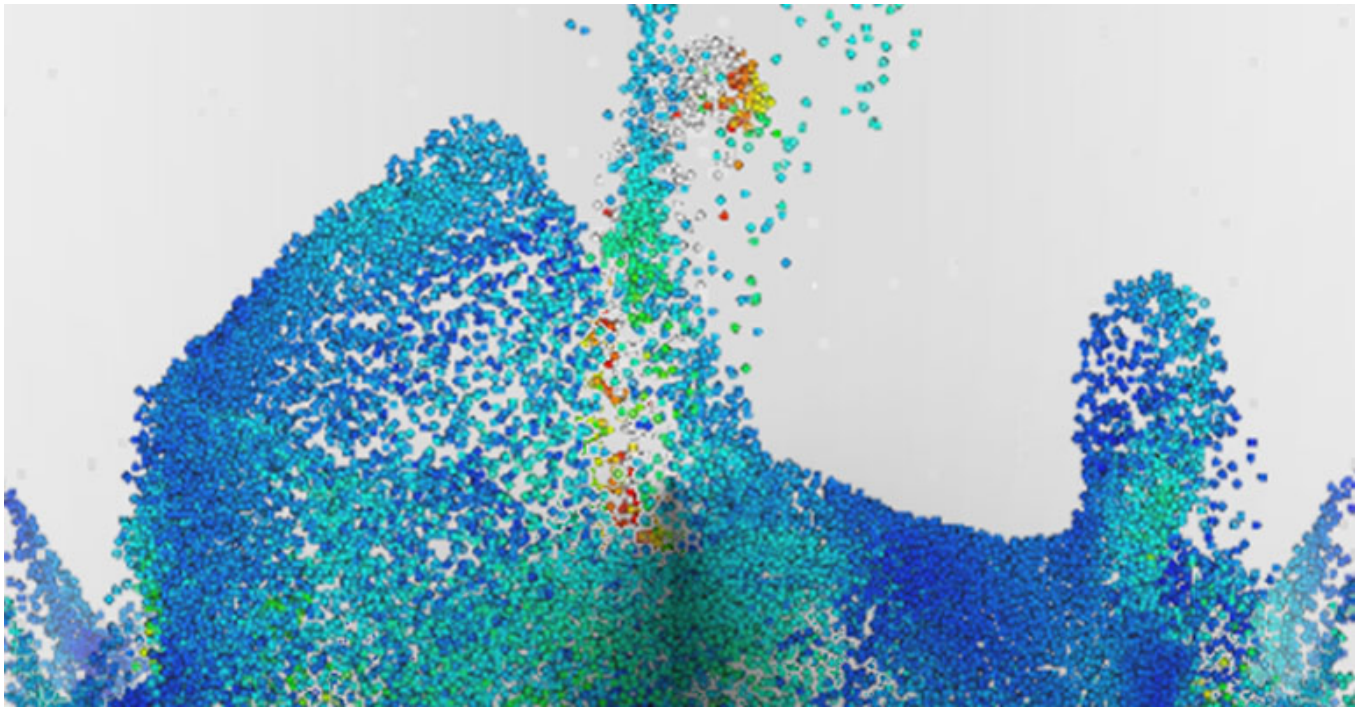


Figure 3: Discrete element method

Common-Plane

- Common-Plane(CP), fue en primera instancia propuesto por Cundall.

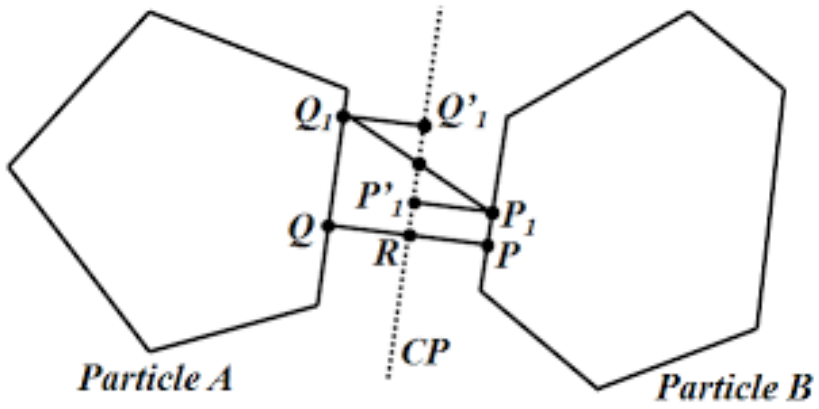
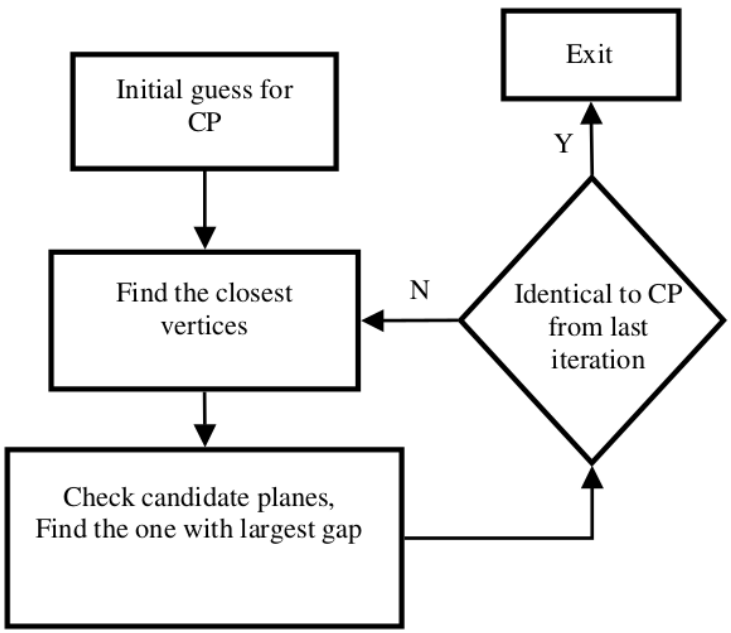


Figure 4: Common-plane

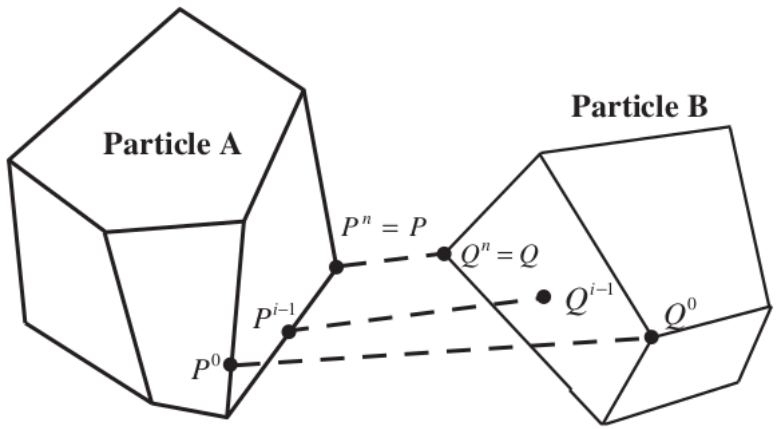
Fast Common-Plane

- En 2004 Nezami [1] propuso una nueva propuesta para el cálculo del CP y se llamó Fast Common-Plane (FCP).
- Con esta nueva propuesta mejoró el orden del algoritmo por ello la eficiencia.



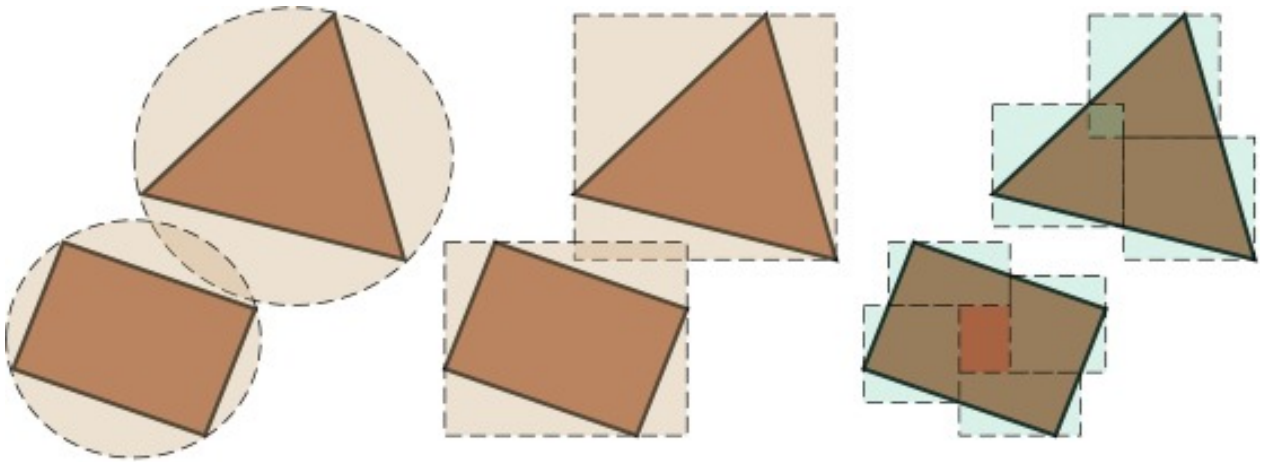
Shortest Link Method

- Nezami en 2006 [2] propuso otro algoritmo ocupando como base FCP.
- En base a resultados expuestos por Nezami [2], el SLM es 17 veces más rápido que otros algoritmos convencionales.



Multi-shell Contact Detection

- Fue desarrollado por Zhuang el 2014 [3].
- MSC como método tanto de detección de vecindario como detección de contacto es uno de los más eficiente, según concluye Zhuang.



(a) NBS (b) DESS (c) MSC

Objetivo

Objetivo General

- Desarrollar una nueva propuesta que detecte colisiones entre poliedros.

Objetivo Específico

- Desarrollar una representación geométrica para los cuerpos rígidos.
- Desarrollar un algoritmo de detección de contactos para vértices y aristas.
- Desarrollar un algoritmo de detección de contacto entre cuerpos rígidos de orden no mayor a $\mathcal{O}(N)$, donde N es la cantidad de cuerpos de la simulación.

Metodología

Herramientas

- El desarrollo de los algoritmos se utilizó python.
- Se llevó un control de versiones con la herramienta Github.

Metodología

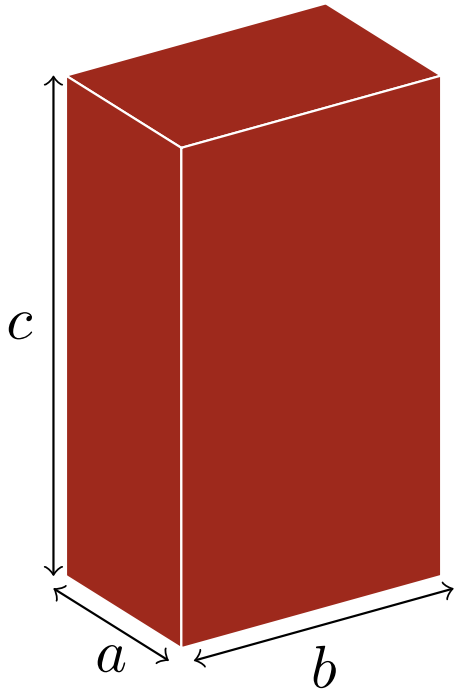


Figure 5: Un cuerpo representado por un poliedro con dimensiones a, b y c.

Metodología

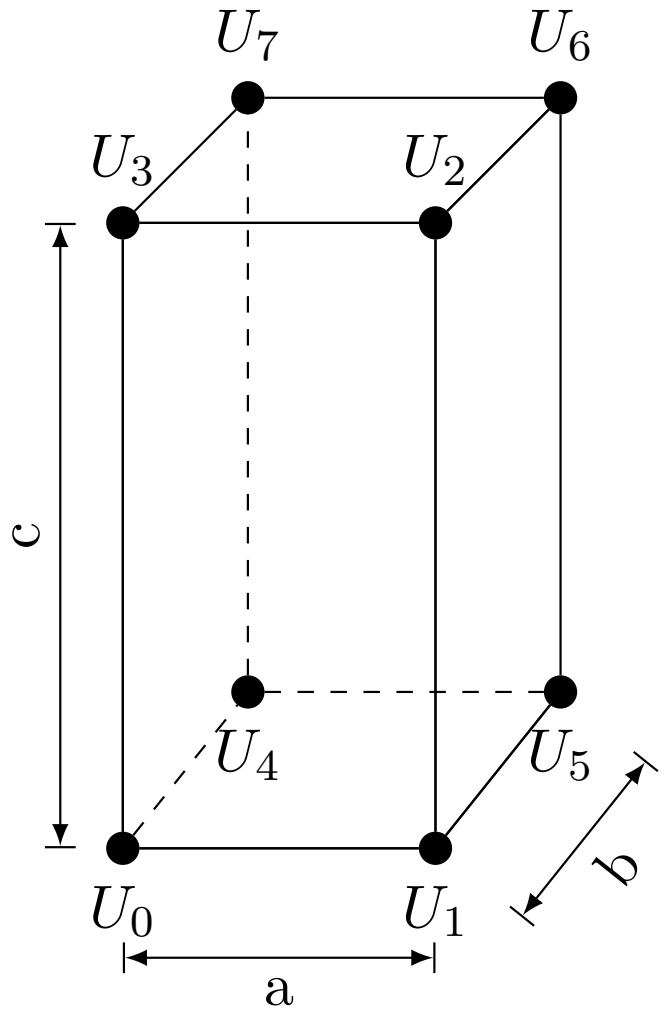


Figure 6: Un poliedro U que se utilizará como referencia para análisis

Metodología

Ecuaciones para un vértice

$$V_j^* := V_j - U_0, j = 0 : 7 \tag{1}$$

Llevar el vértice al nuevo plano de prueba

$$V_j^* = \begin{Bmatrix} V_{1j}^* \\ V_{2j}^* \\ V_{3j}^* \end{Bmatrix} = \begin{Bmatrix} \alpha_{1j} \\ \alpha_{2j} \\ \alpha_{3j} \end{Bmatrix} \tag{2}$$

Determinación de los α's.

$$V_j \in U \Leftrightarrow 0 \leq \alpha_{1j} \leq a \wedge 0 \leq \alpha_{2j} \leq b \wedge 0 \leq \alpha_{3j} \leq c \tag{3}$$

Determinar si el vector se encuentra dentro del poliedro.

Metodología

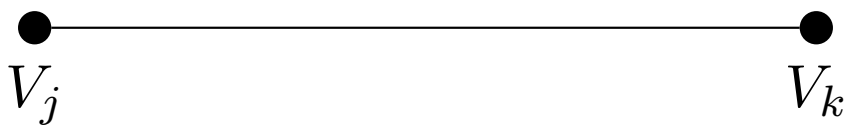


Figure 7: Una arista con inicio en V_j y final en V_k, tomada de un poliedro V.

Metodología

Ecuaciones para una arista

$$r^*(s) = (1-s)V_j^* + sV_k^*, 0 \leq s \leq 1. \tag{4}$$

Ecuación general que determina el segmento que se encuentra en contacto con el poliedro.

$$S = \frac{(\rho_i - V_{ji}^*)}{V_{ki}^* - V_{ji}^*}; i = 0:2 \tag{5}$$

Despejando la variable 'S' de la ecuación general se obtiene el segmento contenido en el poliedro.

Metodología

Ecuaciones para una arista

$$S = \begin{Bmatrix} S_{min0} & S_{max0} \\ S_{min1} & S_{max1} \\ S_{min2} & S_{max2} \end{Bmatrix} \tag{6}$$

La intersección de estos tres intervalos es el segmento que esta en el poliedro.

Metodología

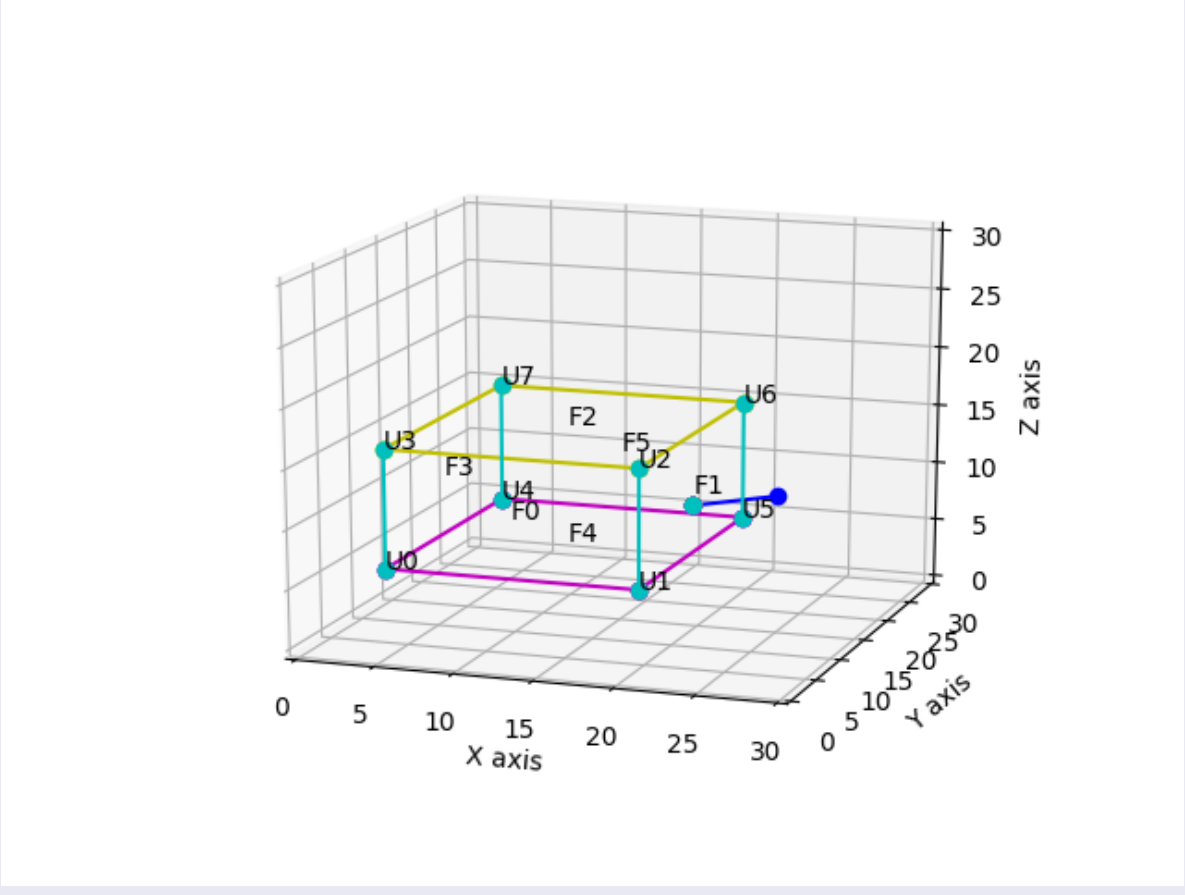
Ecuaciones para la intersección de una arista

$$J := \{s \in [0,1] | 0 \leq \rho_1(s) \leq a\} \cap \{0 \leq \rho_2(s) \leq b\} \cap \{0 \leq \rho_3(s) \leq c\} \tag{7}$$

Ecuación de intervalos que entrega la intersección

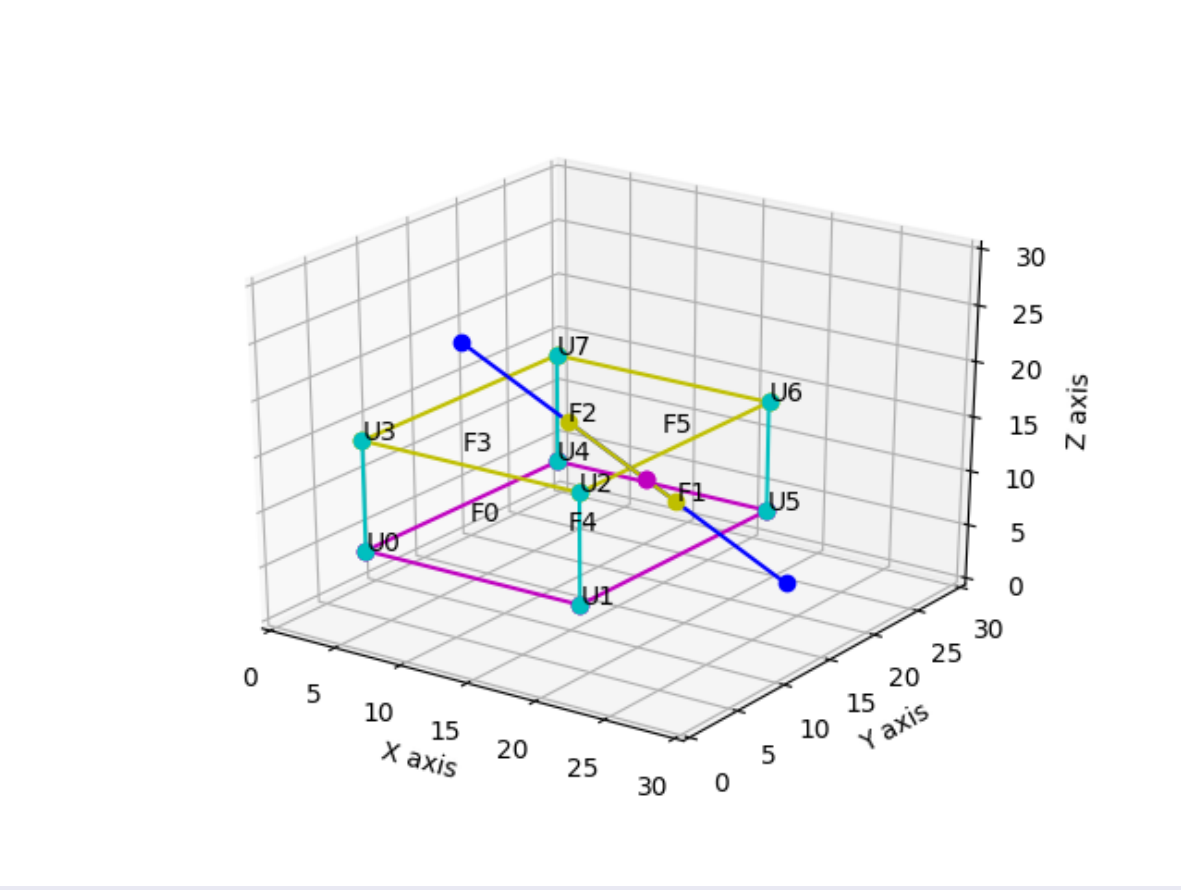
Metodología

Obtención de la cara por donde ingresa un vértice



Metodología

Obtención de las caras por donde ingresa una arista



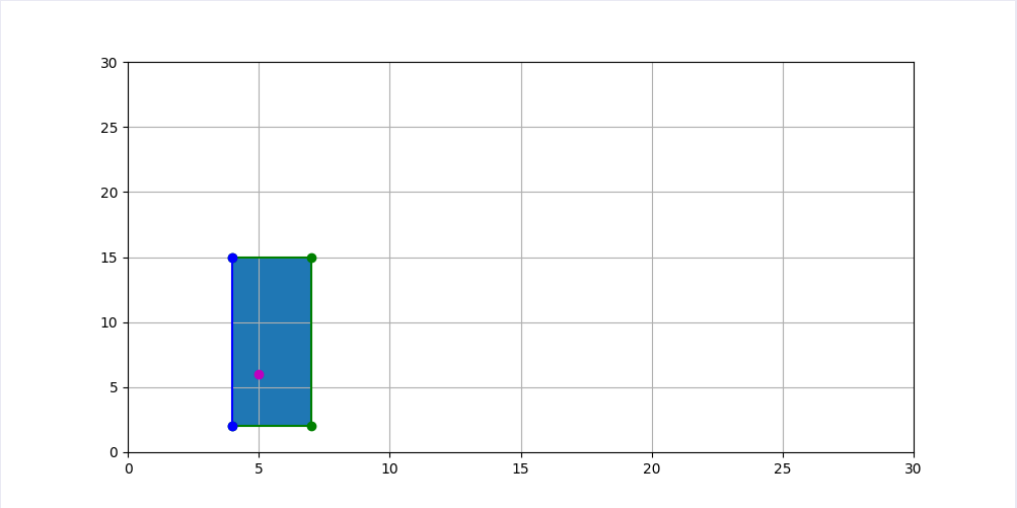
Resultados

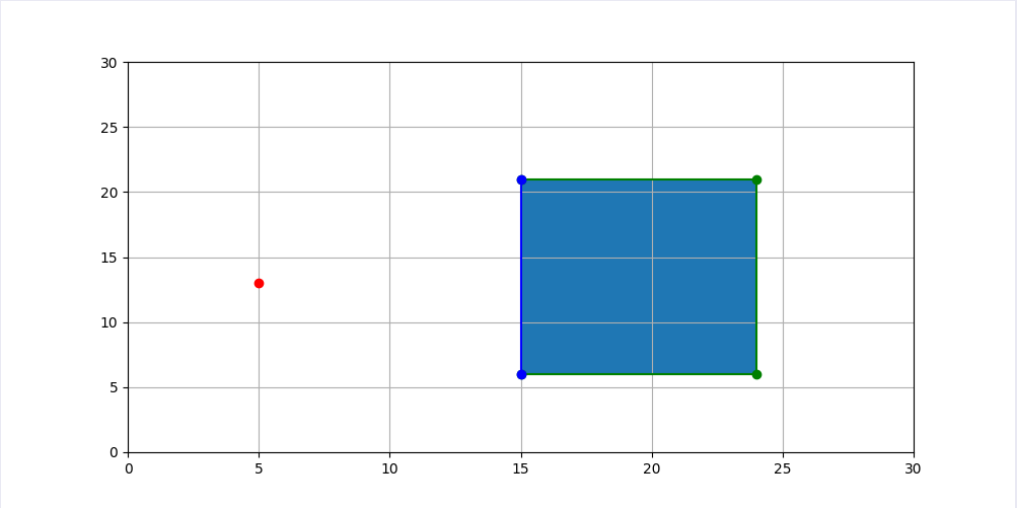
Prueba 2-D

```
def pointdetection(U0, a, b, c, point):
    B = getB()
    V0 = point[0] - U0[0]
    V1 = point[1] - U0[1]
    V2 = point[2] - U0[2]

    V = np.array([[V0], [V1], [V2]])

    alpha = np.linalg.solve(B, V)

    resultx, resulty, resultz = False, False, False

    if (0 <= alpha[0] <= a):
        resultx = True
    if (0 <= alpha[1] <= b):
        resulty = True
    if (0 <= alpha[2] <= c):
        resultz = True
    return resultx and resulty and resultz
```

Resultados

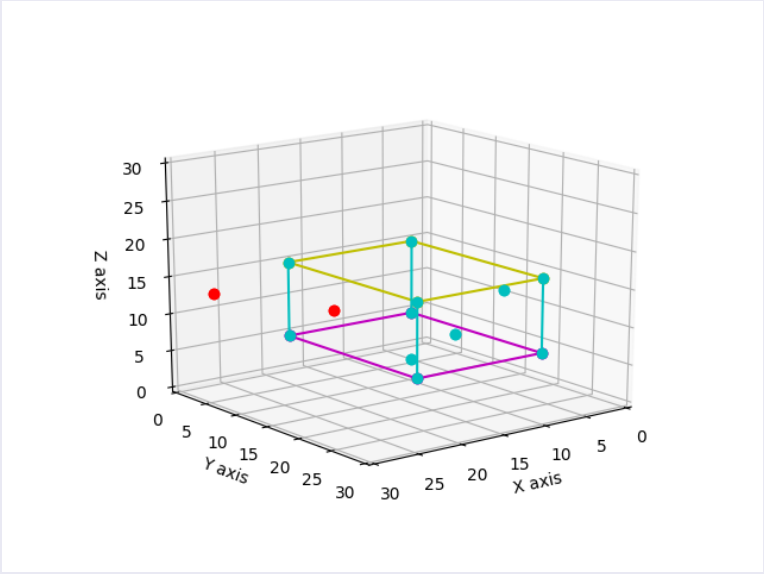
Prueba 2-D

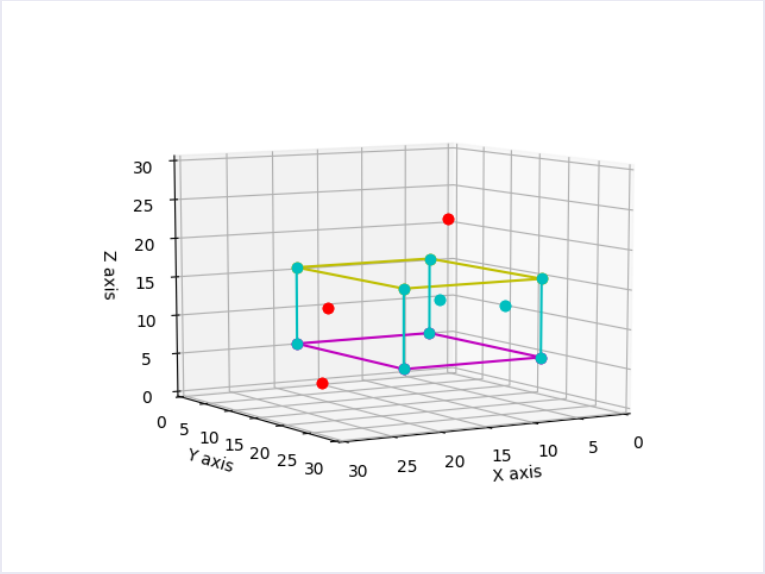




Resultados

Prueba 3-D





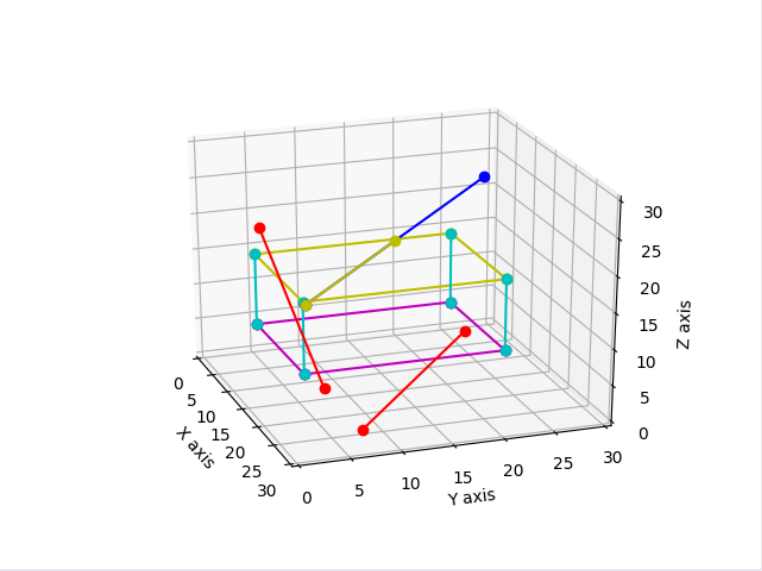
Resultados

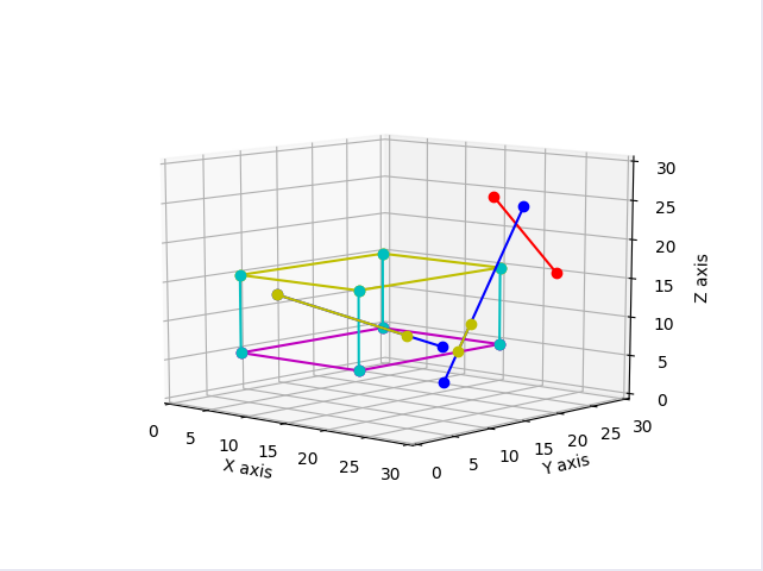
Pruebas con más vértices al mismo tiempo

```
(venv) yerkozec@yerkozec-notebook:~/Desktop/pt/ContactDetection$ python3 testeopunto.py
usar datos random para generar figura y pruebas(True/False)?
True
cuantas pruebas?
100
tiempo de deteccion de punto: 0.012374401092529297
numero de puntos detectados dentro:  14
numero de puntos detectados fuera:  86
(venv) yerkozec@yerkozec-notebook:~/Desktop/pt/ContactDetection$ python3 testeopunto.py
usar datos random para generar figura y pruebas(True/False)?
True
cuantas pruebas?
1000
tiempo de deteccion de punto: 0.06482911109924316
numero de puntos detectados dentro:  113
numero de puntos detectados fuera:  887
(venv) yerkozec@yerkozec-notebook:~/Desktop/pt/ContactDetection$ python3 testeopunto.py
usar datos random para generar figura y pruebas(True/False)?
True
cuantas pruebas?
10000
tiempo de deteccion de punto: 0.34250688552856445
numero de puntos detectados dentro:  1255
numero de puntos detectados fuera:  8745
(venv) yerkozec@yerkozec-notebook:~/Desktop/pt/ContactDetection$ python3 testeopunto.py
usar datos random para generar figura y pruebas(True/False)?
True
cuantas pruebas?
50000
tiempo de deteccion de punto: 1.5083446502685547
numero de puntos detectados dentro:  6420
numero de puntos detectados fuera:  43580
```

Resultados

Prueba 3-D





Resultados

Pruebas con más aristas al mismo tiempo

```
(venv) yerkozec@yerkozec-notebook:~/Desktop/pt/ContactDetection$ python3 testeoarista.py
usar datos random para generar figura y pruebas(True/False)?
True
Cuantas pruebas?
100
tiempo de deteccion de arista:  0.03959774971008301
numero de aristas detectadas dentro:  47
numero de aristas detectadas fuera:  53
(venv) yerkozec@yerkozec-notebook:~/Desktop/pt/ContactDetection$ python3 testeoarista.py
usar datos random para generar figura y pruebas(True/False)?
True
Cuantas pruebas?
1000
tiempo de deteccion de arista:  0.14173412322998047
numero de aristas detectadas dentro:  532
numero de aristas detectadas fuera:  468
(venv) yerkozec@yerkozec-notebook:~/Desktop/pt/ContactDetection$ python3 testeoarista.py
usar datos random para generar figura y pruebas(True/False)?
True
Cuantas pruebas?
10000
tiempo de deteccion de arista:  1.1369974613189697
numero de aristas detectadas dentro:  5424
numero de aristas detectadas fuera:  4576
(venv) yerkozec@yerkozec-notebook:~/Desktop/pt/ContactDetection$ python3 testeoarista.py
usar datos random para generar figura y pruebas(True/False)?
True
Cuantas pruebas?
25000
tiempo de deteccion de arista:  2.7897703647613525
numero de aristas detectadas dentro:  13527
numero de aristas detectadas fuera:  11473
```

Resultados

Detección de cara para un vector

```
(venv) yerkozec@yerkozec-notebook:~/Desktop/pt/ContactDetection$ python3 testeopunto.py < ./pruebas/1.in
usar datos random para generar figura y pruebas(True/False)?
False
[[1, 0, 0], array([20., 14., 10.])]
```

Resultados

Detección de cara para una arista

```
(venv) yerkozec@yerkozec-notebook:~/Desktop/pt/ContactDetection$ python3 testeoarista.py < ./pruebas/2.in
usar datos random para generar figura y pruebas(True/False)?
False
Si entra por la cara: [0, -1, 0]
Sf entra por la cara: [0, 0, -1]
Las caras por la que cruza la arista: (f0, f4)
la arista que componen las caras e0
punto mas cercano: [17.     7.096 10.472]
```

Conclusión

- Se logró representar un cuerpo rígido de manera geométrica.
- En esta fase del desarrollo de la propuesta se logró detectar si un vértice y una arista están en contacto con el poliedro.

Trabajos futuros

- Simulación de contacto entre poliedros a gran escala.

Referencias


Nezami G. Erfan.
A fast contact detection algorithm for 3-d discrete element method.
Elsevier, 31:575 – 587, 2004.


Nezami G. Erfan.
Shortest link method for contact detection in discrete element method.
Wiley InterScience, 30:783 – 801, 2006.


Zhuang Z.
A multi-shell cover algorithm for contact detection in the three dimensional discontinuous deformation analysis.
Elsevier, 72:136 – 149, 2014.

Nueva propuesta para detección de contacto entre poliedros a gran escala

Yerko Zec

Universidad Andrés Bello®
Conectar • Innovar • Liderar

January 27, 2020