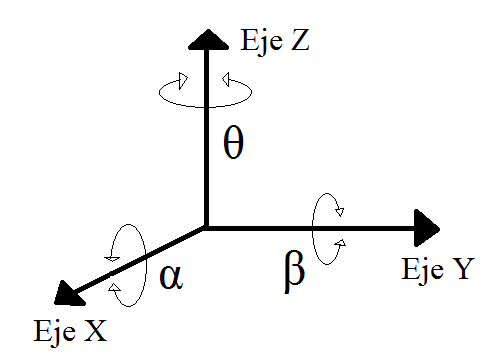
Ing. Computación Inteligente

**Graficación**

*Prof. Alexis Edmundo Gallegos Acosta*

**Práctica 9**. Transformación de rotación



**Objetivo:**

En esta práctica, los estudiantes aprenderán a aplicar las transformaciones de rotación en el plano 2D y en el espacio 3D, ya sea con el centro de rotación en el origen o en otro lugar. Utilizarán matrices de rotación para realizar las rotaciones y podrán observar cómo cambian las coordenadas de los puntos tras la rotación.

**Instrucciones:**

Realizar una aplicación para realizar transformaciones de rotación 2D y 3D:

* Entrada:
  + Un conjunto de puntos (x, y) o (x, y, z), según el caso, para formar una figura regular. (Ejemplos: casa, coche, un árbol, animales, frutas, etc.)
  + Ángulo de rotación
  + Un centro de rotación (x, y) o (x, y, z)
* Proceso:
  + Definir una rutina en la que se observe claramente la definición de la matriz de rotación y el proceso de la transformación de rotación.
* Salida:
  + Gráfica (OpenCV o Matplotlib) en la que se observe la figura y la aplicación de la transformación de rotación 2D y 3D.

**Entregables:**

* Código fuente con la aplicación descrita anteriormente.
* Informe con lo siguiente:
  + Datos del alumno: Nombre, ID, carrera, materia, semestre.
  + Introducción.
  + Descripción de los métodos utilizados para implementar las rotaciones.
  + Ejemplo 2D de puntos y coordenadas antes y después de la rotación.
  + Ejemplo 3D de puntos y coordenadas antes y después de la rotación.
  + Conclusión personal sobre lo aprendido.
  + Referencia y bibliografía en formato APA o IEEE.

**Modalidad:** Individual.

**NOTA:** Cada alumno debe realizar la práctica de forma individual y demostrar su comprensión y habilidades propias. Si se sorprenden trabajos idénticos en código fuente o en el informe causará la anulación del trabajo con calificación de 0 para todos los involucrados.

**Calificaciones:**

* Calificación máxima de 10, si se entrega en tiempo y forma.
* Calificación máxima de 8, si se entrega hasta 2 días hábiles después de la fecha establecida.
  + **NO** Se recibirán trabajos después de este periodo extra causando una calificación de 0.
* Cualquier otro caso, es necesaria la entrega de un justificante (de salud, académico o laboral) para ajustar tiempo de entrega y calificación máxima.

# **Introducción**

Las transformaciones de rotación son una parte fundamental de la geometría en dos y tres dimensiones, permitiendo modificar la orientación de figuras en el plano y en el espacio. Estas transformaciones encuentran aplicaciones en diversos campos, desde la ingeniería y la informática gráfica hasta la física y la biología.

En esta práctica, exploraremos cómo aplicar rotaciones a figuras geométricas utilizando matrices de rotación. Estas matrices son herramientas algebraicas que nos permiten describir de manera compacta y eficiente cómo los puntos de una figura se desplazan alrededor de un punto central bajo una rotación dada, detallando el proceso de implementación, exploraremos el tema junto con ejemplos concretos de su aplicación y conclusiones sobre el aprendizaje obtenido.

# **Descripción de los métodos utilizados para implementar las rotaciones**

Las transformaciones de rotación en 2D y 3D se pueden realizar mediante matrices de rotación, las cuales aplican una rotación a un punto alrededor de un centro dado. En el caso del plano 2D, la matriz de rotación se define como:

Donde es el ángulo de rotación.

Por otro lado, para realizar la rotación en el espacio 3D, se deben considerar matrices de rotación específicas para cada eje (x, y, z). Estas matrices se construyen utilizando las funciones trigonométricas seno y coseno para aplicar la rotación alrededor del eje correspondiente, dando como operadores, entonces, los siguientes casos:

Para el ángulo x:

Para el ángulo y:

Para el ángulo z:

Estos conceptos son implementados en el código de manera sencilla, donde tenemos precisamente sus respectivas funciones para el cálculo de cada caso:

**Rotación 2D**

def *rotacion\_2d*(punto, angulo, centro):

    # *Transformar el ángulo a radianes*

    theta = np.radians(angulo)

    # *Calcular las coordenadas relativas al centro de rotación*

    x\_rel = punto[0] - centro[0]

    y\_rel = punto[1] - centro[1]

    # *Aplicar la matriz de rotación*

    x\_rot = x\_rel \* np.cos(theta) - y\_rel \* np.sin(theta)

    y\_rot = x\_rel \* np.sin(theta) + y\_rel \* np.cos(theta)

    # *Regresar a las coordenadas originales*

    x\_rot += centro[0]

    y\_rot += centro[1]

*return* [x\_rot, y\_rot]

En esta función obtendremos tres parámetros, el punto del vector de puntos de nuestra figura con el que se realizarán los cálculo, el ángulo de rotación y el centro del este. Con estos datos, realizamos los siguientes pasos:

* Inicialmente, el ángulo de rotación, dado en grados, **se convierte a radianes** utilizando la función np.radians de la biblioteca NumPy. Esto es necesario porque las funciones trigonométricas en Python trabajan con ángulos en radianes.
* Después, se calculan las **coordenadas del punto relativas** al centro de rotación. Esto se hace restando las coordenadas del centro de rotación de las coordenadas del punto original.
* Se aplican las fórmulas de la **matriz de rotación para 2D**. Para la coordenada x del punto rotado se calcula como el producto de la coordenada x relativa por el coseno del ángulo, menos la coordenada y relativa por el seno del ángulo. La coordenada y del punto rotado se calcula de manera similar, pero sumando el producto de la coordenada x relativa por el seno del ángulo y la coordenada y relativa por el coseno del ángulo.
* Finalmente, se agregan las coordenadas del centro de rotación a las coordenadas rotadas, para obtener las **coordenadas finales** del punto rotado, las cuales se devuelven para ser almacenados y graficados.

**Rotación 3D**

def *rotacion\_3d*(punto, angulo, centro, eje):

    # *Transformar el ángulo a radianes*

    theta = np.radians(angulo)

    # *Matriz de rotación según el eje seleccionado*

*if* eje == 'x':

        matriz\_rotacion = np.array([[1, 0, 0],

                                     [0, np.cos(theta), -np.sin(theta)],

                                     [0, np.sin(theta), np.cos(theta)]])

*elif* eje == 'y':

        matriz\_rotacion = np.array([[np.cos(theta), 0, np.sin(theta)],

                                     [0, 1, 0],

                                     [-np.sin(theta), 0, np.cos(theta)]])

*elif* eje == 'z':

        matriz\_rotacion = np.array([[np.cos(theta), -np.sin(theta), 0],

                                     [np.sin(theta), np.cos(theta), 0],

                                     [0, 0, 1]])

*else*:

*raise* ValueError("El eje debe ser 'x', 'y' o 'z'")

    # *Calcular las coordenadas relativas al centro de rotación*

    p\_rel = np.array(punto) - np.array(centro)

    # *Aplicar la matriz de rotación*

    p\_rotado = np.dot(matriz\_rotacion, p\_rel)

    # *Regresar a las coordenadas originales*

    p\_rotado += np.array(centro)

*return* p\_rotado

En este caso, los parámetros funcionan de la misma manera, con la excepción de que contamos con el eje con el que el usuario desee trabajar; siendo los siguientes pasos similares:

* Se transforma el ángulo de grados a **radianes**. Además, se selecciona la matriz de rotación adecuada según el **eje de rotación especificado por el usuario** ('x', 'y' o 'z').
* Se calculan las **coordenadas del punto relativas** al centro de rotación, de manera similar a la función de rotación 2D.
* Se aplica la matriz de rotación seleccionada al punto relativo utilizando la función np.dot de NumPy, que realiza el **producto punto** entre la matriz de rotación y las coordenadas relativas del punto.
* Y finalmente, se **agregan las coordenadas del centro de rotación a las coordenadas rotadas**, de manera similar a la función de dos dimensiones, para obtener las **coordenadas finales** del punto rotado en el espacio 3D.

# **Ejemplo 2D de puntos y coordenadas antes y después de la rotación**

Para este caso, fue formada una figura conformada de cinco puntos en distintas coordenadas del plano, los cuales forma una figura de **una** **casa:**

puntos\_2d = [[1, 1], [5, 1], [5, 4], [3, 6], [1, 4]]

Unidas en código de la siguiente forma:

**A screenshot of a computer

Description automatically generated**

Los cuales se trabajan de la siguiente manera al ejecutar nuestro programa:

En consola, será impreso el siguiente mensaje, elegimos la primera opción

A blue screen with white text

Description automatically generated

Posteriormente nos pedirá el ángulo de rotación, para este ejemplo utilizamos 45:



Y por último, los puntos del centro de rotación que, en este caso, serán para ambos ejes X, Y, utilizando como ejemplo en ambos valores el punto 1:

A blue background with white text

Description automatically generated

Tras ingresar los datos, será mostrado en pantalla tanto la figura original como la rotada:

A screenshot of a graph

Description automatically generated

Donde tenemos entonces ambos casos de ser que sean graficados independientemente del otro:

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Como **elemento extra**, es posible igualmente visualizar los valores de las coordenadas correspondientes, lo cual se consigue simplemente eliminando los comentarios del siguiente bloque de código de la función *plot\_2d*:

*for* i, punto *in* enumerate(p):

plt.text(punto[0], punto[1], f'({punto[0]:.3f}, {punto[1]:.3f})', fontsize=8, ha='right')

Dando un resultado como el siguiente:

A screenshot of a graph

Description automatically generated

# **Ejemplo 3D de puntos y coordenadas antes y después de la rotación**

Para este otro caso, trabajamos con una casa igualmente con la diferencia de que ahora será trazado en un plano 3D, por lo que serán requeridos no cinco sino diez en total:

puntos\_3d = pared\_3d(0) + pared\_3d(4)

Obtenido de la función pared\_3d, lo cual nos da la suma de las dos paredes de nuestra casa (ambas iguales, pero una posicionada más lejos en el eje y que la otra), es decir, todos los puntos que lo conforma:

def *pared\_3d*(y):

*return* [[1, y, 1], [5, y, 1], [5, y, 4], [3, y, 6], [1, y, 4]]

Dándonos el siguiente resultado de su unión:

A screenshot of a computer

Description automatically generated

El proceso es similar; comenzando por la consola, seleccionamos la segunda opción inicialmente:

A blue screen with white text

Description automatically generated

Damos un ángulo de rotación, en este caso 30:



Los puntos del centro de rotación para los tres ejes X, Y, Z, para en este caso 0, 1, 0, respectivamente:

A blue screen with white text

Description automatically generated

Y por último, el eje base de rotación, x para este caso:



Obteniendo entonces un siguiente resultado:

A screenshot of a computer

Description automatically generated

Teniendo como modelos independientes:

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Al igual que usando los mismos datos pero para diferentes ejes base (y, z, respectivamente):

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Por último, nuevamente como **elemento extra**, es posible ver los valores de los puntos al eliminar los comentarios de la función *plot\_3d*:

*for* i, punto *in* enumerate(p):

ax.text(punto[0], punto[1], punto[2], f'({punto[0]:.3f}, {punto[1]:.3f}, {punto[2]:.3f})', fontsize=8, ha='right')

Ilustrando, entonces, lo siguiente (es posible apreciarlos mediante su ejecución, siendo la opción recomendable de ser que se deseen visualizar):

A screenshot of a computer

Description automatically generated

# **Conclusión personal sobre lo aprendido**

Esta práctica me permitió comprender en profundidad el concepto de transformaciones de rotación en 2D y 3D, así como su implementación utilizando matrices, especialmente la parte donde la posibilidad de elegir el eje de rotación en el espacio 3D, el amplió mi comprensión de las transformaciones tridimensionales siendo que una depende ciertamente de cada caso que se desee implementar. De igual forma, desarrolle habilidades en programación y visualización de datos, elementos y funciones que pueden ser aplicadas en diversos campos de estudio y proyectos posteriores.

# **Referencia y bibliografía en formato APA o IEEE**

* Silva, C. (2012, November 7). Geometría, cinemática y dinámica robótica [Slide show]. SlideShare. <https://es.slideshare.net/camiloasilva3/rotacin-matricial>
* Rzeronte. (2021, March 11). Movimiento y rotación en el espacio 3D. Brakeza3D. <https://brakeza.com/movimiento-y-rotacion-en-el-espacio-3d/>
* colaboradores de Wikipedia. (2024, February 16). Matriz de rotación. Wikipedia, La Enciclopedia Libre. <https://es.wikipedia.org/wiki/Matriz_de_rotaci%C3%B3n>