МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ

УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТИХНІЧНИЙ ІНСТИТУТ ІМ.ІГОРЯ

СІКОРСЬКОГО»

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Лабораторна робота №2

«Геометричні перетворення координат у просторі»

Виконав:

Студент 2 курсу

Групи ФІ-21

Грунда Ярослав

ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

1. Ознайомитись з теоретичними відомостями до лабораторної роботи. Дослідити кожне перетворення (перетворення, масштабування, поворот, відображення тощо) та їхні комбінації просторі. Ознайомитись з матеріалом, який віднситься до проектування об’єктів на площину.

2. Задати куб через його вершини. Здійснити масштабування куба (збільшення,зменшення у кілька разів). Записати відповідну матрицю.

3. Отриманий результат з попереднього пункту симетрично відобразити відносно початку координат, відносно однієї з координатних площин у тривимірній декартовій системі координат (простір).

4. Задати пряму як у прикладі 1 даної лабораторної роботи. Здійснити поворот куба відносно прямої на деякий кут φ. Вказати перетворення, які необхідні для одержання такого результату.

5. Задати площину, як у прикладі 2, здійснити симетрію куба відносно площини, вказати всі математичні перетворення, показати кінцевий результат, який відповідає вимогам задачі. Записати відповідні математичні перетворення.

6. (для студентів, які навчаються на кафедрі ММАД) Запропонувати інший набір перетворень для отримання результатів, які наведені у прикладі 2 даної лабораторної роботи. Тобто здійснити інші перетворення за загальним підходом (симетрія, поворот, перенесення), показати їх математично. 7. (для студентів, які навчаються на кафедрі ММАД) Запропонувати інший набір перетворень для отримання результатів, які наведені у прикладі 2 даної лабораторної роботи. Тобто здійснити інші перетворення за загальним підходом (симетрія, поворот, перенесення), показати їх математично.

8. (для студентів, які навчаються на кафедрі ММАД) Навести алгоритм з математичними перетвореннями для побудови симетрії відносно площини на основі знань з аналітичної геометрії. Порівняти результати (програмно)

9. До пунктів 2–8 розробити програмне забезпечення (студентам кафедри ММЗІ для пунктів 2–5). Навести результат виконання кожного пункту.

10. (для всіх студентів) Взяти той самий куб, який був поданий у попередніх пунктах. Здійснити ортогональне проектування даного кубу на площину загального виду, яка не паралельна жодній з координатних площин. Вказати відповідні математичні перетворення, розробити відповідне програмне забезпечення.

Хід роботи

Для початку создамо класи куба і лінії:

class Cube:

    def \_\_init\_\_(self, vertices):

        self.vertices = vertices

    def plot(self, ax, color = 'r', label = 'cube'):

        # З'єднуємо вершини для створення граней куба

        edges = np.array([

            [self.vertices[0,:-1], self.vertices[1,:-1], self.vertices[2,:-1], self.vertices[3,:-1]],

            [self.vertices[4,:-1], self.vertices[5,:-1], self.vertices[6,:-1], self.vertices[7,:-1]],

            [self.vertices[0,:-1], self.vertices[1,:-1], self.vertices[5,:-1], self.vertices[4,:-1]],

            [self.vertices[2,:-1], self.vertices[3,:-1], self.vertices[7,:-1], self.vertices[6,:-1]],

            [self.vertices[1,:-1], self.vertices[2,:-1], self.vertices[6,:-1], self.vertices[5,:-1]],

            [self.vertices[0,:-1], self.vertices[3,:-1], self.vertices[7,:-1], self.vertices[4,:-1]]

        ])

        # Відображення осей координат

        ax.quiver(-8, 0, 0, 16, 0, 0, color='k', label='\_X-axis')

        ax.quiver(0, -8, 0, 0, 16, 0, color='k', label='\_Y-axis')

        ax.quiver(0, 0, -8, 0, 0, 16, color='k', label='\_Z-axis')

        # Відображення

        poly3d = Poly3DCollection(edges, facecolors='cyan', linewidths=1, edgecolors=color, alpha=0)

        ax.add\_collection3d(poly3d)

        ax.add\_collection3d(Line3DCollection([], colors=color, linewidths=1, label=label))

        ax.set\_xlabel('X')

        ax.set\_ylabel('Y')

        ax.set\_zlabel('Z')

        ax.set\_xlim([-8, 8])

        ax.set\_ylim([-8, 8])

        ax.set\_zlim([-8, 8])

class Line:

    def \_\_init\_\_(self, points):

        self.points = points

    def plot(self, ax, label, color):

        ax.plot(self.points[:, 0], self.points[:, 1], self.points[:, 2], label = '\_'+label, color = color)

        # Відображення осей координат

        ax.quiver(-8, 0, 0, 16, 0, 0, color='k', label='\_X-axis')

        ax.quiver(0, -8, 0, 0, 16, 0, color='k', label='\_Y-axis')

        ax.quiver(0, 0, -8, 0, 0, 16, color='k', label='\_Z-axis')

        ax.set\_xlabel('X')

        ax.set\_ylabel('Y')

        ax.set\_zlabel('Z')

        # Задаємо діапазон значень та позначки на вісях

        ax.set\_xlim([-8, 8])

        ax.set\_ylim([-8, 8])

        ax.set\_zlim([-8, 8])

2, 3 Задати куб через його вершини. Здійснити масштабування куба (збільшення,зменшення у кілька разів). Записати відповідну матрицю. Отриманий результат з попереднього пункту симетрично відобразити відносно початку координат, відносно однієї з координатних площин у тривимірній декартовій системі координат (простір).

cube\_vertices = np.array([

        [4, 6, 2],

        [6, 6, 2],

        [6, 8, 2],

        [4, 8, 2],

        [4, 6, 4],

        [6, 6, 4],

        [6, 8, 4],

        [4, 8, 4]])

Додаємо одиниці в кінець:

cube\_vertices\_with\_ones = np.hstack([cube\_vertices,np.ones((cube\_vertices.shape[0], 1))])

Знаходимо центр мас куба:

center\_x = np.mean(cube\_vertices[:, 0])

center\_y = np.mean(cube\_vertices[:, 1])

center\_z = np.mean(cube\_vertices[:, 2])

Матриця здвигу в центр:

matrix\_move\_to\_center = np.array([[1, 0, 0, -center\_x],

                                [0, 1, 0, -center\_y],

                                [0, 0, 1, -center\_z],

                                [0, 0, 0, 1]])

Матриця здвигу від центру

matrix\_move\_from\_center = np.array([[1, 0, 0, center\_x],

                                [0, 1, 0, center\_y],

                                [0, 0, 1, center\_z],

                                [0, 0, 0, 1]])

Матриця збільшення

matrix\_increment = np.array([[1.5, 0, 0, 0],

                                [0, 1.5, 0, 0],

                                [0, 0, 1.5, 0],

                                [0, 0, 0, 1]])

Матриця зменшення

matrix\_decrease = np.array([[0.5, 0, 0, 0],

                                [0, 0.5, 0, 0],

                                [0, 0, 0.5, 0],

                                [0, 0, 0, 1]])

Матриця симетричного відображення відного центру координат

matrix\_sym\_start\_coord = np.array([[-1, 0, 0, 0],

                                        [0, -1, 0, 0],

                                        [0, 0, -1, 0],

                                        [0, 0, 0, 1]])

Матриця саметричного відображення відносно XOZ

matrix\_sym\_one\_coord\_plane = np.array([[1, 0, 0, 0],

                                        [0, -1, 0, 0],

                                        [0, 0, 1, 0],

                                        [0, 0, 0, 1]])

В результаті маємо:

Изображение выглядит как текст, диаграмма, линия, дизайн

Автоматически созданное описание

4. Задати пряму як у прикладі 1 даної лабораторної роботи. Здійснити поворот

куба відносно прямої на деякий кут φ. Вказати перетворення, які необхідні

для одержання такого результату.

Точка і вектор

point\_on\_line = np.array([1, 2, 3])

direction\_vector = np.array([2, 1, 3])

direction\_vector\_with\_one = np.array([2, 1, 3, 1])

Одиничний вектор

magnitude = np.linalg.norm(direction\_vector)

direction\_vector = direction\_vector / magnitude

direction\_vector\_with\_one = np.append(direction\_vector, 1)

Деякі координати

l, m, n = direction\_vector[0], direction\_vector[1], direction\_vector[2]

a, b, c = point\_on\_line[0], point\_on\_line[1], point\_on\_line[2]

Косинуси та синуси для одного з повороту

cos\_psi = n/math.sqrt((m\*\*2+n\*\*2))

sin\_psi = m/math.sqrt((m\*\*2+n\*\*2))

Лінія

line\_original = np.array([point\_on\_line + t \* direction\_vector for t in np.linspace(-8, 8, 100)])

line\_original = np.hstack([line\_original, np.ones((line\_original.shape[0], 1))])

Рух до центру

matrix\_move\_to\_center = np.array([[ 1, 0, 0, -a],

                            [ 0, 1, 0, -b],

                            [ 0, 0, 1, -c],

                            [ 0, 0, 0, 1]])

Поворот до XOZ

matrix\_XOZ = np.array([[ 1, 0, 0, 0,],

                    [ 0, cos\_psi, -sin\_psi, 0],

                    [ 0, sin\_psi, cos\_psi, 0],

                    [ 0, 0, 0, 1]])

Поворот до Z

matrix\_Z = np.array([[ d, 0, -l, 0,],

                    [ 0, 1, 0, 0],

                    [ l, 0, d, 0],

                    [ 0, 0, 0, 1]])

Поворот на деякий кут

phi = np.pi/2

cos\_phi = np.cos(phi)

sin\_phi = np.sin(phi)

matrix\_phi = np.array([[ cos\_phi, -sin\_phi, 0, 0,],

                    [ sin\_phi, cos\_phi, 0, 0],

                    [ 0, 0, 1, 0],

                    [ 0, 0, 0, 1]])

Поворот від Z

matrix\_Z\_rev = matrix\_Z.T

Поворот від XOZ

matrix\_XOZ\_rev = matrix\_XOZ.T

Рух від центру

matrix\_move\_from\_center = np.array([[ 1, 0, 0, a],

                            [ 0, 1, 0, b],

                            [ 0, 0, 1, c],

                            [ 0, 0, 0, 1]])

Изображение выглядит как текст, диаграмма, дизайн

Автоматически созданное описание

6. (для студентів, які навчаються на кафедрі ММАД) Запропонувати інший

набір перетворень для отримання результатів, які наведені у прикладі 2 даної

лабораторної роботи. Тобто здійснити інші перетворення за загальним

підходом (симетрія, поворот, перенесення), показати їх математично.

Дії:

cos\_psi = n/math.sqrt((l\*\*2+n\*\*2))

sin\_psi = l/math.sqrt((l\*\*2+n\*\*2))

# Moving to center

matrix\_move\_to\_center = np.array([[ 1, 0, 0, -a],

                            [ 0, 1, 0, -b],

                            [ 0, 0, 1, -c],

                            [ 0, 0, 0, 1]])

# Rotate to ZOY

matrix\_ZOY = np.array([[cos\_psi, 0, -sin\_psi, 0,],

                    [0, 1, 0, 0],

                    [sin\_psi, 0, cos\_psi, 0],

                    [0, 0, 0, 1]])

l,d = dir\_vec\_moved\_to\_ZOY[1], dir\_vec\_moved\_to\_ZOY[2]

# Rotate to Y

matrix\_Y = np.array([[ 1, 0, 0, 0,],

                    [ 0, l, d, 0],

                    [ 0, -d, l, 0],

                    [ 0, 0, 0, 1]])

# Rotate on given angle

phi = np.pi/2

cos\_phi = np.cos(phi)

sin\_phi = np.sin(phi)

matrix\_phi = np.array([[cos\_phi, 0, sin\_phi, 0],

                    [0, 1, 0, 0],

                    [-sin\_phi, 0, cos\_phi, 0],

                    [0, 0, 0, 1]])

# Rotate from Y

matrix\_Y\_rev = matrix\_Y.T

# Rotate from ZOY

matrix\_ZOY\_rev = matrix\_ZOY.T

# Moving to center

matrix\_move\_from\_center = np.array([[ 1, 0, 0, a],

                            [ 0, 1, 0, b],

                            [ 0, 0, 1, c],

                            [ 0, 0, 0, 1]])

Маємо аналогічний результат:

Изображение выглядит как диаграмма, дизайн

Автоматически созданное описание

5. Задати площину, як у прикладі 2, здійснити симетрію куба відносно

площини, вказати всі математичні перетворення, показати кінцевий

результат, який відповідає вимогам задачі. Записати відповідні математичні

перетворення.

Matrices

double\_sqrt = (A\*\*2+B\*\*2)\*\*0.5

Move (plane) to center of coordinates

M1 = np.array([[1,0,0,0],

               [0,1,0,0],

               [0,0,1,D/C],

               [0,0,0,1]])

Rotate around z

M2 = np.array([[A/double\_sqrt, B/double\_sqrt, 0,0],

               [-B/double\_sqrt, A/double\_sqrt, 0,0],

               [0,0,1,0],

               [0,0,0,1]])

Rotate to XOY

M3 = np.array([[C/triple\_sqrt, 0, -double\_sqrt/triple\_sqrt,0],

               [0, 1, 0,0],

               [double\_sqrt/triple\_sqrt,0,C/triple\_sqrt,0],

               [0,0,0,1]])

Symmetry to z

M4 = np.array([[1,0,0,0],

               [0,1,0,0],

               [0,0,-1,0],

               [0,0,0,1]])

Reverse rotate around z

M5 = M3.T

Rotate from XOY

M6 = M2.T

Move from center of coordinates

M7 = np.array([[1,0,0,0],

               [0,1,0,0],

               [0,0,1,-D/C],

               [0,0,0,1]])

Результат:

Изображение выглядит как диаграмма, зарисовка, линия, Технический чертеж

Автоматически созданное описание

8. (для студентів, які навчаються на кафедрі ММАД) Навести алгоритм з

математичними перетвореннями для побудови симетрії відносно площини на

основі знань з аналітичної геометрії. Порівняти результати (програмно)

Geometric calculations

A\*x+B\*y+C\*z+D=0

x0, y0, z0 – координати точки

(x-x0)/A = (y-y0)/B = (z-z0)/C = t

x = A\*t+x0, y = B\*t+y0,  z = C\*t+x0

A\*(A\*t+x0)+B\*(B\*t+y0)+C\*(C\*t+z0)+D=0

t = h

x = A\*h+x0, y = B\*h+y0,  z = C\*h+z0 coordinates of symmetric point

Програмна реалізація

cube\_vertices = np.array([

    [4, 6, 2],

    [6, 6, 2],

    [6, 8, 2],

    [4, 8, 2],

    [4, 6, 4],

    [6, 6, 4],

    [6, 8, 4],

    [4, 8, 4]])

cube\_vertices\_with\_ones = np.hstack([cube\_vertices, np.ones((cube\_vertices.shape[0], 1))])

sym\_cube = []

for vertex in cube\_vertices:

    t = symbols('t')

    x, y, z = vertex[0], vertex[1], vertex[2]

    equation = Eq(A \* (A \* t + x) + B \* (B \* t + y) + C \* (C \* t + z) + D, 0)

    solution = solve(equation, t)

    x\_o, y\_o, z\_o = A\*solution[0]+x, B\*solution[0]+y, C\*solution[0]+z

    x\_sym, y\_sym, z\_sym = 2\*x\_o-x, 2\*y\_o-y, 2\*z\_o-z

    point = [x\_sym, y\_sym, z\_sym]

    sym\_cube.append(point)

Изображение выглядит как диаграмма, линия, зарисовка, текст

Автоматически созданное описаниеРезультат:

Координати симетричного куба з завдання номер 5:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Координати симетричного куба з завдання номер 8:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

10. (для всіх студентів) Взяти той самий куб, який був поданий у попередніх

пунктах. Здійснити ортогональне проектування даного кубу на площину

загального виду, яка не паралельна жодній з координатних площин. Вказати

відповідні математичні перетворення, розробити відповідне програмне

забезпечення.

Знаходимо кут між нормальним вектором і площиною Z

def angle\_between\_planes(norm\_vector):

    # Нормалізуємо вектор

    n = norm\_vector / np.linalg.norm(norm\_vector)

    # Нормалізований вектор до площини z

    z\_normal = np.array([0, 0, 1])

    # Знаходимо косинус кута

    cos\_theta = np.dot(n, z\_normal) / (np.linalg.norm(n) \* np.linalg.norm(z\_normal))

    # Знаходимо кут в радіанах

    theta\_rad = np.arccos(cos\_theta)

    return theta\_rad

angle = angle\_between\_planes(normal\_vector)

# Rotate to make plane || Z

M1\_plane\_to\_Z = np.array([[np.cos(angle),0,np.sin(angle),0],

                          [0, 1, 0, 0],

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

                          [0,0,0,1]])

# Distance to move

z1 = plane\_moved\_y[0][2]

# Move to Z=0 (XOY)

M2\_moved\_to\_z0 = np.array([[1,0,0,0],

                [0, 1, 0, 0],

                [0,0, 1, -z1],

                [0,0,0,1]])

# Orthogonal projection to plane

M3\_projection = np.array([[1,0,0,0],

                [0, 1, 0, 0],

                [0,0, 0, 0],

                [0,0,0,1]])

# Move from Z=0 (XOY)

M4\_moved\_from\_z0 = np.array([[1,0,0,0],

                [0, 1, 0, 0],

                [0,0, 1, z1],

                [0,0,0,1]])

# Rotate to make plane not || XOY

M5\_plane\_from\_Z = M1\_plane\_to\_Z.T

Изображение выглядит как зарисовка, рисунок, диаграмма, дизайн

Автоматически созданное описание

Изображение выглядит как зарисовка, диаграмма, рисунок, линия

Автоматически созданное описание