MASTER OF COMPUTER APPLICATION

SEMESTER 1

DATA VISUALISATION

Unit 13

3D Visualisations in Python

Table of Contents

1. INTRODUCTION

3D scatter plots are an advanced data visualisation form that displays points in a three-dimensional space. Unlike 2D scatter plots that only account for two variables, 3D scatter plots incorporate a third dimension, offering a deeper, more nuanced understanding of data relationships. This visualisation technique benefits datasets with three variables, allowing for a comprehensive view that can reveal patterns, trends, and correlations not evident in traditional two-dimensional plots. Through the strategic use of axes, data points, and visual cues like colour and size, 3D scatter plots provide a rich, detailed canvas for data analysis and interpretation.

1.1 Learning Objectives

After studying this unit, you will be able to:

- ❖ *Describe the essential components of a 3D scatter plot and 3D Mesh*
- ❖ *Explain how the plots reveal relationships between the variables in a dataset*
- ❖ *Explore the various best practices to be considered for creating the 3D visuals*

2. 3D SCATTER PLOTS

A 3D scatter plot is a type of data visualisation that displays data points in three-dimensional space, making it particularly useful for representing and analysing datasets with three variables. Unlike traditional 2D scatter plots that show relationships between two variables on a flat plane, 3D scatter plots add depth by incorporating a third variable, creating a more comprehensive view of the data. Here's a detailed explanation of 3D scatter plots:

Key Components of a 3D Scatter Plot:

1. X, Y, and Z Axes:

In a 3D scatter plot, the X, Y, and Z axes represent the three dimensions of the data being visualised. Each axis is a numerical scale corresponding to a specific variable or dimension of the dataset. These axes create a 3D Cartesian coordinate system.

- **X-Axis:** This typically represents one of your dataset's independent variables or features. The data points are positioned along this axis based on the values of this variable.
- **Y-Axis:** Similar to the X-axis, this represents another independent variable or feature. The values of this variable determine each point's position along the Y-axis.
- **Z-Axis:** The third dimension is represented along the Z-axis. This variable could be a third independent variable or the dependent variable you are trying to understand in relation to the other two.

The intersection of these three axes defines a unique point in 3D space, with coordinates (X, Y, Z) corresponding to the values of the three variables. The position of data points in this 3D space provides insights into the relationships and patterns among these variables.

2. Data Points:

Data points are the individual observations or records from your dataset. In a 3D scatter plot, each data point is visually represented as a marker or symbol. These markers are positioned within the 3D space according to their specific values along the X, Y, and Z axes. Data points are often represented as dots, spheres, or other symbols.

Each data point on the plot corresponds to a specific combination of the three variables under consideration. For example, in a scientific experiment, each point might represent a specific set of conditions (X, Y, Z) and the resulting measured outcome.

3. Color and Size:

To provide even more information and context, additional visual cues can enhance data points in a 3D scatter plot. These cues include colour and size:

- **Color:** You can colour-code data points to represent a fourth variable, such as a categorical variable or a gradient of a continuous variable. For example, you might colour-code data points by region or by the magnitude of a related variable. This enhances the interpretability of the plot by adding an extra layer of information.
- **Size:** You can vary the size of data points to represent a fifth variable. For instance, you might use the data points' size to indicate each data point's importance or significance. Larger points may represent more important observations, while smaller points signify less significant ones.

Use Cases of 3D Scatter Plots:

1. Visualising Multivariate Data:

3D scatter plots are a valuable tool for simultaneously visualising and understanding relationships between three variables. This is particularly useful in scientific research and engineering when analysing data with multiple measured characteristics. For example, in environmental science, a 3D scatter plot can help researchers visualise the interplay between temperature, pressure, and time to gain insights into weather patterns, climate changes, or geological phenomena. By representing data points in a 3D space, scientists and analysts can better comprehend the interactions between variables.

2. Identifying Patterns:

3D scatter plots accurately reveal multidimensional data patterns, clusters, or trends. This makes them indispensable in fields like biology, where they can be used to analyse complex datasets, such as gene expression data. Biologists can use 3D scatter plots to explore relationships between genes, their expressions under different conditions, and their impact on biological processes. Similarly, in geology, 3D scatter plots display geological formations, making identifying patterns in rock structures or mineral deposits easier. Researchers can make meaningful discoveries by spotting trends and groupings in the 3D space.

3. Exploring Data Space:

A 3D scatter plot provides a dynamic way to explore data space when working with large, complex datasets. Users can interact with the data by rotating, zooming, and panning within the 3D visualisation. This allows for a more thorough exploration of the data's structure and relationships. Researchers can comprehensively understand the data, uncover hidden insights, and validate hypotheses. For example, in materials science, a 3D scatter plot can help scientists visualise the properties of various materials across different temperature, pressure, and composition conditions.

4. Outlier Detection:

Outlier detection is essential in data analysis, and 3D scatter plots are instrumental in this process. Outliers, which are data points that significantly deviate from most data, can be easily identified in a 3D scatter plot. For instance, in finance, a 3D scatter plot can be used to identify anomalies in stock prices by visualising factors like trading volume, price changes, and market sentiment. Analysts can detect potential fraud, errors, or unusual market behaviour by highlighting data points that are far from the center or follow different patterns.

5. Regression Analysis:

In regression analysis, 3D scatter plots help visualise the relationships between multiple independent variables and a dependent variable. This makes it easier to understand how different variables impact the outcome. For example, in economics, a 3D scatter plot can explore how factors like GDP, inflation, and interest rates affect exchange rates. By visually inspecting the relationships, analysts can make informed decisions, develop predictive models, and understand the relative importance of each variable.

Challenges and Considerations:

1. Overplotting:

Overplotting occurs when a large number of data points are densely packed in the 3D space, making it challenging to distinguish individual data points. This can lead to a cluttered and confusing visualisation. To mitigate overplotting in 3D scatter plots:

- **Color and Transparency:** Use various colours or transparency to differentiate data points. Assigning different colours to clusters or categories of data points can help reveal patterns and reduce confusion.
- **Sizing:** Adjust the size of data points based on their significance or density. Larger points may represent clusters or regions with more data, making them stand out.
- **Subsampling:** In cases of extremely dense data, consider subsampling the dataset to include only representative data points. This can simplify the plot while preserving essential information.

2. Data Interpretation:

Interpreting 3D scatter plots can be more complex than 2D plots due to the additional dimension. To address this challenge and ensure meaningful interpretation:

- **Clear Axis Labeling:** Ensure the axes are clearly labelled to provide context for each dimension. Clear labels make it easier to understand what each axis represents.
- **Grid Lines:** Adding grid lines in the plot can help users read and interpret the values more accurately.
- **Projection:** Consider projecting the 3D plot onto 2D planes, such as scatter plots on each pair of dimensions (XY, XZ, and YZ), to gain insights into the data from different angles. This simplifies interpretation by breaking the data into smaller, more manageable parts.

3. Rotation and Interaction:

3D scatter plots often require the ability to rotate, zoom, and interact with the plot to explore the data fully. This is particularly important when dealing with complex datasets. Considerations include:

- **Interactive Tools:** Provide users with rotating, zooming, and panning tools within the 3D plot. Interactive visualisation tools or software that allow users to manipulate the plot can help uncover hidden patterns or trends.
- **User Experience:** Consider the user experience when designing 3D scatter plots. Ensure the interface is intuitive and user-friendly, enabling users to explore the data without confusion.
- **Performance:** The performance of interactive 3D visualisation tools can be considered when dealing with very large datasets. Optimising software and hardware for smooth interaction is essential for an efficient user experience.

2.1. Creating 3D Scatter plots in Python

Creating 3D scatter plots in Python is a useful way to visualise data with three dimensions, and it's particularly valuable for understanding relationships between variables. To create 3D scatter plots, you can use libraries such as Matplotlib, Plotly, or Mayavi. This explanation will focus on creating 3D scatter plots using Matplotlib and Python.

Prerequisites:

Before you start, make sure you have Python installed, and consider installing Matplotlib if you haven't already. You can install Matplotlib using pip:

pip install matplotlib

Here's a step-by-step guide to creating 3D scatter plots:

1. Import Necessary Libraries:

First, import the required libraries for data generation, including Matplotlib and NumPy. You may also need the `%matplotlib` magic command to enable interactive plotting if you're working in a Jupyter Notebook.

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline  # Only needed in Jupyter Notebook
```

2. Generate Data:

Create sample data to be plotted in the 3D scatter plot. You need three arrays representing one dimension (X, Y, Z).

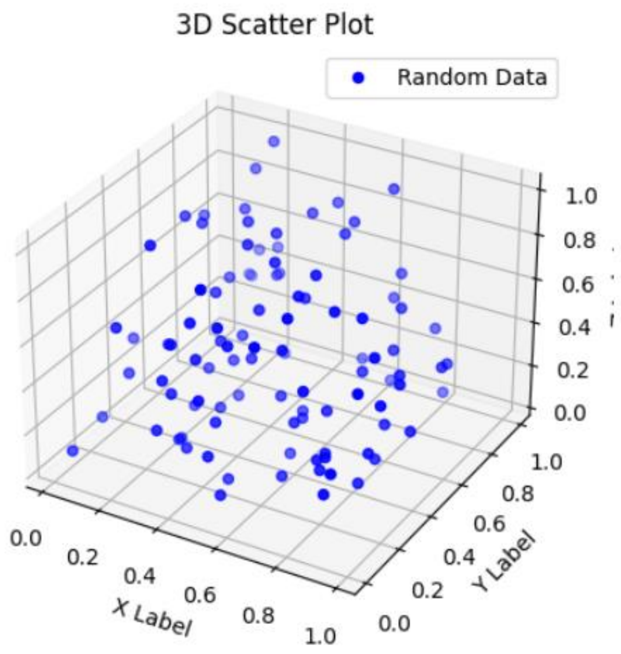
```
# Generate random data for demonstration
num_points = 100
x = np.random.rand(num_points)
y = np.random.rand(num_points)
z = np.random.rand(num_points)
```

3. Create the 3D Scatter Plot:

Use Matplotlib's `scatter` function with the '3d' projection to create the 3D scatter plot. Customise the plot with labels, colours, and markers as needed.

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')  # Create a 3D axis
# Plot the 3D scatter plot
ax.scatter(x, y, z, c='b', marker='o', label='Random Data')
# Customise the plot
ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')
ax.set_title('3D Scatter Plot')
ax.legend()
plt.show()
```

Output:



4. Customisation:

You can further customise the 3D scatter plot to meet your specific needs:

- Adjust the colour and style of data points by modifying the `c` and `marker` parameters in the `scatter` function.
- Label the axes using `set_xlabel`, `set_ylabel`, and `set_zlabel`.
- Add a title using `set_title`.
- Include a legend to explain the data being plotted.

5. Saving the Plot (Optional):

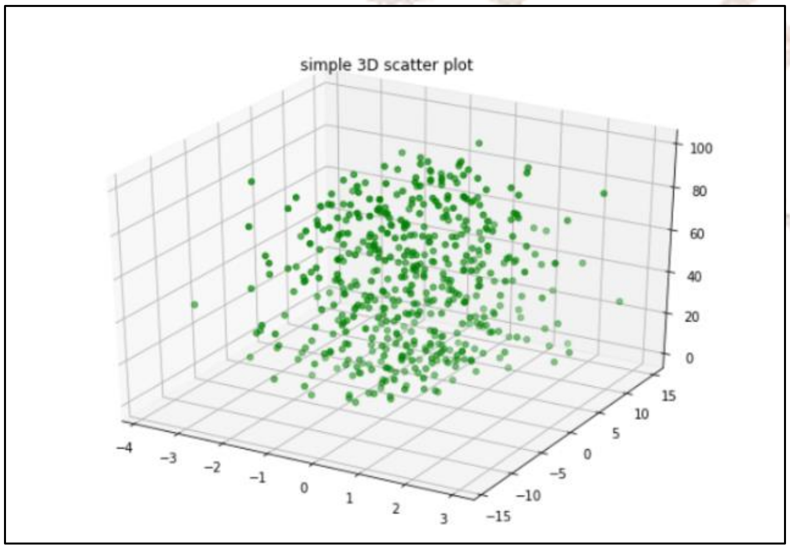
If you want to save the 3D scatter plot as an image, use `plt.savefig('filename.png')` before calling `plt.show()`.

By following these steps, you can create a 3D scatter plot in Python using Matplotlib. You can also explore additional features, such as adding colour maps, legends, and annotations, to enhance the visualisation and make it more informative for your specific data analysis needs.

Example :

```
# Import libraries
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt
# Creating dataset
z = np.random.randint(100, size =(50))
x = np.random.randint(80, size =(50))
y = np.random.randint(60, size =(50))
# Creating figure
fig = plt.figure(figsize = (10, 7))
ax = plt.axes(projection ="3d")
# Creating plot
ax.scatter3D(x, y, z, color = "green")
plt.title("simple 3D scatter plot")
# show plot
plt.show()
```

The output will be shown as :



3. 3D MESH

A 3D mesh, in the context of computer graphics and 3D modelling, refers to a three-dimensional representation of a surface or object created by connecting a series of vertices (points in 3D space) to form polygons like triangles or quadrilaterals. These polygons are organised to create a continuous, structured surface. The term "mesh" comes from the interconnected network of vertices, edges, and faces that define the object's shape.

Here are the key components and concepts associated with 3D meshes:

1. **Vertices (Nodes):** Vertices are the fundamental points in 3D space. They are the building blocks of a 3D mesh and represent the object's surface. Each vertex has a set of 3D coordinates (x, y, z) to define its position.
2. **Edges:** Edges are the line segments that connect vertices. They form the boundaries of the polygons (faces) and help define the object's shape.
3. **Faces (Polygons):** Faces are flat, planar regions formed by connecting three or more vertices with edges. Triangles and quadrilaterals (quads) are the most common faces used in 3D meshes.
4. **Topology:** The arrangement of vertices, edges, and faces in a 3D mesh is known as its topology. A well-structured topology is essential for smooth rendering and efficient mesh manipulation.
5. **Mesh Resolution:** Mesh resolution refers to the density of vertices and faces in a 3D mesh. A higher-resolution mesh contains more vertices and faces, providing finer details but demanding more computational resources.
6. **Normals:** Normals are vectors associated with each face or vertex of a 3D mesh. They indicate the direction of the face or the vertex, which is crucial for lighting and shading calculations in 3D rendering.

3D meshes are used extensively in computer graphics, animation, and 3D modelling for various purposes:

- **Rendering:** 3D meshes are the basis for rendering 3D objects in computer graphics. They define the shape, structure, and surface properties of 3D models.

- **Animation:** Mesh deformation and manipulation allow for character animation in video games and movies. Animators use rigging and skeletal systems to control and deform 3D meshes.
- **3D Modeling:** Artists and designers create 3D models by shaping and manipulating meshes. This process is fundamental for architectural visualisation, product design, and digital art.
- **Simulation:** 3D meshes are used in scientific simulations and engineering applications to model physical objects and phenomena. Finite element analysis (FEA) relies on meshing structures for simulations.

3D meshes come in various forms, from simple geometric shapes to highly detailed models of real-world objects. Depending on the application, the complexity and detail of the mesh can vary significantly. Proper mesh optimisation and efficient rendering techniques are crucial for achieving realistic and responsive 3D graphics.

3.1. Constructing 3D Mesh Plots in Python

3D Mesh Plots

A mesh plot is a plot that has three three-dimensional surfaces, solid edges, and no face colours. A Mesh plot is a way to create a 3D set of triangles with vertices given by x, y, and z. If only coordinates exist, then a triangle is formed by algorithms such as Delaunay triangulation. I, J, and K parameters can also create a triangle.

Syntax:

class plotly.graph_objects.Mesh3d(arg=None, hoverinfo=None, x=None, y=None, z=None, kwargs)

Parameters:

arg: dict of properties compatible with this constructor or an instance of plotly.graph_objects.Mesh3d

hoverinfo: Determines which trace information appears on hover. If none or skip is set, no information is displayed upon hovering. But, if none is set, click and hover events are still fired.

x: Sets the X coordinates of the vertices. The nth element of vectors x, y and z jointly represents the nth vertex's X, Y and Z coordinates.

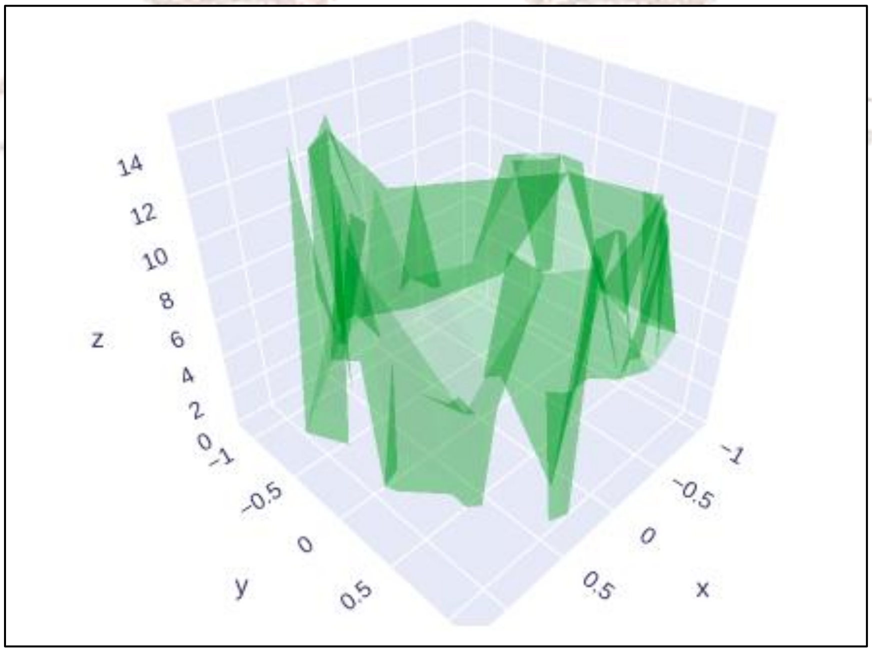
y: Sets the Y coordinates of the vertices. The nth element of vectors x, y and z jointly represents the nth vertex's X, Y and Z coordinates.

z: Sets the Z coordinates of the vertices. The nth element of vectors x, y and z jointly represents the nth vertex's X, Y and Z coordinates.

Example 1:

```
import plotly.graph_objects as go

import numpy as np

# Data for three-dimensional scattered points

z = np.random.random(100) * 15  # Generates an array of random values multiplied by 15

x = np.sin(z) + 0.1 * np.random.randn(100)  # Added missing multiplication operator (*)

y = np.cos(z) + 0.1 * np.random.randn(100)  # Added missing multiplication operator (*)

fig = go.Figure(data=[go.Mesh3d(x=x, y=y, z=z, color='green', opacity=0.20)])

fig.show()
```

Output:



3.2. Displaying Surface Plots

Displaying surface plots in a 3D mesh involves creating visual representations of surfaces defined by the vertices and faces of a 3D mesh. Surface plots provide a way to visualise complex 3D data and are commonly used in scientific and engineering fields, computer graphics, and data analysis. Here's how you can display surface plots using 3D meshes:

1. Generate or Load 3D Mesh Data:

You need 3D mesh data to create surface plots. This data includes the vertices and faces that define the object's shape or surface you want to visualise. You can generate this data using software or load existing 3D mesh models in formats like .obj, .stl, or .ply.

2. Choose a Visualisation Library:

To display 3D surface plots, you'll need a visualisation library or software that can render 3D graphics. Common choices include Matplotlib (Python), Three.js (JavaScript), Unity (C#), and other 3D modelling and rendering tools.

3. Create a 3D Scene:

In the chosen library or software, create a 3D scene where you'll render the surface plot. Set up a 3D viewport and camera to control the view of the plot.

4. Mapping Data to the Mesh:

If you have data that you want to visualise on the surface, map this data to the vertices or faces of the 3D mesh. For example, if you have temperature data for a 3D object, assign each vertex or face a colour corresponding to its temperature value.

5. Render the Surface Plot:

Using appropriate functions or tools to render the 3D mesh as a surface plot depending on the chosen library or software. Here are some common methods:

- **Matplotlib (Python):** Matplotlib offers a 3D plotting toolkit, including functions like `plot_surface` for rendering 3D surfaces. You can set colours and shading based on your data.

- **Three.js (JavaScript):** Three.js is a popular JavaScript library for 3D graphics. You can create a mesh using your 3D mesh data and apply materials and textures to visualise the surface.
- **Unity (C#):** Unity is a game engine that allows you to import 3D models and render them in 3D scenes. You can use Unity's material and shader systems to create surface plots.

6. Adjust Visualisation Properties:

Customise the appearance of the surface plot by adjusting properties such as lighting, shading, colours, transparency, and texture mapping. These properties enhance the visual quality and understanding of the surface.

7. Interactivity (Optional):

Depending on the application, you may want to add interactivity to the surface plot. Users can interact with the plot by rotating, zooming, or selecting specific regions of the surface. Interactive features can be implemented using event handlers and user interface elements.

8. Rendering and Display:

Finally, render and display the surface plot in the 3D viewport or window. Save the plot as an image or display it interactively in a 3D viewer or application.

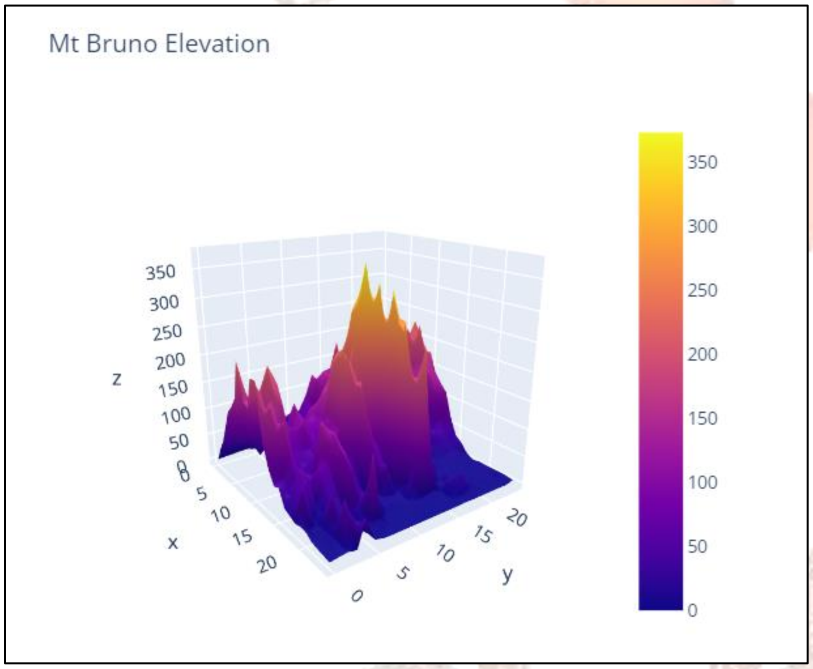
Surface plots created using 3D meshes are valuable for visualising complex data, understanding spatial relationships, and presenting information in scientific research, engineering, computer-aided design (CAD), and computer graphics. They provide a powerful way to communicate 3D information and insights to various audiences.

Example

```
import plotly.graph_objects as go

import pandas as pd

# Read data from a csv

z_data =
pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/api_docs/mt_
bruno_elevation.csv')
```

```
fig = go.Figure(data=[go.Surface(z=z_data.values)])
fig.update_layout(title='Mt Bruno Elevation', autosize=False,
              width=500, height=500,
              margin=dict(l=65, r=50, b=65, t=90))
fig.show()
```

Output :



4. BEST PRACTICES FOR 3D VISUALS

Use 3D Only When Necessary

- **Justify the Third Dimension:** Use 3D representations only when the third dimension adds valuable insights that cannot be effectively conveyed through 2D visuals. This could include spatial data, volumetric measurements, or multidimensional relationships.

Keep It Simple and Clear

- **Avoid Overcrowding:** 3D visuals can quickly become cluttered. Ensure that every element serves a purpose and contributes to understanding the data.
- **Simplify Complexity:** Use simplification techniques like filtering, highlighting, or focusing on key data points to make complex data more understandable.

Focus on Legibility

- **Readable Text:** Ensure that any text is legible from various angles. Use bold, sans-serif fonts and consider dynamic sizing or orientation for readability.
- **Contrast and Colours:** Use colours with high contrast and a coherent colour scheme to differentiate data points. Be mindful of colour blindness by choosing colour palettes that are accessible to all users.

Effective Lighting and Shading

- **Use Lighting to Enhance Depth:** Proper lighting can help emphasise texture, depth, and the hierarchical structure of the data. Ambient and directional lighting are crucial in making 3D visuals more understandable.
- **Shading Techniques:** Implement shading techniques to improve the perception of depth and volume, such as drop shadows or gradients.

Interactive Elements

- **Rotation and Zoom:** This allows users to interact with the 3D visualisation through rotation, zooming, and panning to explore data from different perspectives.

- **Tooltips and Details on Demand:** Implement interactive elements such as tooltips or clickable regions to provide additional information without cluttering the primary view.

Perspective and Viewpoint

- **Choose the Right Angle:** Start with an intuitive angle showcasing the data's most important aspects. Consider offering preset viewpoints for common data explorations.
- **Avoid Distortion:** Be wary of perspective distortion, which can mislead interpretation. Parallel projection can be used to mitigate perspective distortion in certain cases.

Animation and Transition

- **Smooth Transitions:** Use animations to smoothly transition between different states or views of the data. This helps maintain context and understand the flow of data.
- **Narrative Flow:** Leverage animation to build a narrative or guide the viewer through the data exploration process, highlighting key points and relationships.

Testing and Feedback

- **User Testing:** Conduct user testing to gather feedback on the visualisation's usability, interpretability, and overall effectiveness.
- **Iterative Design:** Be prepared to iterate on the design based on user feedback, performance issues, or new insights into the data.

Performance Considerations

- **Optimise for Performance:** 3D visualisations can be resource-intensive. Optimise models, textures, and rendering techniques to ensure smooth device performance.

Accessibility

- **Provide Alternatives:** Not everyone can effectively interpret 3D visuals. Provide alternative 2D visualisations, descriptive text, or data tables for accessibility.

5. SUMMARY

3D scatter plots are invaluable tools in data visualisation, offering a multidimensional perspective on complex datasets. By mapping data points in a three-dimensional space, they simultaneously facilitate the exploration of relationships between three variables. Key elements such as the X, Y, and Z axes lay the foundation for these visualisations while using colour and size adds layers of meaning, representing additional variables. Despite their advantages, 3D scatter plots also present challenges such as overplotting and data interpretation complexities, necessitating strategies like colour differentiation and interactive exploration to ensure clarity and insight. When used appropriately, 3D scatter plots can unveil intricate data relationships, aid in outlier detection, and support robust analytical processes like regression analysis.

6. QUESTIONS

Self-Assessment Questions

1. What is the primary purpose of creating 3D scatter plots in Python?
2. How can 3D scatter plots help in analysing multidimensional data?
3. Define the term "3D mesh" in the context of computer graphics.
4. What are the key components of a 3D mesh?
5. In the context of 3D visualisation, what does the term "mesh resolution" refer to?
6. Why is mapping data to vertices or faces important when displaying a 3D mesh plot?
7. Which popular Python library is often used for creating 3D scatter plots?
8. What is the benefit of displaying surface plots in 3D visualisation?
9. Name an application domain where constructing 3D mesh plots is crucial.
10. What are some considerations for addressing overplotting in 3D scatter plots?

Terminal questions

1. Explain the process of creating 3D scatter plots in Python.
2. Describe the significance of visualising three-dimensional relationships in data analysis.
3. Provide real-world examples of situations where 3D visualisation is essential and explain how it aids in gaining insights from complex data.
4. Discuss the components of a 3D mesh and their roles in computer graphics and 3D modelling.
5. Explain how to generate or load 3D mesh data, choose a visualisation library, and render the mesh plot.
6. Explain the steps involved in displaying surface plots using 3D meshes.
7. Discuss the challenges and considerations in 3D visualisation, including issues like overplotting, data scaling, interactivity, and interpretation.
8. What are some common use cases of 3D scatter plots?
9. What challenges are associated with 3D scatter plots, and how can they be addressed?
10. List the best practices to be considered while creating 3D visuals.

7. ANSWERS

Self-Assessment Questions

1. The primary purpose of creating 3D scatter plots in Python is to visually represent and analyse relationships between three variables in a three-dimensional space.
2. 3D scatter plots help in analysing multidimensional data by providing a visual representation of data points, allowing users to observe patterns, correlations, and trends within the data.
3. In the context of computer graphics, a 3D mesh refers to a three-dimensional representation of a surface created by connecting vertices with edges to form polygons. These polygons define the object's shape.
4. The key components of a 3D mesh include vertices (nodes), edges (line segments connecting vertices), and faces (flat regions formed by connecting vertices with edges).
5. Mesh resolution in 3D visualisation refers to the density of vertices and faces within a 3D mesh. Higher resolution means more vertices and faces, providing finer details in the representation.
6. Mapping data to vertices or faces is essential when displaying a 3D mesh plot because it associates data values with specific points or regions on the mesh, allowing for data visualisation on the surface.
7. Matplotlib is a popular Python library often used for creating 3D scatter plots.
8. The benefit of displaying surface plots in 3D visualisation is that they provide a way to visualise and understand complex 3D data, including spatial relationships and patterns.
9. Constructing 3D mesh plots is crucial in fields like computer graphics, animation, and scientific simulations.
10. Considerations for addressing overplotting in 3D scatter plots include using techniques like color coding, size variation, transparency, and subsampling to make data points distinguishable.

Terminal Questions

1. Refer section 3.1
2. Refer section 3.2
3. Refer section 3
4. Refer section 4.1
5. Refer section 4.2
6. Refer section 4.2
7. Refer section 3.1
8. Refer section 3
9. Refer section 3
10. Refer section 5