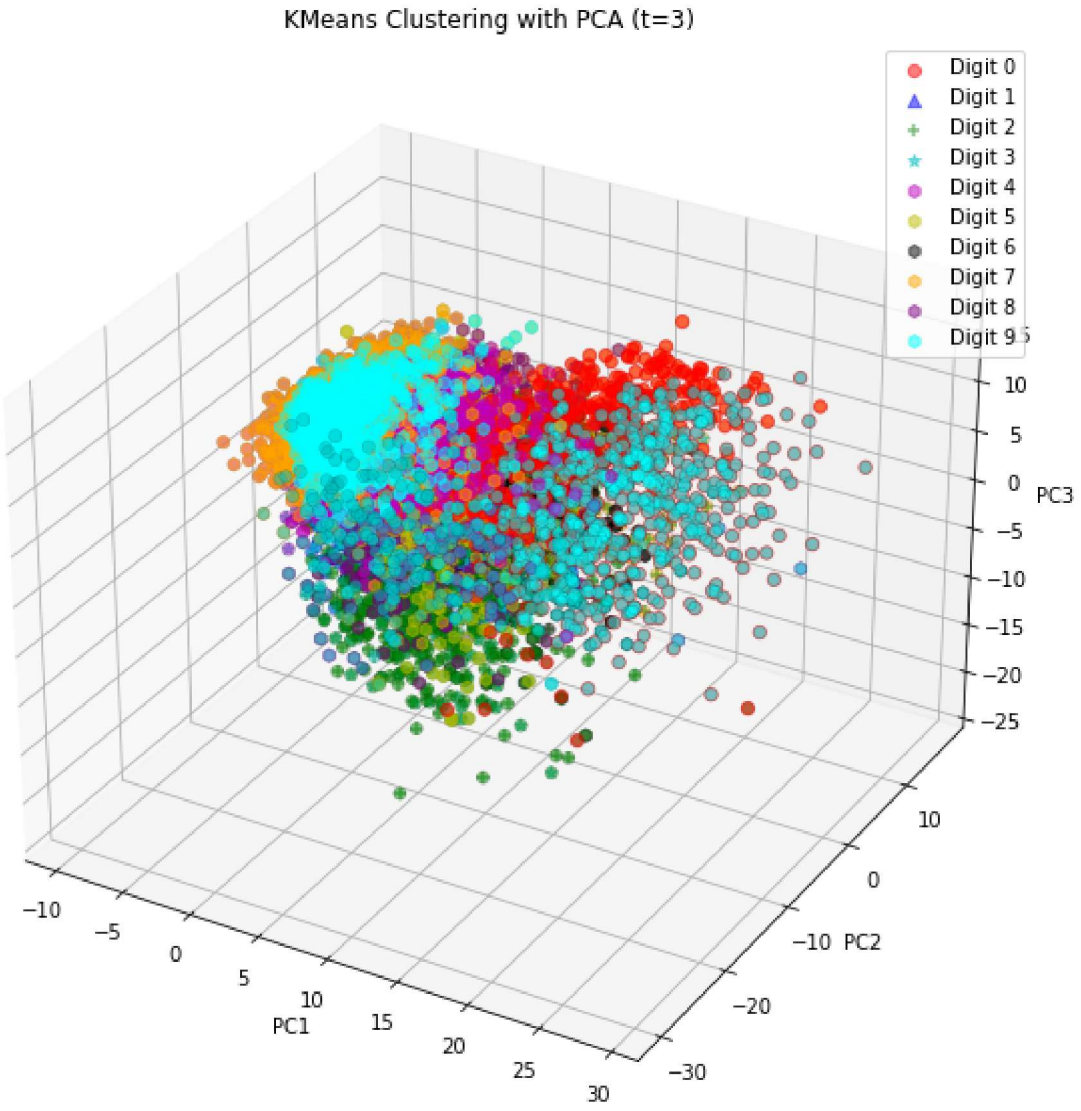
Problem 4

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import KMeans
from sklearn.datasets import fetch_openml
from sklearn.utils import shuffle
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings('ignore')
```

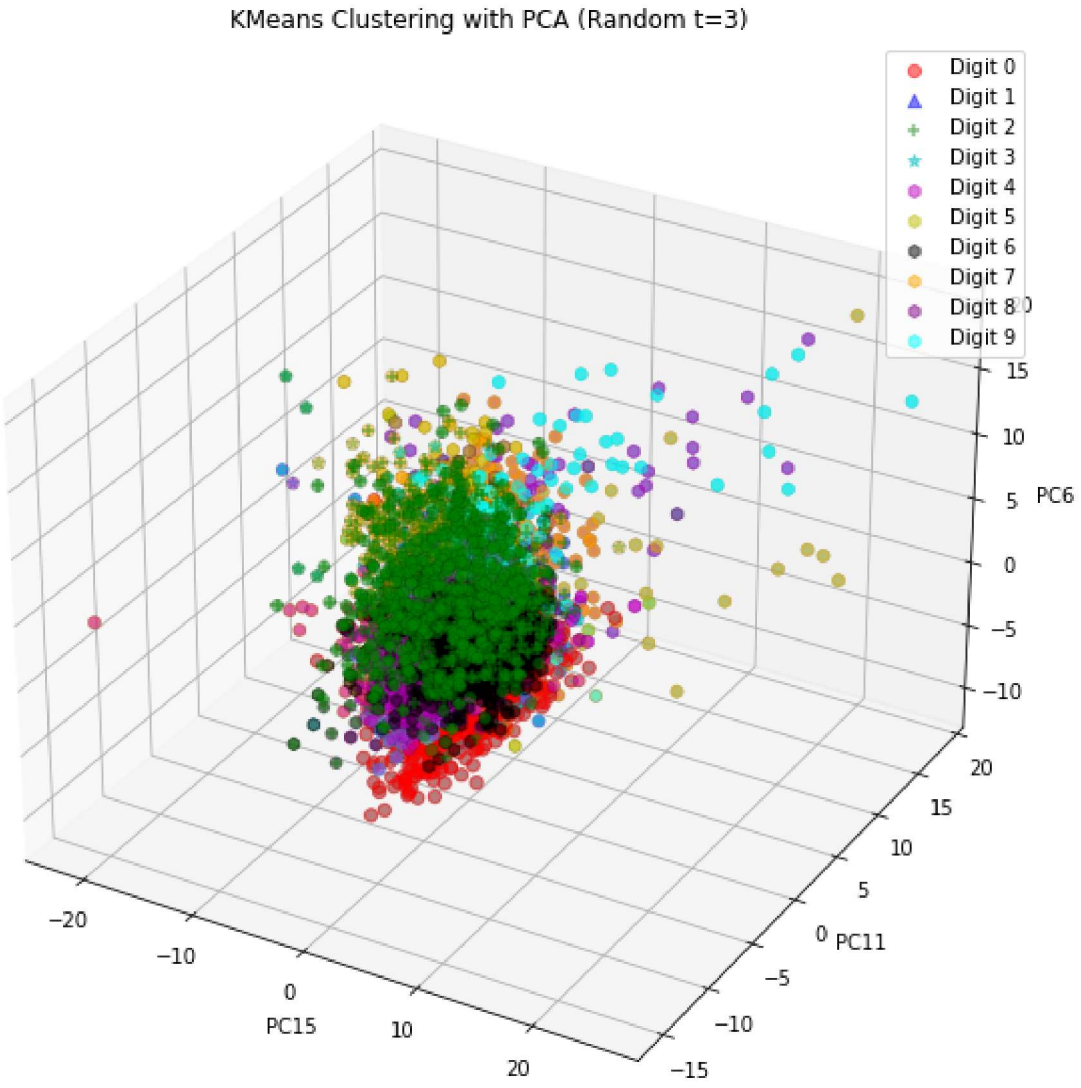
In [2]:
```
# Load MNIST dataset
mnist = fetch_openml('mnist_784')
X, y = mnist.data, mnist.target.astype(int)

# Shuffle and select a sample of 10000 data points
X, y = shuffle(X, y, random_state = 42)
X, y = X[:10000], y[:10000]

# Normalize the data by subtracting the mean and dividing by the standard devi
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Perform KMeans clustering with 10 clusters
kmeans = KMeans(n_clusters = 10, random_state=42)
y_pred = kmeans.fit_predict(X)
```

In [3]:
```
# Calculate the covariance matrix
covariance_matrix = np.cov(X.T)

# Calculate the eigenvalues and eigenvectors of the covariance matrix
eigenvalues, eigenvectors = np.linalg.eig(covariance_matrix)

# Sort the eigenvalues in descending order
idx = np.argsort(eigenvalues)[::-1]
eigenvalues = eigenvalues[idx]
eigenvectors = eigenvectors[:, idx]

# Choose the number of principal components (K)
K = 3

# Project the dataset onto the K principal components
projection_matrix = eigenvectors[:, :K]
X_pca = np.dot(X, projection_matrix)
```

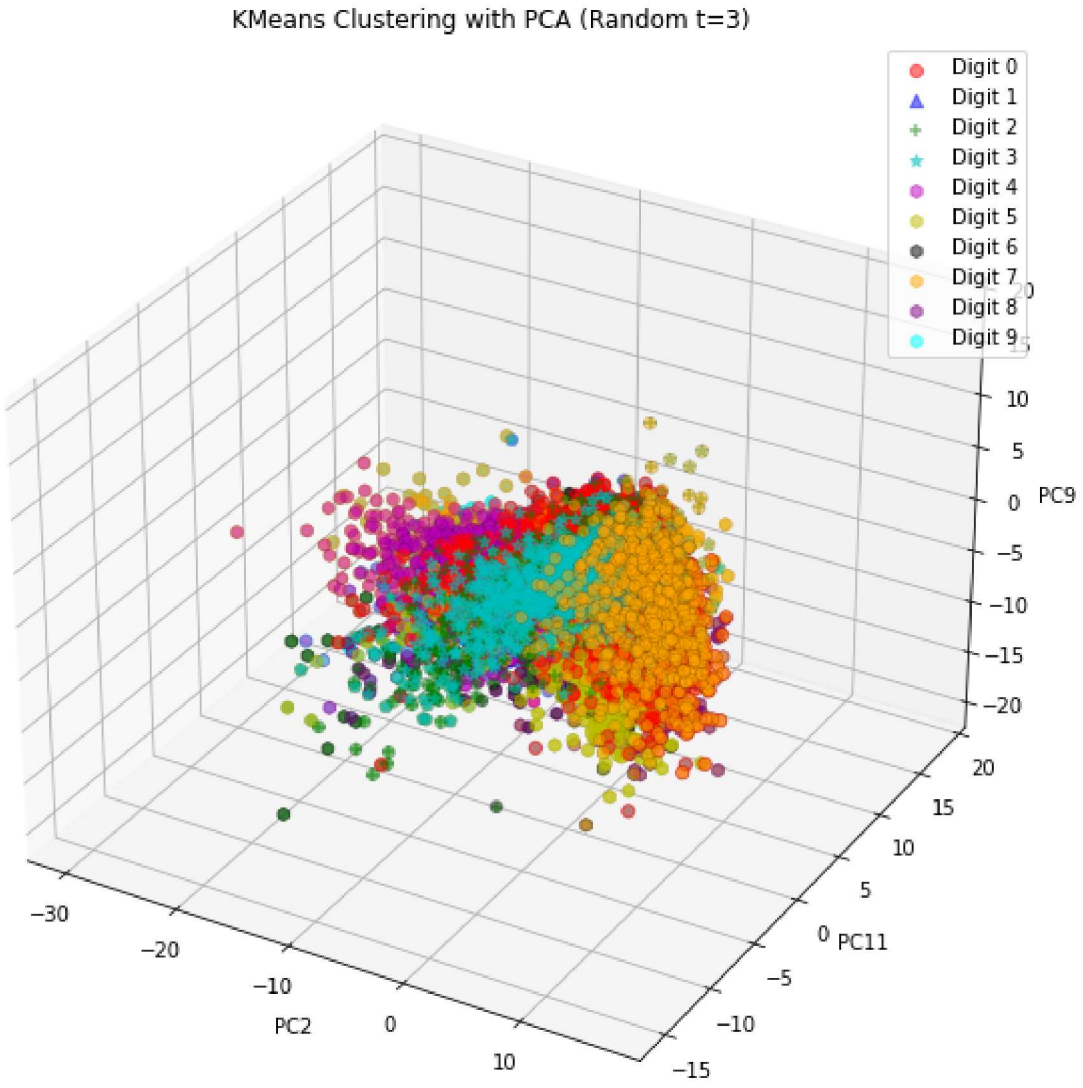
In [4]:
```
# Define shape and color mappings
markers = ['o', '^', '+', '*', 's', 'x', 'D', 'v', '>', '<']
colors = ['r', 'b', 'g', 'c', 'm', 'y', 'k', '#FFA500', '#800080', '#00FFFF']

# Plot the data in 3D with shape and color markers
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')

for i in range(10):
    marker = markers[i] if i < 4 else 'h'
    ax.scatter(
        X_pca[y == i, 0], X_pca[y == i, 1], X_pca[y == i, 2],
        marker=marker, s=40, c=colors[i], alpha=0.5, label=f"Digit {i}"
    )

for i in range(10):
    ax.scatter(
        X_pca[y_pred == i, 0], X_pca[y_pred == i, 1], X_pca[y_pred == i, 2],
        marker='o', s=40, c=colors[i], alpha=0.5, edgecolors='k', linewidths=0
    )

ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_zlabel('PC3')
ax.set_title('KMeans Clustering with PCA (t=3)')
ax.legend()
plt.show()
```


KMeans Clustering with PCA (t=3)
Digit 0
Digit 1
Digit 2
Digit 3
Digit 4
Digit 5
Digit 6
Digit 7
Digit 8
Digit 9
PC1
PC2
PC3

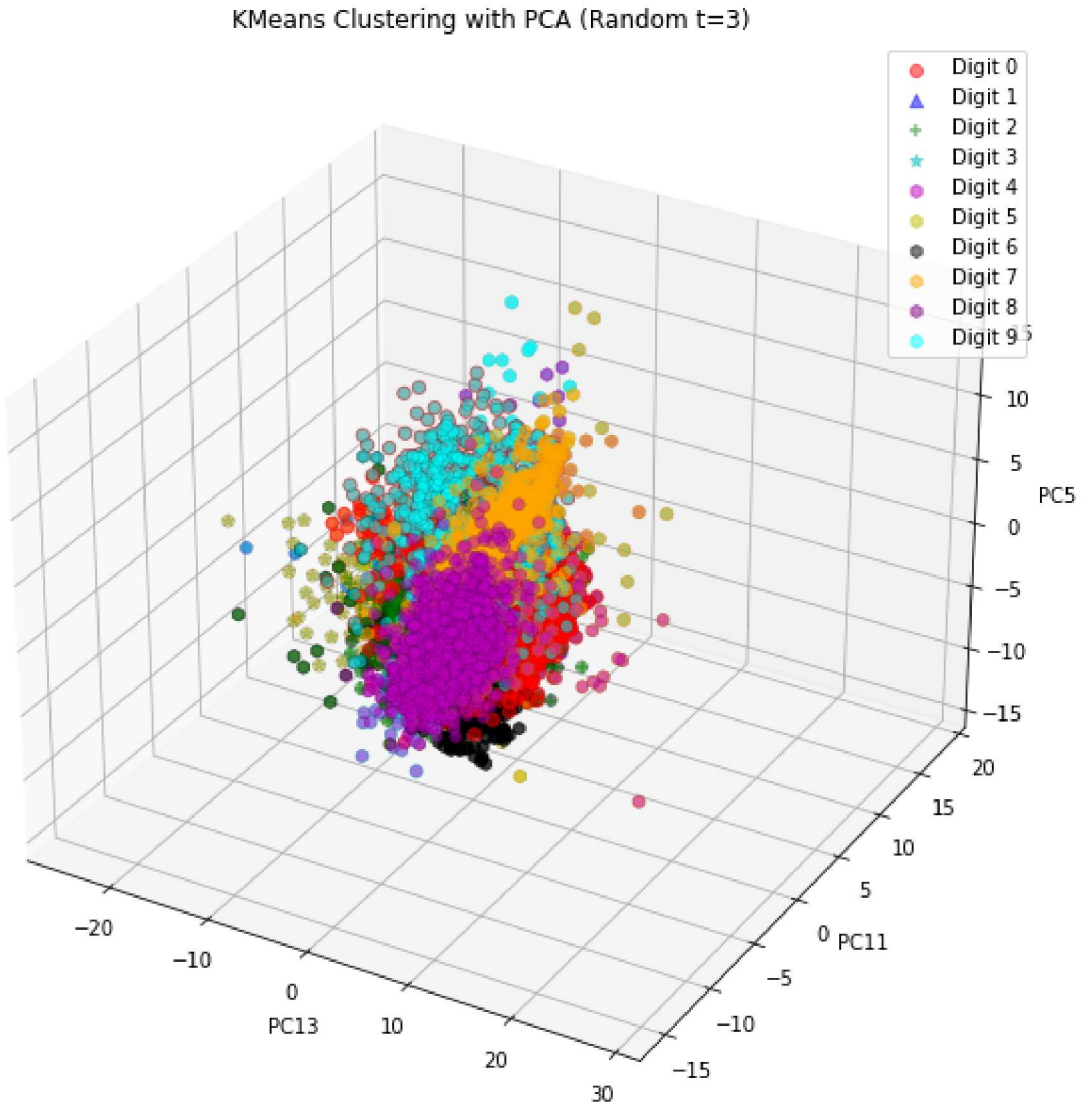
In [5]:
```
# Select 3 random eigenvalues from the top 20
random_eig_idx = np.random.choice(range(20), size=3, replace=False)
random_eig_vecs = eigenvectors[:, random_eig_idx]

# Project the dataset onto the 3 random principal components
X_pca_random = np.dot(X, random_eig_vecs)

# Plot the data in 3D with shape and color markers
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')

for i in range(10):
    marker = markers[i] if i < 4 else 'h'
    ax.scatter(
        X_pca_random[y == i, 0], X_pca_random[y == i, 1], X_pca_random[y == i,
        marker=marker, s=40, c=colors[i], alpha=0.5, label=f"Digit {i}"
    )

for i in range(10):
    ax.scatter(
        X_pca_random[y_pred == i, 0], X_pca_random[y_pred == i, 1], X_pca_rand
        marker='o', s=40, c=colors[i], alpha=0.5, edgecolors='k', linewidths=0
    )

ax.set_xlabel(f"PC{random_eig_idx[0]+1}")
ax.set_ylabel(f"PC{random_eig_idx[1]+1}")
ax.set_zlabel(f"PC{random_eig_idx[2]+1}")
ax.set_title('KMeans Clustering with PCA (Random t=3)')
ax.legend()
plt.show()
```


KMeans Clustering with PCA (Random t=3)
Digit 0
Digit 1
Digit 2
Digit 3
Digit 4
Digit 5
Digit 6
Digit 7
Digit 8
Digit 9
PC6
PC11
PC15
−20
−10
0
10
20
−15
−10
−5
0
5
10
15

In [6]:
```
# Select 3 random eigenvalues from the top 20
random_eig_idx = np.random.choice(range(20), size=3, replace=False)
random_eig_vecs = eigenvectors[:, random_eig_idx]

# Project the dataset onto the 3 random principal components
X_pca_random = np.dot(X, random_eig_vecs)

# Plot the data in 3D with shape and color markers
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')

for i in range(10):
    marker = markers[i] if i < 4 else 'h'
    ax.scatter(
        X_pca_random[y == i, 0], X_pca_random[y == i, 1], X_pca_random[y == i,
        marker=marker, s=40, c=colors[i], alpha=0.5, label=f"Digit {i}"
    )

for i in range(10):
    ax.scatter(
        X_pca_random[y_pred == i, 0], X_pca_random[y_pred == i, 1], X_pca_rand
        marker='o', s=40, c=colors[i], alpha=0.5, edgecolors='k', linewidths=0
    )

ax.set_xlabel(f"PC{random_eig_idx[0]+1}")
ax.set_ylabel(f"PC{random_eig_idx[1]+1}")
ax.set_zlabel(f"PC{random_eig_idx[2]+1}")
ax.set_title('KMeans Clustering with PCA (Random t=3)')
ax.legend()
plt.show()
```


KMeans Clustering with PCA (Random t=3)
Digit 0
Digit 1
Digit 2
Digit 3
Digit 4
Digit 5
Digit 6
Digit 7
Digit 8
Digit 9
PC9
PC11
PC2

In [7]:
```
# Select 3 random eigenvalues from the top 20
random_eig_idx = np.random.choice(range(20), size=3, replace=False)
random_eig_vecs = eigenvectors[:, random_eig_idx]

# Project the dataset onto the 3 random principal components
X_pca_random = np.dot(X, random_eig_vecs)

# Plot the data in 3D with shape and color markers
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')

for i in range(10):
    marker = markers[i] if i < 4 else 'h'
    ax.scatter(
        X_pca_random[y == i, 0], X_pca_random[y == i, 1], X_pca_random[y == i,
        marker=marker, s=40, c=colors[i], alpha=0.5, label=f"Digit {i}"
    )

for i in range(10):
    ax.scatter(
        X_pca_random[y_pred == i, 0], X_pca_random[y_pred == i, 1], X_pca_rand
        marker='o', s=40, c=colors[i], alpha=0.5, edgecolors='k', linewidths=0
    )

ax.set_xlabel(f"PC{random_eig_idx[0]+1}")
ax.set_ylabel(f"PC{random_eig_idx[1]+1}")
ax.set_zlabel(f"PC{random_eig_idx[2]+1}")
ax.set_title('KMeans Clustering with PCA (Random t=3)')
ax.legend()
plt.show()
```


KMeans Clustering with PCA (Random t=3)
Digit 0
Digit 1
Digit 2
Digit 3
Digit 4
Digit 5
Digit 6
Digit 7
Digit 8
Digit 9
PC5
PC11
PC13

In [7]: