SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE,
STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

IZRADA APLIKACIJE ZA OBRADU REZULTATA 3D SKENIRANJA IZ OBLAKA TOČAKA I PRIPREMA MODELA ZA 3D ISPIS

Mijo Jurić-Pešić

Split, Rujan 2024

ZADATAK ZAVRŠNOG RADA

NASLOV: Izrada aplikacije za obradu rezultata 3D skeniranja iz oblaka točaka i priprema modela za 3D ispis

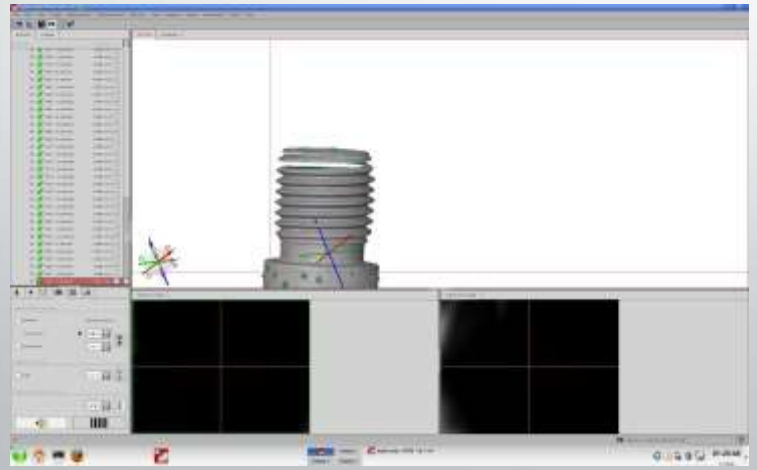
ZADATAK: Objasniti postupak 3D skeniranja strukturiranim svjetlom. Objasniti postupak 3D ispisa FFF tehnologijom te oblik potrebne datoteke (stl,...) za postojeće aplikacije za generiranje g-koda. Za jedan primjer napraviti postupak 3D skeniranja i dobivanje 3D geometrije kao oblaka točaka. Napisati skriptu za pripremu oblaka točaka te izvoz u prikladni geometrijski format za 3D ispis.

Datum obrane: 17.09.2024.

SADRŽAJ

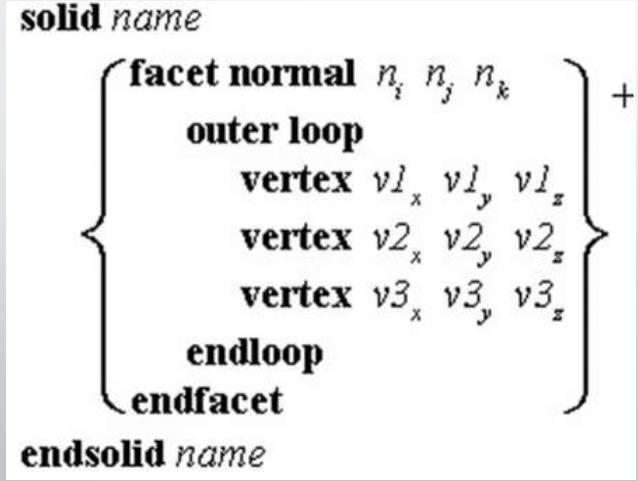
1.1 Općenito o 3D skeniranju

- Koristimo 3D skener sa strukturnim svjetlom i 2 kamere
- Prije samog skeniranja potrebno je pravilno pripremiti predmet
- Skeniranje obavljamo u više mjerenja
- Nakon skeniranja slijedi proces poligonizacije



1.2 STL format datoteke

- STL → Standard Triangle Language ili STereoLithography)
- Format trokutastih poligona
- ASCII ili binarni format
- Sastoji se od 3 vrha (3D koordinate) i vektora normale



2.1 Izrada softvera za plotanje STL datoteke

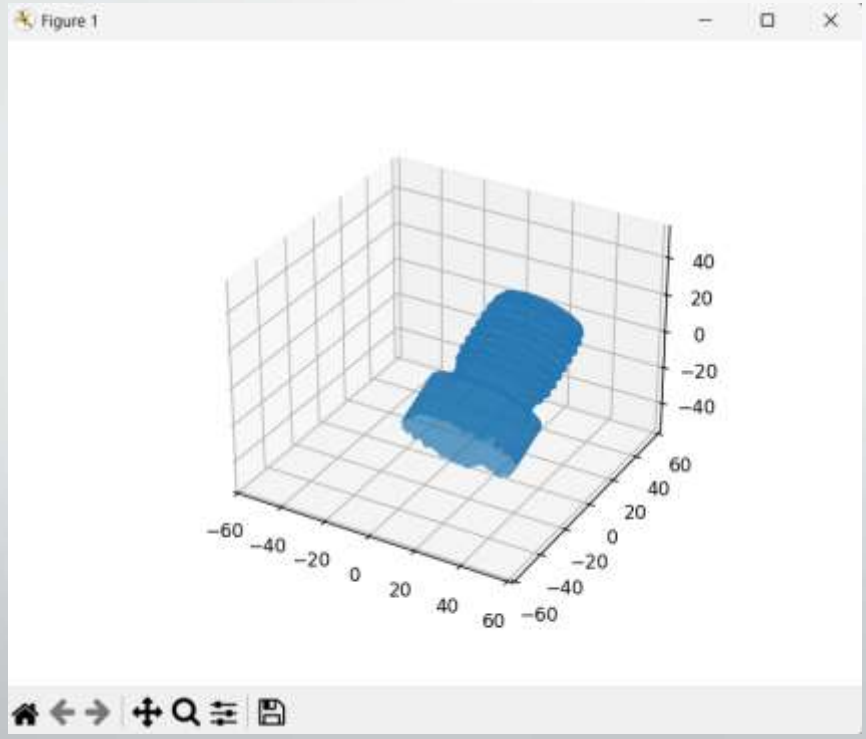
- Provjeravamo da li je STL datoteka pravilno skenirana izradom kratke Python skripte za 3D ploting
- Koraci izrade kôda:

1. Uključivanje biblioteka
2. Kreiranje novog 3D plota
3. Učitavanje STL datoteke (cijelog mesha i svih vektora)
4. Skaliranje koordinatnih osi
5. Prikaz plota

```
5   # Kreiranje novog plot-a
6   figure = pyplot.figure()
7   axes = figure.add_subplot(projection='3d')
8
9   # Učitavanje STL filea i dodavanje vektora u plot
10  vijak_mesh = mesh.Mesh.from_file('STL files/vijak2_2_8_zakrpano.stl')
11  axes.add_collection3d(mplot3d.art3d.Poly3DCollection(vijak_mesh.vectors))
12
13  # Skalranje predmeta sa osima
14  scale = vijak_mesh.points.flatten()
15  axes.auto_scale_xyz(scale, scale, scale)
16
17  # Prikaz plota na ekran
18  pyplot.show()
```

2.1 Izrada softvera za plotanje STL datoteke

- Rezultat dane napisane skripte



2.2 Izrada softvera za detekciju šupljina u STL datoteci

- Glavni dio ovog završnog rada
- Cilj: pronaći sve nepravilnosti tj. šupljine u danoj STL datoteci
- Šupljine ćemo pronaći koristeći svojstvo zatvorenosti predmeta u STL datoteci
- Svojstvo zatvorenosti – svaki rub se nalazi između točno dva trokuta, što znači da nema nigdje ruba koji bi okruživao šupljinu
- Detekciju šupljina obavljamo iteriranjem po svim rubovima svakog trokuta i brojimo koliko se puta određeni rub pojavio
- Ako je broj ponavljanja 2 – rub se nalazi između 2 trokuta
- Ako je broj ponavljanja 1 – taj rub okružuje šupljinu
- Kroz iduće slajdove ćemo detaljno proučiti kôd

2.2 Izrada softvera za detekciju šupljina u STL datoteci

1. Definiramo funkciju za pronalazak šupljina čije će ulazni kojoj ćemo proslijediti sve trokute iz STL datoteke, a kao rezultat vraća sve rubove koji okružju šupljinu (bound_edges)
2. Definiramo rječnik edges{} u koji ćemo pohraniti sve rubove
3. Iteriramo po svim rubovima te ih spremamo sortirane radi mogućnosti pronalaska

```python
# Funkcija za detekciju šupljina
def find_holes(triangles):
    # Pronalazak svih rubova sa jednim "rubom"
    edges = {}
    for triangle in triangles:
        for i in range(3):
            edge = tuple(sorted((tuple(triangle[i]), tuple(triangle[(i + 1) % 3]))))
            if edge in edges:
                edges[edge] += 1
            else:
                edges[edge] = 1

    # Pronalaženje rubova koje se pojavljuju samo jednom, tj. šupljina
    bound_edges = [edge for edge, count in edges.items() if count == 1]

    return bound_edges
```

2.2 Izrada softvera za detekciju šupljina u STL datoteci

4. Kada naiđemo na novi rub, spremamo ga u rječnik sa brojem ponavljanja 1 (linija 23)
5. Ako iterirajući naiđemo na neki rub više od jednom, njegov broj ponavljanja povećamo za 1 (linija 21)
6. Definiramo bound_edges tj. sve rubove koji se ponavljaju jednom na način da prođemo kroz sve rubove te ih spremimo u bound_edges ako je count (broj ponavljanja) tog ruba jednak 1

```
13  # Funkcija za detekciju šupljina
14  def find_holes(triangles):
15      # Pronalazak svih rubova sa jednim "rubom"
16      edges = {}
17      for triangle in triangles:
18          for i in range(3):
19              edge = tuple(sorted((tuple(triangle[i]), tuple(triangle[(i + 1) % 3]))))
20              if edge in edges:
21                  edges[edge] += 1
22              else:
23                  edges[edge] = 1
24
25      # Pronalaženje rubova koje se pojavljuju samo jednom, tj. šupljina
26      bound_edges = [edge for edge, count in edges.items() if count == 1]
27
28      return bound_edges
```

3. PLOTANJE STL DATOTEKE SA PRONAĐENIM ŠUPLJINAMA

- Kada smo pronašli sve rubove koji okružuju šupljinu, potrebno je nacrtati 3D plotati originalnu STL datoteku (metalni vijak) sa pronađenim šupljinama
- Originalan vijak ćemo prikazati u plavoj, a sve rubove koji okružuju šupljine u crvenoj boji
- Funkcija plot_screw_and_holes(triangles, holes) će imati 2 argumenta
- Triangles – svi učitani trokuti iz STL datoteke
- Holes – rezultat funkcije find_holes() tj. svi rubovi koji okružuju šupljinu
- Koraci:

1. Definiranje novog 3D plota
2. Crtanje svih trokuta u plavoj boji
3. Iteriramo po svim rubovima koje prikazujemo u crvenoj boji izvlačeći njihove X, Y I Z koordinate
4. Odradimo skaliranje kako bismo cijeli vijak prikazali u proporcijama

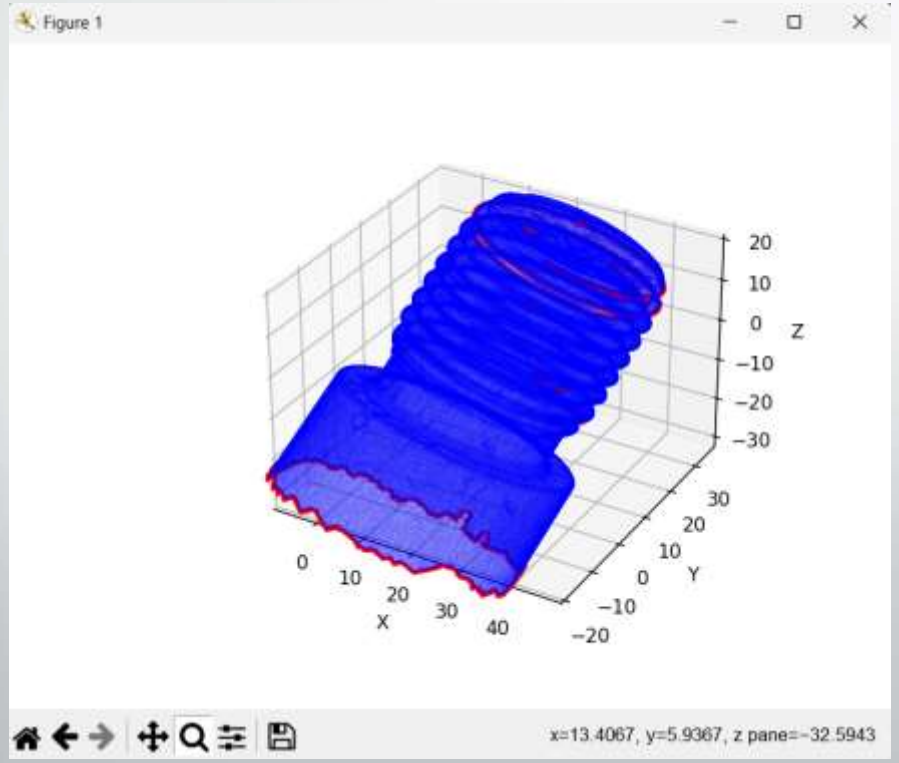
3. PLOTANJE STL DATOTEKE SA PRONAĐENIM ŠUPLJINAMA

- Prikaz kôda funkcije plot_screw_and_holes()

```
# Funkcija za crtanje STL modela i šupljina
def plot_screw_and_holes(triangles, holes):
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')

    # Crtanje svih trokuta u plavoj boji tako da vijak bude plave boje
    ax.add_collection3d(Poly3DCollection(triangles, color='blue', alpha=0.3, linewidths=0.2, edgecolor='b'))

    # Crtanje šupljina tj. rubnih rubova sa crvenim rubovima kako bi vidjeli razliku
    for edge in holes:
        edge = np.array(edge)
        ax.plot(edge[:, 0], edge[:, 1], edge[:, 2], color='red', linewidth=2)

    # Podesavanje koordinatnih osiju
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')

    # Skaliranje
    scale = triangles.flatten(order='C')
    ax.auto_scale_xyz(scale, scale, scale)

    plt.show()

# Prikazivanje modela sa vijka sa šupljinama
plot_screw_and_holes(triangles, holes)
```

3. PLOTANJE STL DATOTEKE SA PRONAĐENIM ŠUPLJINAMA

- Rezultat funkcije plot_screw_and_holes()



4. ZAKLJUČAK

- Koristeći 3D skener sa strukturnim svjetlom i dvije kamere moguće je precizno skeniranje jednostavnijih i složenijih modela
- STL format datoteke je format u kojem je 3D objekt pohranjen kao niz trokuta definiran sa 3 vrha i vektorom normale
- STL datoteku daljnjom obradom možemo pripremiti za ispis
- Koristeći svojstvo zatvorenosti STL datoteke, jednostavnim iteriranjem po rubovima objekta možemo pronaći šupljine koje su ostale kao posljedica nepravilnog skeniranja ili su svojstvo tog objekta
- 3D skeniranje i ispis su danas često korištene tehnologije čiji se alati i tehnike neprestano razvijaju i primjenu pronalaze u sve više sfera ljudskog života

HVALA NA PAŽNJI!
Imate li pitanja?