

BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM  
GÉPÉSZMÉRNÖKI KAR  
MŰSZAKI MECHANIKAI TANSZÉK

Mátraházi Krisztián Márk

Nyílt forráskódú mozgás- és deformációelemző szoftver  
fejlesztése Python környezetben

TDK dolgozat

*Melléklet*

2024/25 őszi félév

Konzulens: Dr. Kossa Attila, egyetemi docens

# Tartalomjegyzék

|                                                                                 |          |
|---------------------------------------------------------------------------------|----------|
| <b>Függelék</b>                                                                 | <b>1</b> |
| A <code>Node</code> és az <code>Element</code> osztályok konstruktora . . . . . | 1        |
| A főprogram, a <code>main.py</code> . . . . .                                   | 2        |
| A virtuális tesztvideók létrehozását végző kód . . . . .                        | 5        |
| A mozgáskövető algoritmus tesztvideós teljesítménye . . . . .                   | 9        |
| A főprogram használatának szemléltetése . . . . .                               | 10       |

# Függelék

## A Node és az Element osztály konstruktora

Az alábbiakban olvasható a háló adatstruktúrája. A csomópontokat a globális  $x, y$  koordinátáik, a globális sorszámuk, illetve az aktuális képkocka sorszáma alapján hozzuk létre a hálózás során. Emellett mindegyikre vonatkozóan eltároljuk a számított elmozdulás-, alakváltozás-, illetve feszültségértékeket. A vizsgált tartomány belsejében elhelyezkedő csomópontokban az alakváltozási-, illetve feszültségvektorok négy-négy eltérő értéket vesznek fel. Mindezeket külön tároljuk, azonban csomóponti átlagoláskor az `_AVG` végződésű változók is felhasználásra kerülnek.

Látható, hogy egy végelem objektum létrehozásakor elmentjük az elem globális sorszámát, illetve az aktuális képkocka számát, azonban a kapcsolódó négy csomóponttra is szükségünk van. Az adott elemen belül interpolált különféle mezőváltozók tárolását is objektum-orientáltan végezzük.

```
1 class Node:
2     def __init__(self, x_abs, y_abs, glob_nr: int, frame_nr: int):
3         self.XY = np.array([x_abs, y_abs])
4         self.glob_nr = glob_nr
5         self.frame_nr = frame_nr
6         self.disp: np.array = None
7         self.strain_NW: np.array = None
8         self.strain_NE: np.array = None
9         self.strain_SW: np.array = None
10        self.strain_SE: np.array = None
11        self.strain_AVG: np.array = None
12        self.stress_NW: np.array = None
13        self.stress_NE: np.array = None
14        self.stress_SW: np.array = None
15        self.stress_SE: np.array = None
16        self.stress_AVG: np.array = None
17
18 class Element:
19     def __init__(self, nr_of_element: int, local1: Node, local2: Node,
20                  local3: Node, local4: Node, frame_nr: int):
21         self.nr_of_element = nr_of_element
22         self.local1 = local1
23         self.local2 = local2
24         self.local3 = local3
```

```

24     self.local4 = local4
25     self.frame_nr = frame_nr
26     self.u_xi_eta: np.array = None
27     self.v_xi_eta: np.array = None
28     self.epsx_xi_eta: np.array = None
29     self.etsy_xi_eta: np.array = None
30     self.gamxy_xi_eta: np.array = None
31     self.sigx_xi_eta: np.array = None
32     self.sigy_xi_eta: np.array = None
33     self.tauxy_xi_eta: np.array = None
34     self.disp_field_2: np.array = None

```

## A főprogram, a main.py

Ebben a fejezetben olvasható a főprogram, amelynek paramétereit és változóit a saját videófelvételünk adottságai alapján, illetve a vizsgálati igényeinket figyelembe véve szükséges beállítanunk.

```

1  # Headers
2  import h_video_prep as vp
3  import h_CORE_LOGIC as core
4  import h_visualization as vis
5  from h_grid_motion_tracking import grid_motion_tracking
6  # Python Libraries
7  import time
8  import copy
9  import cv2
10
11  if __name__ == "__main__":
12      ##### Input video preparation #####
13      video_path = "DIC_Video_1.mp4"
14      frames = vp.extract_frames(video_path, save_every_nth_frame = 20)
15      print(f"Number of frames extracted: {len(frames)}")
16
17      # output after framerate reduction
18      output_path = "output.mp4"
19      vp.create_video(frames, output_path, fps=1) # set fps as needed
20
21      # watch the video
22      video_path = "output.mp4"
23      window_desired_width = 1700

```

```

24 vp.play_video(video_path, window_desired_width)
25
26 ##### Motion tracking & Grid parameters #####
27 # the size in pixel of the correlation windows
28 win_size_px = (200,200)
29 lk_parameters = dict(winSize = win_size_px, maxLevel = 5,
30                       criteria = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT,
31                                  10, 0.02))
32 # the size in pixel (dx,dy) of the correlation grid
33 grid_size_px = (150,150)
34 result_file = "grid_points_in_time.dic"
35 # Pick your area of interest in the OpenCV window
36 grid_motion_tracking(frames, grid_size_px, result_file,
37                      window_desired_width, lk_parameters)
38
39 start_time = time.time()      # measure execution time (roughly)
40
41 ### Parameters of the resulting images images and videos ###
42 colorbar_x0 = 2500           # left hand side x coordinate
43 colorbar_y0 = 500           # top y coordinate
44
45 image_params = {
46     # calculate your monitor's dpi first, (https://dpi.lv/)
47     "dpi": 100,
48     # colorbar params
49     "colorbar_resize_factor": 2.5,
50     "latex_font": False,
51     "nr_of_ticks": 5,
52     "fontsize": 26,
53     "alpha": 0.75,
54 }
55
56 ### Calculation of the displacement and strain fields ###
57 rm_rigid_body_transform = False
58 assume_rectangles = True
59 avg_nodal_strain = False
60 # number of interpolation points along one edge of an element
61 interp_res = 10
62
63 ##### Calculation of the stress field (optional) #####
64 Stress_Calc = False
65 stress_state = 'plane_stress'

```

```

63 Young = 7e4      # check unit! (default: Pa, supports MPa)
64 unit = 'MPa'
65 Poisson = 0.32
66 avg_nodal_stress = False
67 von_Mises = True
68
69 core.calculate_disp_strain(result_file, rm_rigid_body_transform,
    assume_rectangles, avg_nodal_strain, Stress_Calc, stress_state,
    Young, Poisson, avg_nodal_stress, unit, interp_res,
    image_params=image_params)
70
71 if Stress_Calc:
72     core.calculate_stress(stress_state, Young, Poisson,
    avg_nodal_stress, interp_res, image_params, unit, von_Mises)
73
74 ##### Visualization of the results #####
75 # DISP
76 vis.visualize_disp_field_u_component(copy.deepcopy(frames),
    assume_rectangles, colorbar_x0, colorbar_y0, fps=1)
77 vis.visualize_disp_field_v_component(copy.deepcopy(frames),
    assume_rectangles, colorbar_x0, colorbar_y0, fps=1)
78
79 # STRAIN
80 vis.visualize_strain_field_epsx_component(copy.deepcopy(frames),
    assume_rectangles, colorbar_x0, colorbar_y0, fps=1)
81 vis.visualize_strain_field_epsy_component(copy.deepcopy(frames),
    assume_rectangles, colorbar_x0, colorbar_y0, fps=1)
82 vis.visualize_strain_field_gamxy_component(copy.deepcopy(frames),
    assume_rectangles, colorbar_x0, colorbar_y0, fps=1)
83
84 # STRESS
85 if Stress_Calc:
86     vis.visualize_stress_field_sigx_component(copy.deepcopy(frames)
    ), assume_rectangles, colorbar_x0, colorbar_y0, fps=1)
87 vis.visualize_stress_field_sigy_component(copy.deepcopy(frames)
    ), assume_rectangles, colorbar_x0, colorbar_y0, fps=1)
88 vis.visualize_stress_field_tauxy_component(copy.deepcopy(
    frames), assume_rectangles, colorbar_x0, colorbar_y0, fps=1)
89
90 if von_Mises:
91     vis.visualize_stress_field_sig_vM(copy.deepcopy(frames),
    assume_rectangles, colorbar_x0, colorbar_y0, fps=1)

```

```

92     end_time = time.time()
93     elapsed_time = end_time - start_time
94     print(f"Elapsed time: {elapsed_time: .2f} seconds")
95     print('DONE')

```

## A virtuális tesztvideók létrehozását végző kód

Jelen fejezetben a virtuális tesztvideók készítéséért felelős kód olvasható. Látható, hogy a pöttyök méretében, szürkeárnyaltos intenzitásában, illetve elmozdulásában is megjelenik a véletlenszerűség. Az itt szereplő beállítás szerint a mentett koordinátájú pontok mindkét irányban 20 pixeles osztásközzel jönnek létre, míg a háttérrel képező pöttyök 5 pixelenként generálódnak.

Az első `for` ciklusban történik a véletlenszerű elmozdulások generálása, a nagyobb osztásközzel rendelkező ponthalmaz időfüggő koordinátáinak fájlba mentése, illetve az egyes képkockák ábrázolása. Ezután következik a valós felvétel zajának modellezése, majd az elkészült 10 képkockából egy 10 másodperces, 24 FPS-es videót hozunk létre.

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3  import cv2
4
5  def add_gaussian_noise(image, mean=0, sigma=25):
6      row, col, ch = image.shape
7      gauss = np.random.normal(mean, sigma, (row, col, ch))
8      gauss = gauss.reshape(row, col, ch)
9      noisy = image + gauss
10     return np.clip(noisy, 0, 255).astype(np.uint8)
11
12     # points being tracked, init
13     x_range = np.arange(100, 701, 20)
14     y_range = np.arange(100, 601, 20)
15     x_grid, y_grid = np.meshgrid(x_range, y_range)
16     grid_height = x_grid.shape[0]
17     grid_width = x_grid.shape[1]
18
19     DPI = 130
20     MIN_SIZE = 1
21     MAX_SIZE = 10
22     MIN_INTENSITY = 0.0    # black
23     MAX_INTENSITY = 0.8

```

```

24 NR_of_frames = 10
25
26 random_sizes = np.random.uniform(MIN_SIZE, MAX_SIZE, x_grid.size)
27 random_intensities = np.random.uniform(MIN_INTENSITY, MAX_INTENSITY,
28     x_grid.size)
29 colors = [(intensity, intensity, intensity) for intensity in
30     random_intensities]
31 x_grid_mx = np.empty((x_grid.shape[0], x_grid.shape[1], NR_of_frames))
32 y_grid_mx = np.empty((y_grid.shape[0], y_grid.shape[1], NR_of_frames))
33 x_grid_mx[:, :, 0] = x_grid
34 y_grid_mx[:, :, 0] = y_grid
35
36 # BACKGROUND init
37 x_range_bckgr = np.arange(100, 701, 5)
38 y_range_bckgr = np.arange(100, 601, 5)
39 x_grid_bckgr, y_grid_bckgr = np.meshgrid(x_range_bckgr, y_range_bckgr)
40
41 random_offsets_x = np.random.uniform(-4, 4, size=x_grid_bckgr.shape)
42 random_offsets_y = np.random.uniform(-4, 4, size=y_grid_bckgr.shape)
43 new_x_grid_bckgr = x_grid_bckgr + random_offsets_x
44 new_y_grid_bckgr = y_grid_bckgr + random_offsets_y
45
46 random_sizes_bckgr = np.random.uniform(MIN_SIZE, MAX_SIZE,
47     x_grid_bckgr.size)
48 random_intensities_bckgr = np.random.uniform(MIN_INTENSITY,
49     MAX_INTENSITY, x_grid_bckgr.size)
50 colors_bckgr = [(intensity, intensity, intensity) for intensity in
51     random_intensities_bckgr]
52 x_grid_mx_bckgr = np.empty((x_grid_bckgr.shape[0], x_grid_bckgr.shape
53     [1], NR_of_frames))
54 y_grid_mx_bckgr = np.empty((y_grid_bckgr.shape[0], y_grid_bckgr.shape
55     [1], NR_of_frames))
56 x_grid_mx_bckgr[:, :, 0] = new_x_grid_bckgr
57 y_grid_mx_bckgr[:, :, 0] = new_y_grid_bckgr
58
59 for k in range(0, NR_of_frames):
60     if k == 0:
61         new_x_grid = x_grid_mx[:, :, 0].reshape((x_grid.shape[0], x_grid
62             .shape[1]))
63         new_y_grid = y_grid_mx[:, :, 0].reshape((y_grid.shape[0], y_grid
64             .shape[1]))

```

```

57 else:
58     # generate and apply random offsets
59     random_offsets_x = np.random.uniform(-1, 1, size=x_grid.shape)
60     random_offsets_y = np.random.uniform(-1, 1, size=y_grid.shape)
61     new_x_grid = x_grid_mx[:,:,:,(k-1)].reshape((x_grid.shape[0],
62         x_grid.shape[1])) + random_offsets_x
63     new_y_grid = y_grid_mx[:,:,:,(k-1)].reshape((y_grid.shape[0],
64         y_grid.shape[1])) + random_offsets_y
65
66 x_grid_mx[:,:,:k] = new_x_grid
67 y_grid_mx[:,:,:k] = new_y_grid
68
69 coordinates_string = ""
70 for j in range(grid_height):    # rows
71     for i in range(grid_width):    # columns
72         coordinates_string += f'{x_grid_mx[j,i,k]}\t{y_grid_mx[j,i
73             ,k]}\t'    # large string
74
75 # BACKGROUND
76 if k == 0:
77     with open(f'testdata_coordinates.dic', "w") as file:
78         file.write(f'{x_range.min()}\t'+f'{x_range.max()}\t'+f'\n'
79             +f'{len(x_range)}\n')
80         file.write(f'{y_range.min()}\t'+f'{y_range.max()}\t'+f'\n'
81             +f'{len(y_range)}\n')
82         file.write(f'img{k+1}\t'+coordinates_string+'\n')
83
84 new_x_grid_bckgr = x_grid_mx_bckgr[:,:,:0].reshape((
85     x_grid_bckgr.shape[0], x_grid_bckgr.shape[1]))
86 new_y_grid_bckgr = y_grid_mx_bckgr[:,:,:0].reshape((
87     y_grid_bckgr.shape[0], y_grid_bckgr.shape[1]))
88 else:
89     with open(f'testdata_coordinates.dic', "a") as file:
90         file.write(f'img{k+1}\t'+coordinates_string+'\n')
91     # generate and apply random offsets
92     random_offsets_x_bckgr = np.random.uniform(-1, 1, size=
93         x_grid_bckgr.shape)
94     random_offsets_y_bckgr = np.random.uniform(-1, 1, size=
95         y_grid_bckgr.shape)
96     new_x_grid_bckgr = x_grid_mx_bckgr[:,:,:,(k-1)].reshape((
97         x_grid_bckgr.shape[0], x_grid_bckgr.shape[1])) +
98         random_offsets_x_bckgr

```

```

88         new_y_grid_bckgr = y_grid_mx_bckgr[:, :, (k-1)].reshape((
            y_grid_bckgr.shape[0], y_grid_bckgr.shape[1])) +
            random_offsets_y_bckgr
89
90     x_grid_mx_bckgr[:, :, k] = new_x_grid_bckgr
91     y_grid_mx_bckgr[:, :, k] = new_y_grid_bckgr
92
93     # plot
94     fig, ax = plt.subplots()
95     ax.set_axis_off()
96     ax.set_aspect('equal')
97     plt.figure(figsize=(8, 7))
98     plt.scatter(new_x_grid, new_y_grid, color=colors, s=random_sizes)
99     plt.scatter(new_x_grid_bckgr, new_y_grid_bckgr, color=colors_bckgr
        , s=random_sizes_bckgr)
100    plt.gca().set_aspect('equal', adjustable='box')
101    plt.gca().invert_yaxis() # invert y-axis to match the coord. sys.
102    plt.xlim(0, 800)
103    plt.ylim(0, 700)
104    plt.gca().set_facecolor('white')
105    plt.axis('off')
106    plt.savefig(f'points{k+1}.jpg', format='jpg', dpi=DPI, bbox_inches
        ='tight', pad_inches=0)
107    plt.close()
108
109    # duplicate each frame 24 times to get 24fps + add noise
110    frames = []
111    for i in range(10):
112        frame = cv2.imread(f'points{i+1}.jpg')
113        noisy_frame = add_gaussian_noise(frame, mean=0, sigma=25)
114        cv2.imwrite(f'gauss{i+1}.jpg', noisy_frame)
115        for _ in range(24):
116            frames.append(noisy_frame)
117
118    # 10 seconds at 24 fps
119    num_frames_to_keep = 10 * 24
120    frames = frames[:num_frames_to_keep]
121
122    # create an .mp4 video
123    height, width, _ = frames[0].shape
124    fourcc = cv2.VideoWriter_fourcc(*'mp4v') # codec
125    out = cv2.VideoWriter('testdata.mp4', fourcc, 24.0, (width, height))

```

```

126     for frame in frames:
127         out.write(frame)
128     out.release()

```

## A mozgáskövető algoritmus tesztvideós teljesítménye

Az alábbiakban olvasható a `testdata_coordinates.dic` és `testresults.dic` fájlok összehasonlítását végző kód, amely meghatározza a virtuális tesztvideón generált koordináták, illetve az optical flow algoritmus által nyújtott koordináták különbségét pixelben. Az utolsó sorok segítségével a tetszőlegesen állítható `'threshold'` küszöbértéknél nagyobb különbségek számossága íratható ki a konzolra.

```

1  import numpy as np
2  # read testdata
3  def read_and_parse_testdata(filename):
4      data = []
5      with open(filename, 'r') as file:
6          line_num = 0
7          for line in file:
8              line_num += 1
9              if line_num >= 3:
10                 values = line.strip().split('\t')[1:]
11                 data.append([float(value) for value in values])
12     return np.array(data)
13  # read the results coming from 'main.py'
14  def read_and_parse_results(filename):
15      data = []
16      with open(filename, 'r') as file:
17          line_num = 0
18          for line in file:
19              line_num += 1
20              if line_num != 6 and line_num != 12:
21                 values = line.strip().split('\t')[1:]
22                 data.append([float(value) for value in values])
23     return np.array(data)
24  testdata = read_and_parse_testdata('testdata_coordinates.dic')
25  testresult = read_and_parse_results('testresults.dic')
26  difference = np.abs(testresult - testdata)
27  np.savetxt('difference.dic', difference, delimiter='\t', fmt='%.3e',
28             header='', comments='')

```

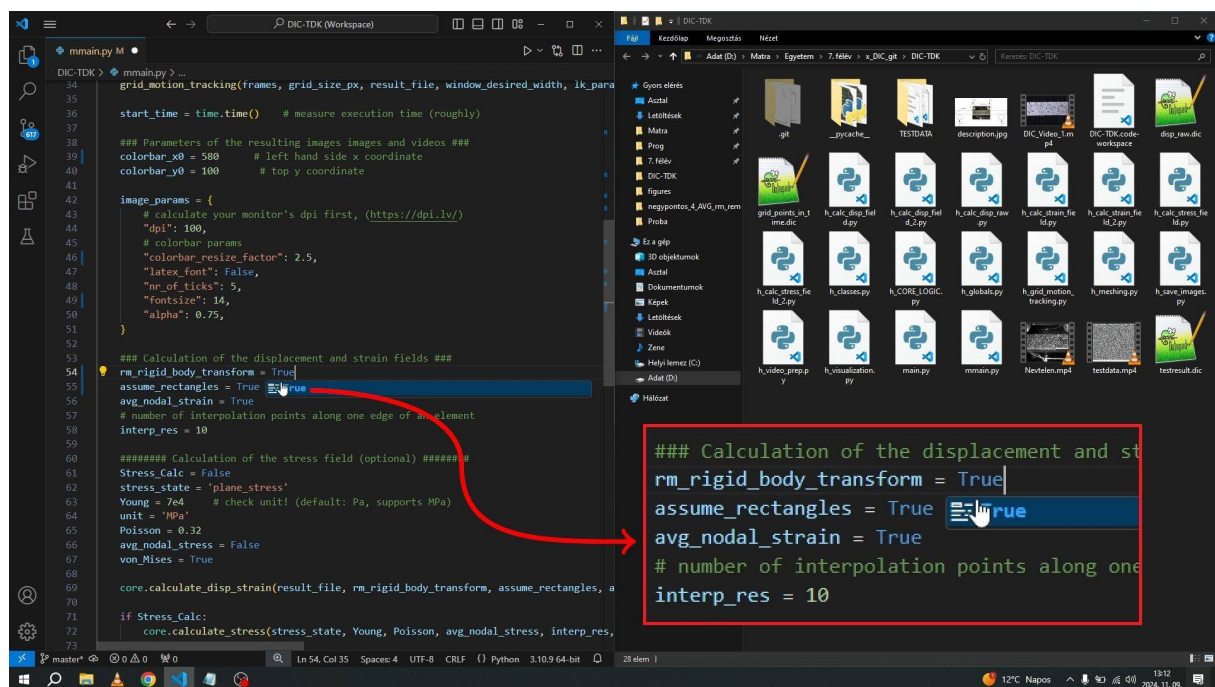
```

29  ### EVALUATION ###
30  print(f'total points (nodes) tracked: = {difference.size}')
31  for i in range(3,6):
32      count = np.sum(difference > i)
33      print(f'difference above {i} px: {count}')
34  threshold = 10
35  count = np.sum(difference > threshold)
36  print(f'difference above {threshold} px: {count}')

```

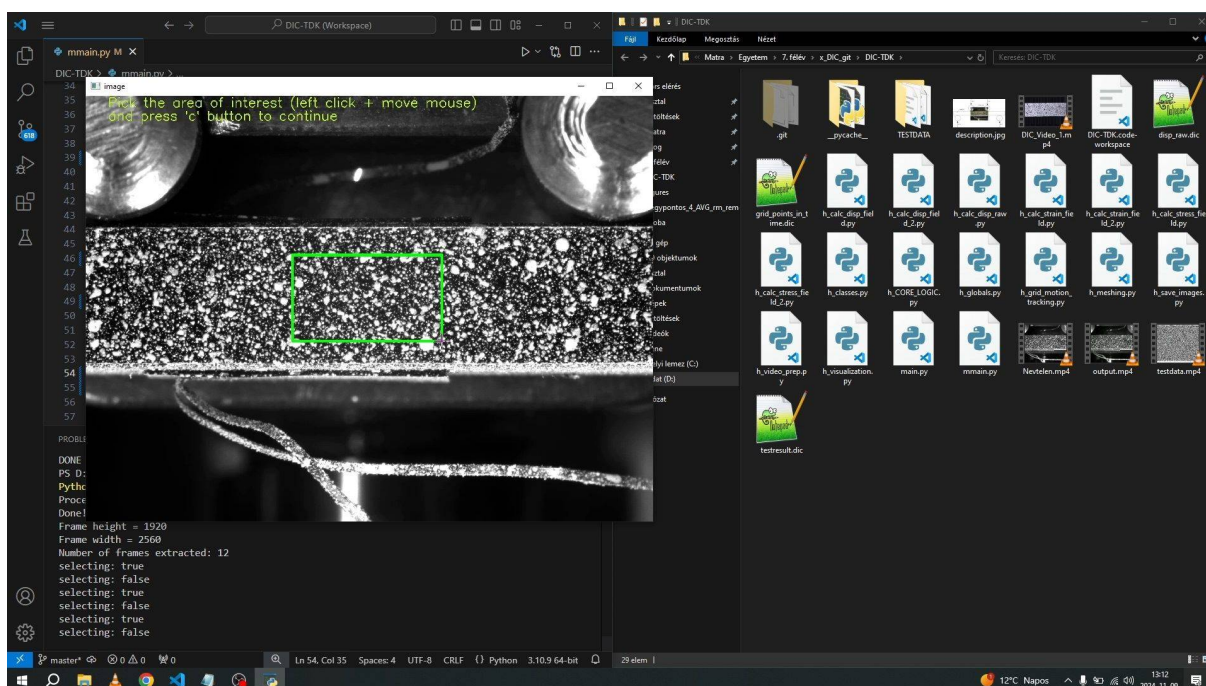
## A főprogram használatának szemléltetése

A jelen fejezetben bemutatott képernyőképek a főprogram használatának kulcslépéseit illusztrálják. A szoftver adott bemeneti videófelvétélhez és felhasználói igényekhez történő adaptálását illusztrálja a 8.5.1 ábra.



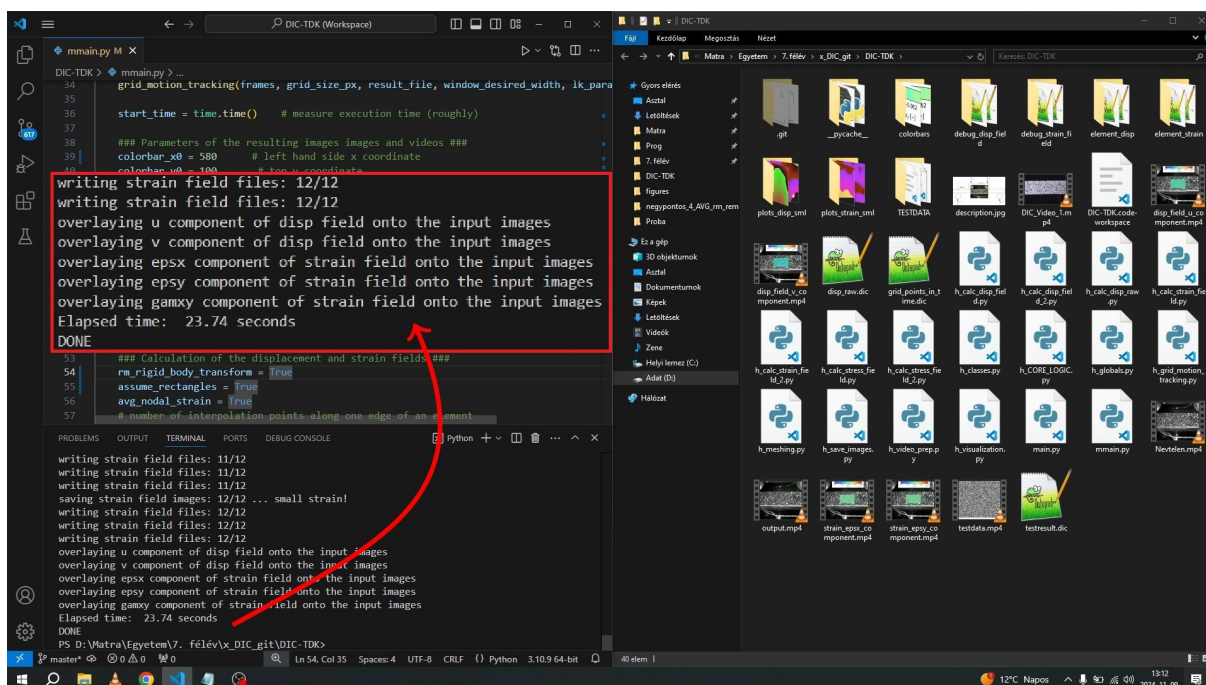
8.5.1. ábra: Különbféle paraméterek és változók beállítása a főprogramban.

Az elemezni kívánt területet a bal egérgomb nyomva tartásával jelölhetjük ki. A visszajelzésről egy zöld színű, téglalap alakú keret gondoskodik, a 8.5.2 ábra szerint.



8.5.2. ábra: A vizsgálni kívánt terület kijelölése a felugró ablakban.

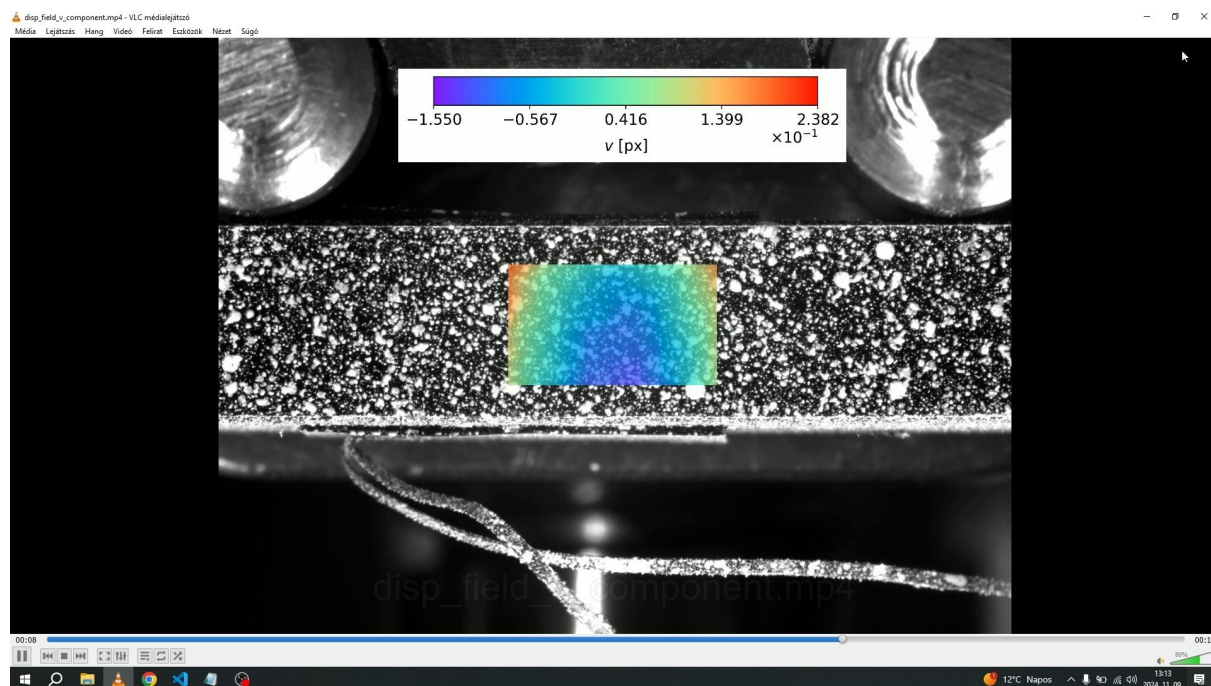
A szoftver futtatása közben a konzolablakon tájékozódhatunk a program állapotáról (lásd 8.5.3 ábra), ahol néhány részeredmény is elérhető a felhasználó számára. Ilyenek például a számolt mezőváltozók teljes mozgásra vonatkozó szélsőértékei.



8.5.3. ábra: A részeredmények és a program állapotának nyomon követése a konzolon.

A futtatást követően a 8.5.3 ábrán megfigyelhető, hogy a munkakönyvtárban megjelentek a különféle eredményvideók, illetve az ellenőrző fájlokat tartalmazó mappák is. A

8.5.4 ábrán a vizsgált terület  $v(x, y)$  mezeje látszik a teljes mozgás kétharmadánál.



8.5.4. ábra: A kívánt mezőváltozót ábrázoló eredményvideó megtekintése.

A felhasználó többféle szövegfájl segítségével ellenőrizheti a számított eredmények helyességét. A 8.5.5 ábra az  $\varepsilon_x(x, y)$  mező tizedik képkockára vonatkozó értékeit mutatja.

|    |             |             |             |             |             |             |             |             |             |             |
|----|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 1  | -7.9441e-04 | -7.8268e-04 | -7.7095e-04 | -7.5922e-04 | -7.4748e-04 | -7.3575e-04 | -7.2402e-04 | -7.1229e-04 | -7.0055e-04 | -6.8882e-04 |
| 2  | -7.5576e-04 | -7.4409e-04 | -7.3243e-04 | -7.2076e-04 | -7.0910e-04 | -6.9743e-04 | -6.8577e-04 | -6.7411e-04 | -6.6244e-04 | -6.5078e-04 |
| 3  | -7.1710e-04 | -7.0551e-04 | -6.9391e-04 | -6.8231e-04 | -6.7072e-04 | -6.5912e-04 | -6.4752e-04 | -6.3593e-04 | -6.2433e-04 | -6.1273e-04 |
| 4  | -6.7845e-04 | -6.6692e-04 | -6.5539e-04 | -6.4386e-04 | -6.3233e-04 | -6.2080e-04 | -6.0927e-04 | -5.9774e-04 | -5.8622e-04 | -5.7469e-04 |
| 5  | -6.3979e-04 | -6.2833e-04 | -6.1687e-04 | -6.0541e-04 | -5.9395e-04 | -5.8249e-04 | -5.7102e-04 | -5.5956e-04 | -5.4810e-04 | -5.3664e-04 |
| 6  | -6.0114e-04 | -5.8974e-04 | -5.7835e-04 | -5.6696e-04 | -5.5556e-04 | -5.4417e-04 | -5.3278e-04 | -5.2138e-04 | -5.0999e-04 | -4.9860e-04 |
| 7  | -5.6248e-04 | -5.5115e-04 | -5.3983e-04 | -5.2850e-04 | -5.1718e-04 | -5.0585e-04 | -4.9453e-04 | -4.8320e-04 | -4.7188e-04 | -4.6055e-04 |
| 8  | -5.2382e-04 | -5.1257e-04 | -5.0131e-04 | -4.9005e-04 | -4.7879e-04 | -4.6754e-04 | -4.5628e-04 | -4.4502e-04 | -4.3376e-04 | -4.2251e-04 |
| 9  | -4.8517e-04 | -4.7398e-04 | -4.6279e-04 | -4.5160e-04 | -4.4041e-04 | -4.2922e-04 | -4.1803e-04 | -4.0684e-04 | -3.9565e-04 | -3.8446e-04 |
| 10 | -4.4651e-04 | -4.3539e-04 | -4.2427e-04 | -4.1315e-04 | -4.0202e-04 | -3.9090e-04 | -3.7978e-04 | -3.6866e-04 | -3.5754e-04 | -3.4641e-04 |
| 11 | -4.0785e-04 | -3.9673e-04 | -3.8561e-04 | -3.7449e-04 | -3.6337e-04 | -3.5225e-04 | -3.4113e-04 | -3.3001e-04 | -3.1889e-04 | -3.0777e-04 |
| 12 | -3.6919e-04 | -3.5807e-04 | -3.4695e-04 | -3.3583e-04 | -3.2471e-04 | -3.1359e-04 | -3.0247e-04 | -2.9135e-04 | -2.8023e-04 | -2.6911e-04 |
| 13 | -3.3053e-04 | -3.1941e-04 | -3.0829e-04 | -2.9717e-04 | -2.8605e-04 | -2.7493e-04 | -2.6381e-04 | -2.5269e-04 | -2.4157e-04 | -2.3045e-04 |
| 14 | -2.9187e-04 | -2.8075e-04 | -2.6963e-04 | -2.5851e-04 | -2.4739e-04 | -2.3627e-04 | -2.2515e-04 | -2.1403e-04 | -2.0291e-04 | -1.9179e-04 |
| 15 | -2.5321e-04 | -2.4209e-04 | -2.3097e-04 | -2.1985e-04 | -2.0873e-04 | -1.9761e-04 | -1.8649e-04 | -1.7537e-04 | -1.6425e-04 | -1.5313e-04 |
| 16 | -2.1455e-04 | -2.0343e-04 | -1.9231e-04 | -1.8119e-04 | -1.7007e-04 | -1.5895e-04 | -1.4783e-04 | -1.3671e-04 | -1.2559e-04 | -1.1447e-04 |
| 17 | -1.7589e-04 | -1.6477e-04 | -1.5365e-04 | -1.4253e-04 | -1.3141e-04 | -1.2029e-04 | -1.0917e-04 | -9.805e-05  | -8.693e-05  | -7.581e-05  |
| 18 | -1.3723e-04 | -1.2611e-04 | -1.1499e-04 | -1.0387e-04 | -9.275e-05  | -8.163e-05  | -7.051e-05  | -5.939e-05  | -4.827e-05  | -3.715e-05  |
| 19 | -9.857e-05  | -8.745e-05  | -7.633e-05  | -6.521e-05  | -5.409e-05  | -4.297e-05  | -3.185e-05  | -2.073e-05  | -9.61e-06   | 1.5e-06     |
| 20 | 6.043e-05   | 4.931e-05   | 3.819e-05   | 2.707e-05   | 1.595e-05   | 4.83e-06    | -3.28e-06   | -1.41e-06   | -2.54e-07   | -3.67e-08   |
| 21 | 1.257e-04   | 1.2091e-04  | 1.2925e-04  | 1.3759e-04  | 1.4594e-04  | 1.5428e-04  | 1.6262e-04  | 1.7096e-04  | 1.7930e-04  | 1.8764e-04  |
| 22 | 1.1257e-04  | 1.2091e-04  | 1.2925e-04  | 1.3759e-04  | 1.4594e-04  | 1.5428e-04  | 1.6262e-04  | 1.7096e-04  | 1.7930e-04  | 1.8764e-04  |
| 23 | 1.2057e-04  | 1.2711e-04  | 1.3365e-04  | 1.4019e-04  | 1.4673e-04  | 1.5327e-04  | 1.5982e-04  | 1.6636e-04  | 1.7290e-04  | 1.7944e-04  |
| 24 | 1.2857e-04  | 1.3331e-04  | 1.3805e-04  | 1.4279e-04  | 1.4753e-04  | 1.5227e-04  | 1.5701e-04  | 1.6175e-04  | 1.6649e-04  | 1.7123e-04  |

8.5.5. ábra: A kívánt szöveges ellenőrző fájl megtekintése.