

BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
GÉPÉSZMÉRNÖKI KAR
MŰSZAKI MECHANIKAI TANSZÉK

Mátraházi Krisztián Márk

Nyílt forráskódú mozgás- és deformációelemző szoftver
fejlesztése Python környezetben

TDK dolgozat

2024/25 őszi félév

Konzulens: Dr. Kossa Attila, egyetemi docens

Tartalomjegyzék

1. Bevezetés	1
1.1. A mozgáselemzés rövid ismertetése	1
1.2. Interneten elérhető DIC megoldások	2
1.3. A dolgozat célja, tartalmi áttekintése	2
2. Mozgáskövetés	4
2.1. Az optical flow algoritmus működése	5
2.2. A Lucas-Kanade módszer	6
2.3. A Lucas-Kanade algoritmus paraméterei	8
2.4. A rácspontok elmozdulásának számítása	9
2.5. Merev testszerű mozgások kiszűrésének lehetősége	10
3. A szoftver használata	14
4. A mezőváltozók számítása, interpolációja	16
4.1. Hálózás	16
4.2. Az elmozdulásmező létrehozása	17
4.2.1. Az elemenkénti elmozdulásmező interpolációja	17
4.3. Az alakváltozási mező létrehozása	20
4.3.1. A csomóponti alakváltozások és az elemenkénti alakváltozási mező .	21
4.3.2. A csomóponti átlagolás kérdése	25
4.4. A feszültségmező létrehozása	27
4.4.1. A csomóponti feszültségek és az elemenkénti feszültségmező	27
4.4.2. Csomóponti átlagolás	30
4.4.3. A von Mises-féle egyenértékű feszültség számítása	31
5. A program kimenetei, eredményei	33
5.1. A globális mezők összeállítása, ábrázolása	33
5.2. A színes illusztrációk jelmagyarázata	35
5.3. Videófelvételek létrehozása	35
5.4. Részeredményeket tartalmazó szövegfájlok	36

5.5.	A fejlesztéshez használt videófelvétel bemutatása	37
5.6.	A fő funkciók bemutatása	37
5.6.1.	Merev testszerű mozgások kiszűrése	38
5.6.2.	Csomóponti alakváltozások átlagolása	40
5.6.3.	Csomóponti feszültségek átlagolása	41
6.	A szoftver tesztelése mintapéldákon keresztül	43
6.1.	Virtuális tesztvideók használata	43
6.2.	Tesztelés saját felvételekkel	45
6.2.1.	A program futtatásához kapcsolódó beállítások	47
6.2.2.	Az egytengelyű húzás elmozdulásmezeje	47
6.2.3.	Az egytengelyű húzás alakváltozási mezeje	48
7.	Összegzés, konklúzió	51
7.1.	Korlátok, továbbfejlesztési lehetőségek	51
7.2.	A nagy alakváltozások kezelésének első lépései	52
	Köszönetnyilvánítás	53
	Hivatkozások	54

1. Bevezetés

Az elmúlt évtizedekben végbemenő számítástechnikai fejlődés nagyon sok új lehetőséget nyitott a mérnöki élet összes területén. Egyre több új technológia alapul a nagyfelbontású digitális kamerák és a modern képfeldolgozó algoritmusok szinergiáján. A mechanikai anyagvizsgálatok területén is folyamatosan nő a digitális mozgáselemzéssel elvégzett tesztek szerepe. Az ilyen vizsgálatok nagy előnye, hogy érintésmentesen és roncsolásmentesen lehet ezeket kivitelezni, emellett nagy mennyiségű adatot szolgáltatnak, így pontosabb képet kaphatunk a végbemenő folyamatokról. Gondoljuk meg, hogy míg a felvételen lekövetett pontok számának szinte csak a kamera felbontása és a számítási kapacitás szab határt, addig egy hagyományosabb megközelítés esetén jól meg kell fontolnunk például, hogy a rendelkezésre álló nyúlásmérő bélyegeinket a tanulmányozott test mely pontjain helyezzük el.

Egy másik nagy előnye a módszernek, hogy bizonyos vizsgálatok távolról is elvégezhetők általa. Az Egyesült Államokban egy kutatócsoport^[1] digitális mozgáselemzésre felkészített drónnal vizsgálta vasúti talpfák szerkezeti állapotát. Az eredmények pontosnak bizonyultak a rögzített helyzetű kamerák felvételeihez képest és LVDT szenzoradatokkal való összevetésben is. Mások^[2] szintén egy drónra alapuló eljárást fejlesztettek hidak szerkezeti állapotának felmérésére. A törésmechanika területén is előszeretettel fejlesztenek digitális mozgáselemzésre alapuló, repedésterjedést és kifáradást vizsgáló rendszereket^[3,4]. Továbbá, a digitális mozgáselemzés módszere kompozit szerkezetek elemzésére is kiválóan alkalmas^[5,6].

1.1. A mozgáselemzés rövid ismertetése

A digitális mozgáselemzés módszere legtöbbször a digitális képkorrelációra épül^[7], angolul *digital image correlation* - mozaikszóval DIC. Az eljárás a referenciaképen követni kívánt pontok kijelölésével kezdődik. Ezután az algoritmus az egyes pontok körüli tet-szőleges méretű (de nem túl nagy) környezet intenzitásértékeit tekinti, amely alapján az eredeti pontokkal legjobban korreláló pontok koordinátáit adja meg a következő képkockán. Jelenleg az intenzitás kifejezés a 8 bites szürkeárnyaltos skálán felvett 0 ... 255 közötti értékre utal. Egy adott pont eredeti pozíciója és a következő képen megtalált helyzete közötti differencia adja meg az elmozdulást az idő függvényében. Az elmozdulás-mező ismeretében további kinematikai mennyiségek (például alakváltozások) különösebb nehézség nélkül számíthatók.

Visszatérve az elmozdulásra, az újabb optimalizációs technikák használatával nem csak egész számú pixel távolságot kaphatunk eredményül, hanem akár egy pixel alatti differenciát is. Az utóbbi néhány évben jelentős törekvések irányulnak a digitális képkorrelációs rendszerek neurális hálózatokkal való kiegészítésére^[8], illetve a hagyományos DIC alternatívájaként is jelennek meg különféle, mély tanuláson alapuló eljárások^[9–11].

1.2. Interneten elérhető DIC megoldások

Az alábbi sorokban néhány különböző, interneten fellelhető mozgáselemző rendszer kerül említésre. A cél, hogy az 1.2.1 táblázat segítségével az olvasó átfogó képet kapjon a piacon elérhető DIC megoldásokról, illetve az egyes módszerek jellegzetességeiről.

1.2.1. táblázat: Néhány interneten elérhető DIC lehetőség.

Név	Ismérvék
GOM Correlate ^[12]	A saját mérési rendszereikkel szimbiózisban működő 2D és 3D mozgáselemző szoftver, amelynek van ingyenes (de funkcionalitás-ban korlátozott) változata. Felhasználói felülettel rendelkezik.
X-Sight ^[13]	Teljes optikai rendszerrel együtt kínált 2D és 3D mozgáselemző rendszerek, a síkbeli változat valós idejű mérést tesz lehetővé.
pydic ^[14]	Egy Python alapú ingyenes mozgáselemző szoftver, amely képek sorozatával dolgozik és többfajta módszert ajánl az alakváltozások számítására.
μ DIC ^[15]	Szintén egy Python nyelven írt ingyenes DIC program, amely egy " <i>virtuális labor</i> " is tartalmaz, saját tesztvideók készítésére. A digitális képkorreláció számítására nemlineáris megoldót alkalmaz.
VIC-2D ^[16]	A szoftver mellé opcionálisan kamerarendszert, de akár mintázat-készítő eszköztárat is vásárolhatunk. A programot illetően 30 napos ingyenes verzió áll rendelkezésünkre.
OpenCorr ^[17]	Egy nyílt forráskódú, C++ nyelvre alapuló moduláris megoldás, amely akár 35%-os alakváltozást is kezelni tud. Sebességre optimalizált.

1.3. A dolgozat célja, tartalmi áttekintése

A munkám célja egy olyan nagyfelbontású felvételek kezelésére alkalmas síkbeli mozgáselemző szoftver létrehozása, amely ingyenesen elérhető képfeldolgozó algoritmusokra épül Python környezetben. További célom, hogy a számított mezőváltozókat vizuálisan ráillesszem a bemeneti videó vizsgálatra kijelölt területére, ezzel létrehozva a program fő eredményét, videós formában. Célkitűzéseim között szerepel a programom saját felvé-

telekkel történő tesztelése is. Nagy motivációt jelent számomra, hogy a projekt megvalósításával egy olyan megoldást nyújthatok a digitális mozgáselemzés kihívásaira, amely a Műszaki Mechanikai Tanszéken ezidáig nem volt elérhető. A jelen dolgozatban prezentált DIC szoftver hasznosnak bizonyulhat a Tanszéken folyó kutatások során, de akár hallgatótársaim munkáit is segítheti.

A következő néhány mondatban összefoglalom a dolgozat tartalmi felépítését, ezzel segítve az olvasó eligazodását. A bevezetőt követően, a második fejezetben a mozgáskövetési algoritmus alapjait ismertetem. Ugyanitt szót ejtek az elmozdulások számításának lehetséges koncepcióiról is. A harmadik fejezetben a program használatának menetét részletezem, hasonlóan egy használati útmutatóhoz. Ebben a fejezetben említésre kerülnek a `main.py` nevezetű főprogram alapvető funkciói, képességei. A negyedik fejezet a mechanikai alapokra támaszkodva ismerteti azokat az elmozdulás-adatokra épülő interpolációs technikákat, amelyek meghatározó szerepet játszanak a mezőváltozók illusztrációinak létrehozásában.

Az ötödik fejezet a program által számított eredményekről szól, így többek között kifejtésre kerül az imént említett ábrák elkészítésének módja, de a kimeneti videófelvetelek létrehozásakor felmerülő szempontokról is értekezek. Ezt követően prezentálom a program fejlesztése során igénybe vett mintapéldát, amelyet a szoftver fő funkcióinak bemutatására, elemzésére is felhasználok. Egy, a Tanszéki laborban készített videófelvétel fontosabb eredményeit ismertetem a hatodik fejezetben, emellett itt értekezek a saját fejlesztésű kódom által generált virtuális tesztvideó használatáról is, amely a mozgáskövető algoritmus paramétereinek beállításakor tehet jó szolgálatot. Az utolsó, összegző jellegű fejezetben összefoglalom a munkám során szerzett tapasztalatokat, a szoftver képességeit, emellett pedig kitekintést nyújtok a továbbfejlesztési lehetőségeket illetően.

A dolgozat függelékében található elemek részletesebben bemutatják a projekt különböző aspektusait az olvasó számára. A mellékletben többek között bemutatásra kerül a nagyjából 100 soros főprogram (`main.py`) is. Ugyanitt megtalálható továbbá a virtuális tesztvideó készítéséhez szükséges kód, valamint a mozgáskövető algoritmus pontosságának kiértékelését végző kódrészlet is, amely az imént említett tesztvideóra épít. A dolgozat csatolmányaként rendelkezésre áll néhány képernyőkép is, amelyek a szoftver használatát kívánják bemutatni, megelevenítve ezzel a 3. fejezetet.

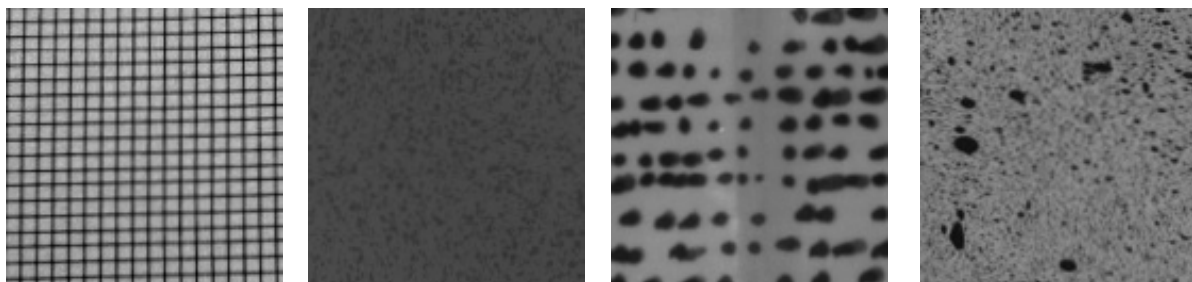
2. Mozgáskövetés

Jelen dolgozatban a mozgáskövetés módszere a digitális képkorreláción^[7] alapszik. A bevezetőben tárgyalt előnyök után tekintsük az eljárás nehézségeit, amelyek leginkább a próbatest előkészítésekor jelentkeznek.

A módszer egyik hátránya, hogy a vizsgált területnek optikailag inhomogénnek kell lennie az intenzitásértékek szempontjából. Ha ez a videófelvételen nem teljesül, akkor a digitális képkorreláció könnyen fals eredményre juthat, hiszen a következő képkockán sok, intenzitás szempontjából hasonló pont mutathat magas korrelációt egy adott referenciaponttal. Gyakorlatban ezt a problémát úgy kezelik, hogy a próbatestet egyfajta pettyes mintázattal látják el, például festékszórás segítségével. Ez hivatott biztosítani a mintadarab kellő optikai inhomogenitását. Az ideális mintázat ismerve Lovaas^[18] szerint:

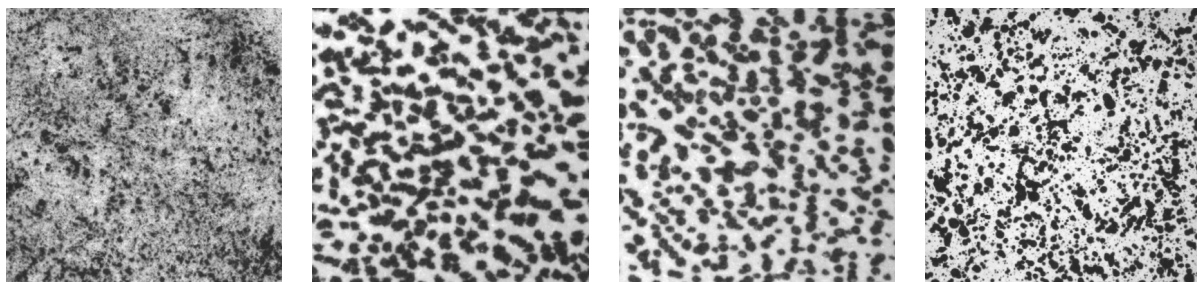
- nagy kontraszt (monokróm kép lehetőleg),
- a sötét és világos területek aránya nagyjából egyenlő,
- a pöttyök nagyjából azonos méretűek (legyen minél több, de a kicsiket kerülni kell),
- izotrop, vagyis a mintázatnak nincs kifejezett irányultsága,
- véletlenszerű: ne legyenek repetitív, ismétlődő területek.

A 2.0.1 ábra olyan mintázatokat mutat be, amelyek nem felelnek meg az imént tárgyalt követelményeknek.



(a) Repetitív. (b) Alacsony kontraszt. (c) Anizotrop. (d) Eltérő pontméretek.

2.0.1. ábra: Rossz mintázatok Lovaas^[18] szerint.



2.0.2. ábra: Jó mintázatok Lovaas^[18] szerint.

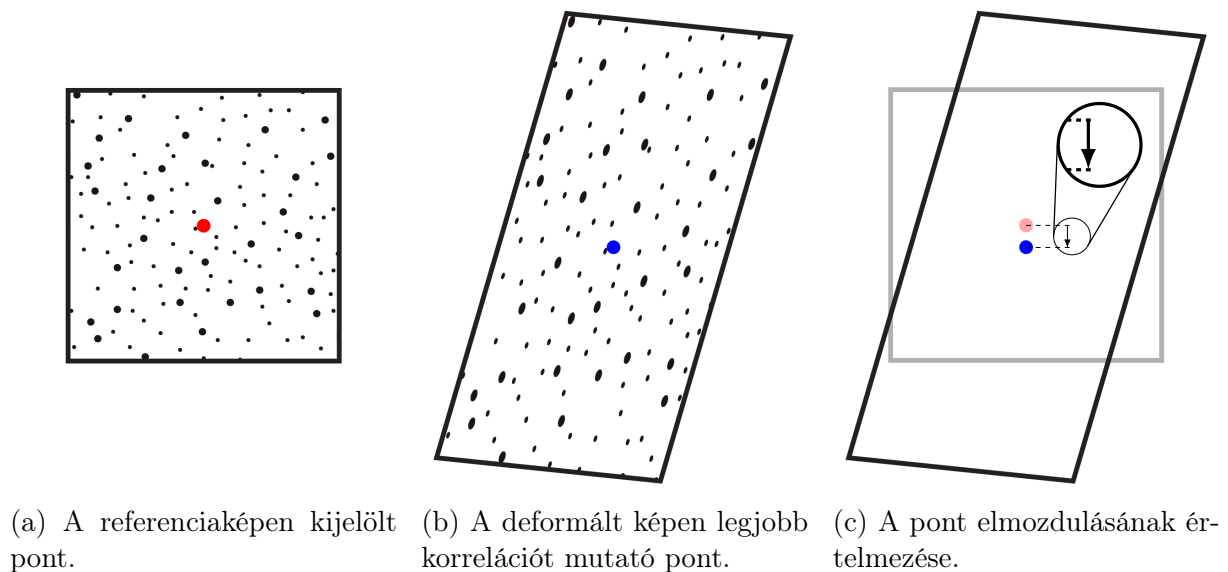
Azonban, hogy ne csak rossz példákat lásson az olvasó, a 2.0.2 ábrán illusztrált min-tázatok már megfelelően használhatók digitális képkorrelációs alkalmazásokra.

2.1. Az optical flow algoritmus működése

A munkám során egy olyan algoritmust kerestem, amely az általam kijelölt pontok mozgását képes követni. A futási idő minimalizálásának céljából egy saját képkorrelációs eljárás készítésének ötletét elvetettem. A választott algoritmus ingyenesen elérhető az OpenCV könyvtárban `calcOpticalFlowPyrLK()` ^[19] név alatt. Az optical flow módszer képi objektumok követésére ad lehetőséget egymást követő felvételek sorozatán. Az eljárás kettő fontos feltételezést tesz:

- a követett objektum pixeleinek intenzitása nem változik számottevően kettő kép között,
- a szomszédos pixelek hasonló pályát követnek.

Színes bemeneti felvételek esetén az optical flow függvény meghívása előtt az összehasonlítandó képeket szürkeárnyalatossá szükséges konvertálni.



2.1.1. ábra: A digitális képkorreláció működése egy ponttal szemléltetve ^[20].

Vegyük tehát egy tetszőleges pont intenzitását. A *pont* kifejezés jelen esetben nem egy diszkrét pixelt jelöl, sokkal inkább a fizikai valóságban létező pontra, entitásra érdemes gondolni jelen szöveggörnyezetben. Egy ilyen *pontot* láthatunk a 2.1.1a ábrán, piros színnel jelölve. A 2.1.1 ábra eredeti verzója LePage ^[20]-től származik, ám a jobb szemléltetés érdekében módosítottam rajta. Az említett illusztráció összessége a képkorrelációs

mozgáselemzés fő elvét kívánja bemutatni, ebből adódóan az ideális pettyes mintázat reprezentálása itt nem szempont. Emellett a könnyebb megértést szolgálja az egymást követő képkockákon megjelenő túlzott mértékű deformáció.

A keresett elmozdulás felírásához induljunk ki a korábbi feltételezéseinkből. Egy kicsi elmozdulást (dx, dy) követően a vizsgált *pont* intenzitásértéke nem változik a következő képen (dt idővel később), vagyis:

$$I(x, y, t) = I(x + dx, y + dy, t + dt). \quad (2.1.1)$$

Tekintsük az egyenlet jobb oldalának elsőrendű közelítését az (x, y, t) helyen.

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + dx \frac{\partial I}{\partial x} + dy \frac{\partial I}{\partial y} + dt \frac{\partial I}{\partial t}. \quad (2.1.2)$$

A visszahelyettesítést és az egyszerűsítést elvégezve, illetve az egyenletet dt -vel elosztva a 2.1.3 egyenletben szereplő alakra jutunk. Az így kapott összefüggést az optical flow differenciálegyenletének nevezzük.

$$0 = \frac{\partial I}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial I}{\partial y} \cdot \frac{dy}{dt} + \frac{\partial I}{\partial t}. \quad (2.1.3)$$

Az intenzitásfüggvény parciális deriváltjai képi gradiensszámítással megadhatók, az egyenlet ismeretlenei a dx/dt és dy/dt sebességkomponensek. Kettő ismeretlen lévén az egyenletet nem tudjuk jelen formájában megoldani. E probléma megoldására született többféle megközelítés is.

2.2. A Lucas-Kanade módszer

Az általam választott `calcOpticalFlowPyrLK()` módszer nevében szereplő LK betűk a Lucas-Kanade eljárásra utalnak^[21], amely a legkisebb négyzetek módszerét alkalmazza az optical flow egyenletének megoldására. Ehhez felhasználja azt a feltételezést, miszerint a szomszédos objektumok elmozdulása hasonló. Az optical flow differenciálegyenletét most egyszerűbb jelöléssel ellátva tekintjük. Legyen

$$\frac{\partial I}{\partial \alpha} = I_\alpha \quad \text{és} \quad \frac{d\alpha}{dt} = v_\alpha.$$

Az eljárás $i = 1 \dots n$ darab tetszőlegesen kiválasztott pontot vesz alapul a kettő

sebességkomponens meghatározására, amely alkalmassá teszi DIC módszerek implementálására.

$$0 = I_{x,i} \cdot v_x + I_{y,i} \cdot v_y + I_{t,i}. \quad (2.2.1)$$

Feltesszük, hogy a 2.2.1 összefüggés az összes kiválasztott pontra ($i = 1 \dots n$) igaz. A legkisebb négyzetek módszere szerint minimalizálandó az alábbi kifejezés.

$$Q = \sum_{i=1}^n (I_{x,i} \cdot v_x + I_{y,i} \cdot v_y + I_{t,i})^2. \quad (2.2.2)$$

Lokális minimum ott lehet, ahol a fenti kifejezés sebességkomponensek szerinti parciális deriváltjai zérust adnak.

$$\frac{\partial Q}{\partial v_x} = 2 \sum_{i=1}^n (I_{x,i} \cdot v_x + I_{y,i} \cdot v_y + I_{t,i}) I_{x,i} = 0. \quad (2.2.3)$$

$$\frac{\partial Q}{\partial v_y} = 2 \sum_{i=1}^n (I_{x,i} \cdot v_x + I_{y,i} \cdot v_y + I_{t,i}) I_{y,i} = 0. \quad (2.2.4)$$

Egyszerűbben:

$$v_x \sum_{i=1}^n I_{x,i}^2 + v_y \sum_{i=1}^n (I_{x,i} \cdot I_{y,i}) + \sum_{i=1}^n (I_{x,i} \cdot I_{t,i}) = 0, \quad (2.2.5)$$

$$v_x \sum_{i=1}^n (I_{x,i} \cdot I_{y,i}) + v_y \sum_{i=1}^n I_{y,i}^2 + \sum_{i=1}^n (I_{y,i} \cdot I_{t,i}) = 0. \quad (2.2.6)$$

A 2.2.5 és 2.2.6 kifejezések v_x -re és v_y -ra nézve lineárisak, így mátrixos alakban is összefoglalhatjuk őket.

$$\begin{bmatrix} \sum_{i=1}^n I_{x,i}^2 & \sum_{i=1}^n (I_{x,i} \cdot I_{y,i}) \\ \sum_{i=1}^n (I_{x,i} \cdot I_{y,i}) & \sum_{i=1}^n I_{y,i}^2 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} - \sum_{i=1}^n (I_{x,i} \cdot I_{t,i}) \\ - \sum_{i=1}^n (I_{y,i} \cdot I_{t,i}) \end{bmatrix}. \quad (2.2.7)$$

Vagyis a keresett ismeretlenek:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n I_{x,i}^2 & \sum_{i=1}^n (I_{x,i} \cdot I_{y,i}) \\ \sum_{i=1}^n (I_{x,i} \cdot I_{y,i}) & \sum_{i=1}^n I_{y,i}^2 \end{bmatrix}^{-1} \begin{bmatrix} - \sum_{i=1}^n (I_{x,i} \cdot I_{t,i}) \\ - \sum_{i=1}^n (I_{y,i} \cdot I_{t,i}) \end{bmatrix}, \quad (2.2.8)$$

$$\mathbf{v} = \mathbf{A}^{-1} \mathbf{b},$$

ahol $\mathbf{v}$ az úgynevezett *flow vector*.

Természetesen a módszer működéséhez elengedhetetlen, hogy az $\mathbf{A}$ *térbeli gradiens-mátrix* invertálható legyen. Az eljárás akkor adja a legpontosabb eredményeket, hogyha az említett mátrix sajátértékeire igaz az, hogy $\lambda_1 \approx \lambda_2 \gg 0$, vagyis mindkettő sajátérték pozitív, nagyjából megegyeznek, de nem "túl kicsik". Szimmetrikus mátrix lévén $\lambda_1, \lambda_2 \in \mathbb{R}$. Megjegyzendő, hogy ezek a követelmények sarokpontok esetén teljesülnek leginkább. Olyan alkalmazási területen ez a feltétel fontos lehet, ahol nem közvetlenül a felhasználó jelöli ki a követni kívánt pontok halmazát. Ekkor lehetőségünk van a `goodFeaturesToTrack()` függvény használatára, amely igyekszik a fentebbi kritériumoknak leginkább megfelelő (sarok)pontokat kiválasztani a Shi-Tomasi módszer szerint^[22].

Fontos még kiemelni, hogy az optical flow differenciálegyenletét linearizálás segítségével írtuk fel, ebből adódóan csak kis elmozdulásokra, azaz kicsi flow vector értékekre érvényes. Ez azonban nem okoz problémát, ha a vizsgált jelenség nem túl gyorsan zajlik le. Ha mégis ez a helyzet, akkor kellően nagy FPS¹ értékkel rendelkező kamerát kell választanunk a feladatra. A végső elmozdulásértékeket iteráció segítségével kapjuk meg, amely során az egyik leállási kritérium az lehet, hogy a $\mathbf{v}$ vektor komponensei egy bizonyos nullaközeli ε küszöb alatti értéket vesznek fel.

2.3. A Lucas-Kanade algoritmus paraméterei

Az imént ismertetett eljárás paramétereinek tárgyalása előtt fontos néhány szót szólni az úgynevezett piramisszintekről^[23]. Korábban említettem, hogy a Lucas-Kanade módszer akkor működik jól, ha a referencia és a deformált kép között kicsi, maximum néhány pixeles elmozdulások történnek. Az algoritmus ezen hátrányának kiküszöbölésére fejlesztették ki a piramisszinteket. Az elnevezés arra utal, hogy mind a referencián, mind a vizsgált képen szándékos felbontáscsökkentést hajtanak végre, Gauss elmosással együtt, akár több lépcsőben. Az eredeti képek a legalsó piramisszinthez tartoznak, és ahogy egyre fentebbi szintekre lépünk, úgy lesznek egyre alacsonyabb felbontásúak, elmosottabbak a képek.

A nagyobb elmozdulások oly módon válnak kezelhetővé ezen módszer használatával, hogy kezdetben a legfelső szinten, a legkisebb felbontású képpáron fut végig az algoritmus, és keresi meg a vizsgált képen a követett pont helyét. Ekkor természetesen nagyobb magabiztossággal találja meg a Lucas-Kanade módszer a keresett pont helyét a következő képen, hiszen a felbontás kicsi, vagyis az elmozdulás pixelben tekintve szintén kicsi. Ezt követően a lentebbi piramisszintre lépve, az imént számolt új pozíció kezdeti becslésként

¹FPS: frames per second, képkocka/másodperc

működik, ahonnan nézve a felbontásnövekedésből fakadó elmozdulás szintén kicsi lesz. Az iteráció egészen a legalsó szintig folytatódik, ezzel lehetőleg minél jobban kiküszöbölve a Lucas-Kanade eljárás hátulütőjét. A technikát *coarse-to-fine estimation* néven azonosítják az irodalomban^[24]. Immár a 2.3.1 táblázatban olvasható különféle paraméterek jelentése könnyen érthető.

2.3.1. táblázat: A Lucas-Kanade módszer paraméterei^[25].

Név	Leírás
winSize	Az egyes pontok keresési ablakának mérete pixelben, minden piramisszinten.
maxLevel	A piramisszintek maximális száma (ha 0, nincsenek piramisszintek).
criteria.epsilon	Az iteráció leáll a keresési ablak ε -nál kisebb elmozdulása esetén.
criteria.maxCount	Az iteráció leáll a beállított számú lépés után.
minEigThreshold	Ha a pontokhoz tartozó $\mathbf{A}$ mátrix kisebbik sajátértéke a beállított küszöb alatti, akkor a pontthalmazzal nem számolunk tovább.

2.4. A rácspontok elmozdulásának számítása

Az általam írt programban téglalap alakú területet lehet mozgáselemzés alá vetni. Ez azt jelenti, hogy a kijelölt területen a felhasználó által beállított vízszintes, illetve függőleges irányú osztásközökkel létrejön egy pontrács, amelynek pontjai a Lucas-Kanade algoritmus bemeneteként szolgálnak. Az optical flow paraméterek beállítása, illetve a követni kívánt rácspontok létrehozása után az elmozdulások számítása már nem okoz nagy kihívást.

```

1  for i in range(len(frames)-1):
2      image_ref = cv2.cvtColor(frames[i], cv2.COLOR_BGR2GRAY)
3      image_next = cv2.cvtColor(frames[i+1], cv2.COLOR_BGR2GRAY)
4      next_points = cv2.calcOpticalFlowPyrLK(image_ref, image_next,
5          points_to_process, **lk_params)
        points_to_process = next_points

```

2.4.1. ábra: A mozgáskövetés implementációja Lucas-Kanade módszerrel.

A 2.4.1 ábrán látható kódrészletben az olvasható, hogy a `calcOpticalFlowPyrLK()` eljárásnak a két összehasonlítandó képre, illetve a keresett pontok referenciaképi koordi-

nátáira van szüksége a 2.3.1 táblázat paraméterein kívül. Mindezek segítségével megadja a következő képkockán a keresett pontok koordinátáit. Ezeket a kód további részében egy fájlba mentem ki, hiszen szükség van rájuk a további számításokhoz. A ciklus következő lépésében az aktuálisan megtalált pontok válnak keresett pontokká.

Ily módon a ciklus lefutása után minden vizsgált képkockán ismerjük a követett rácspontok koordinátáit. Ekkor az elmozdulások számítása úgy történik, hogy minden képkockát a legelsőhöz viszonyítunk és szimplán képezzük az egymásnak megfelelő koordináták különbségét. Így például az első képi elmozdulásmező azonosan nulla. Fontos megemlíteni, hogy az elmozdulás-komponensek y koordinátáját célszerűen megszorozom (-1) -gyel, hiszen a képelemzés világában az y tengely pozitív iránya általában lefelé mutat, ahogy jelen esetben is.

```

1  def compute_displacement(point_ref: list, point_next: list):
2      assert len(point_ref)==len(point_next)
3      disp = []
4      for i, pt0 in enumerate(point_ref):
5          pt1 = point_next[i]
6          disp.append(np.array([pt1[0]-pt0[0], -1*(pt1[1]-pt0[1])]))
7      return disp

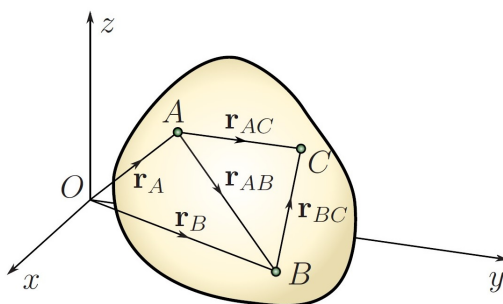
```

2.4.2. ábra: Elmozdulások számítása egy adott képkockán.

A 2.4.2 ábrán olvasható függvényt minden képkockára iteratívan meghívva számolom az egyes rácspontok elmozdulását. Az így nyert elmozdulásértékeket szintén kimentem egy fájlba a folyamat végén.

2.5. Merev testszerű mozgások kiszűrésének lehetősége

Sok felvételen a vizsgálat jellegéből adódóan, esetleg a kamera nem kívánt mozgása miatt könnyen előfordulhatnak merev testszerű elmozdulások és rotációk két képkocka között.



2.5.1. ábra: Merev testek bármely két pontja közötti távolság állandó^[26].

A klasszikus mechanikában a merev test modellje azzal a feltételezéssel él, hogy a mozgás során a test bármely két pontjának távolsága időben állandó. Tehát a 2.5.1 ábrán például az A és B pontot tekintve $|\mathbf{r}_{AB}| = \text{állandó}$. Ennek megfelelően a merev testszerű mozgás definíció szerint azt jelenti, hogy a test minden pontjában azonos elmozdulásvektor hat. Ezek kiszűrése nem létszükséglet az alakváltozási mező számítása – vagyis a numerikus deriválás – szempontjából, hiszen az ilyen jellegű elmozdulásmező térkoordináták szerinti parciális deriváltjai zérusak. Ez azt jelenti, hogy a merev testszerű elmozdulások és rotációk nem okoznak alakváltozást.

Ennek ellenére a deformációk jellegéről pontosabb képet kaphat a felhasználó a merev testszerű mozgások kiszűrésével, amelynek lehetőségét implementáltam a szoftverben. Keressük tehát azt az $\mathbf{R} \in \mathbb{R}^{2 \times 2}$ rotációs mátrixot és $\mathbf{t} \in \mathbb{R}^{2 \times 1}$ translációs vektort, amely a legkisebb négyzetek módszerének felhasználásával a legpontosabban áttanszformálja a referenciaképi $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n\}$ ponthalmaz elemeit a következő képkockához tartozó $\mathcal{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n\}$ ponthalmaz megfelelő pontjaiba, ahol $\mathbf{p}_i, \mathbf{q}_i \in \mathbb{R}^{2 \times 1}$, $i = 1, 2, \dots, n$.

A számítás Sorkine-Hornung and Rabinovich^[27] cikkében leírtak alapján történik, ahol a módszer általánosítása és bizonyítása is olvasható. Tehát minimalizálandó a

$$\sum_{i=1}^n \|(\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i)\|^2 \quad (2.5.1)$$

kifejezés. Első lépésként kiszámoljuk mindkét ponthalmaz súlypontját, azaz:

$$\bar{\mathbf{p}} = \frac{\sum_{i=1}^n \mathbf{p}_i}{n}, \quad \bar{\mathbf{q}} = \frac{\sum_{i=1}^n \mathbf{q}_i}{n}.$$

Ezután mindkét ponthalmaz esetében képezzük az egyes pontok súlyponttól vett távolságát:

$$\mathbf{x}_i := \mathbf{p}_i - \bar{\mathbf{p}}, \quad \mathbf{y}_i := \mathbf{q}_i - \bar{\mathbf{q}}.$$

Az így számolt különbségeket elhelyezzük az $\mathbf{X}$ és $\mathbf{Y}$ ($\in \mathbb{R}^{2 \times n}$) mátrixok egymást követő oszlopaiban. Azaz legyen

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_n \end{bmatrix} \quad \text{és} \quad \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \dots & \mathbf{y}_n \end{bmatrix}.$$

Innen már összeállítható a 2×2 -es kovariancia mátrix az alábbi módon:

$$\mathbf{S} = \mathbf{X}\mathbf{Y}^\top. \quad (2.5.2)$$

Ezt a mátrixot kell a következő lépésben szinguláris értékek szerint felbontani, vagyis előállítani úgy, hogy:

$$\mathbf{S} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \quad (2.5.3)$$

ahol az egyenlőség jobb oldalán szereplő mátrixok mind 2×2 -esek jelen esetben, hiszen kétdimenziós síkbeli problémával dolgozunk. A $\mathbf{\Sigma}$ mátrix főátlójában szerepelnek a szinguláris értékek, vagyis az $\mathbf{S}\mathbf{S}^T$ mátrix sajátértékeinek négyzetgyökei. Ezenkívül $\mathbf{U}$ és $\mathbf{V}$ ortogonális mátrixok, oszlopaikat rendre bal és jobb oldali szinguláris vektoroknak nevezzük. A felbontás számításának módszerét jelen dokumentumban nem fejtem ki, azonban a matematikai háttér Rózsa^[28] könyvében olvasható. A szinguláris felbontást nagyon sok területen alkalmazzák, például gépi tanulás során nagy adathalmazok dimenziócsökkentésére^[29], képfeldolgozás esetén tömörítési eljárásokban^[30], de rosszul kondicionált lineáris rendszerek megoldására is alkalmas^[31].

A felbontás elvégzése után a keresett rotációs mátrix a következő alakban írható fel:

$$\mathbf{R} = \mathbf{V} \begin{bmatrix} 1 & 0 \\ 0 & \det(\mathbf{V}\mathbf{U}^T) \end{bmatrix} \mathbf{U}^T, \quad (2.5.4)$$

ezzel a keresett transláció pedig:

$$\mathbf{t} = \bar{\mathbf{q}} - \mathbf{R}\bar{\mathbf{p}}. \quad (2.5.5)$$

A célunk azonban az, hogy a következő képkockához tartozó $\mathbf{q}_i$ pontok koordinátáiból a merev testszerű mozgást kiszűrjük az imént kiszámított mennyiségek segítségével. Ezt a legegyszerűbben úgy tudjuk megtenni, hogy az $\mathbf{X}$ és $\mathbf{Y}$ mátrixok mintájára elkészítjük a $\mathcal{P}$ és $\mathcal{Q}$ halmaz pontjaiból a

$$\mathbf{P} = \left[\mathbf{p}_1 \mid \mathbf{p}_2 \mid \dots \mid \mathbf{p}_n \right] \quad \text{és} \quad \mathbf{Q} = \left[\mathbf{q}_1 \mid \mathbf{q}_2 \mid \dots \mid \mathbf{q}_n \right]$$

mátrixokat, melyek dimenzója $2 \times n$. Emellett a $\mathbf{t}$ translációt minden pontra alkalmaznunk kell, így célszerűen létrehozuk a szintén $2 \times n$ -es

$$\mathbf{T} = \left[\mathbf{t} \mid \mathbf{t} \mid \dots \mid \mathbf{t} \right]$$

mátrixot. Ha a $\mathbf{P}$ mátrixra alkalmazzuk $\mathbf{R}$ és $\mathbf{T}$ transzformáló hatását, akkor megkapjuk azt a $\mathcal{Q}^*$ ponthalmazt, amely az eredeti $\mathcal{P}$ -hez képest kizárólag a merev testszerű mozgás

hatását tartalmazza. Ezen pontok mátrixa legyen a $\mathbf{Q}^*$.

$$\mathbf{Q}^* = \mathbf{R}\mathbf{P} + \mathbf{T}. \quad (2.5.6)$$

Ezt a hatást le kell vonnunk az optical flow algoritmus által meghatározott $\mathcal{Q}$ pont-halmaz koordinátaiból, hogy megkapjuk a merev testszerű mozgásoktól mentesített elmozdulásértékeket. Azaz

$$\mathbf{D} = \mathbf{Q} - \mathbf{Q}^*, \quad (2.5.7)$$

ahol a $\mathbf{D} \in \mathbb{R}^{2 \times n}$ mátrix tartalmazza az egyes pontok korrigált elmozdulását. Természetesen az elmozdulások y komponensét, vagyis a $\mathbf{D}$ mátrix második sorát ismét megszorozom (-1) -gyel, ellensúlyozva ezzel a lefelé irányított y tengelyt. Legutolsó lépésként a kiszámolt elmozdulásértékeket ebben az esetben is kimentem egy ellenőrző fájlba.

3. A szoftver használata

Ebben a fejezetben áttekintést kap az olvasó arról, hogy az általam készített program mire alkalmas és hogyan használható. Az itt körülírt munkamenet a dolgozat függelékében található képernyőképek sorozatán keresztül kerül szemléltetésre.

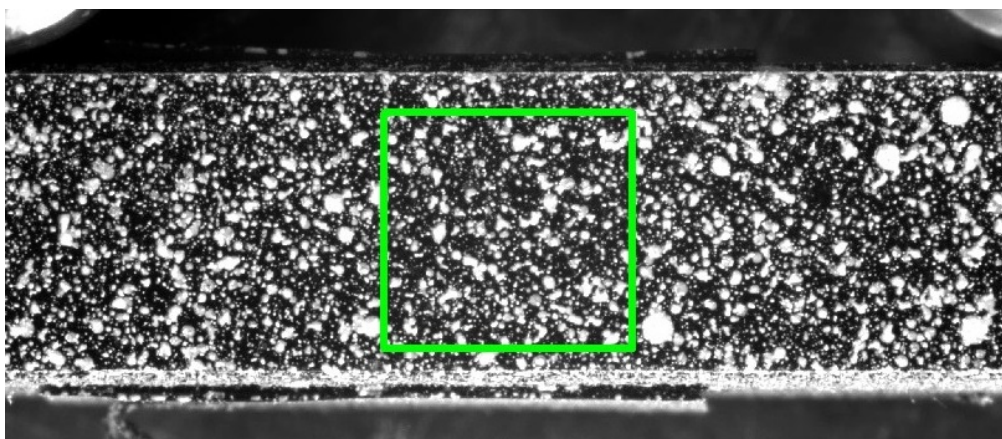
Fontos kiemelni, hogy a vizsgálat tárgyát képező videófelvételt különálló képek sorozatára bontja a szoftver. Az így kapott képeken fut majd a digitális mozgáselemzés. Éppen ezért célszerű legelső lépésként a bemeneti videó elejéről és végéről minden olyan részletet eltávolítani, amely nem szigorúan a mechanikai vizsgálathoz tartozik. A munkamenet a mellékletben szereplő `main.py` főprogram módosításával folytatódik az alábbi lépések szerint.

1. Az `.mp4` formátumú videó elérési útvonalának megadása. A számítások során lehetőség van csupán minden n -edik képkocka felhasználására.
2. Az optical flow algoritmus 2.3.1 táblázatban olvasható paramétereinek beállítása.
3. A pontrács vízszintes és függőleges irányú osztásközének megadása pixelben.
4. A kimeneti illusztrációkon szereplő színskála testreszabása.
5. Az elmozdulás- és az alakváltozási mező számításához tartozó funkciók kiválasztása.
 - Merev testszerű mozgások kiszűrése: Igen/Nem.
 - Csomóponti alakváltozások átlagolása: Igen/Nem.
6. Feszültségek számítása síkfeszültségi állapotra: Igen/Nem.
7. A feszültségmezőhöz tartozó funkciók beállítása, anyagparaméterek megadása.
 - E rugalmassági modulusz megadása, [Pa] vagy [MPa] mértékegységben.
 - ν [–] Poisson tényező megadása.
 - Csomóponti feszültségek átlagolása: Igen/Nem.
 - von Mises egyenértékű feszültség számítása: Igen/Nem.
8. Vizualizációs függvények meghívása azon mezőváltozókra, amelyek eredményeit a felhasználó videós formában kívánja megtekinteni.

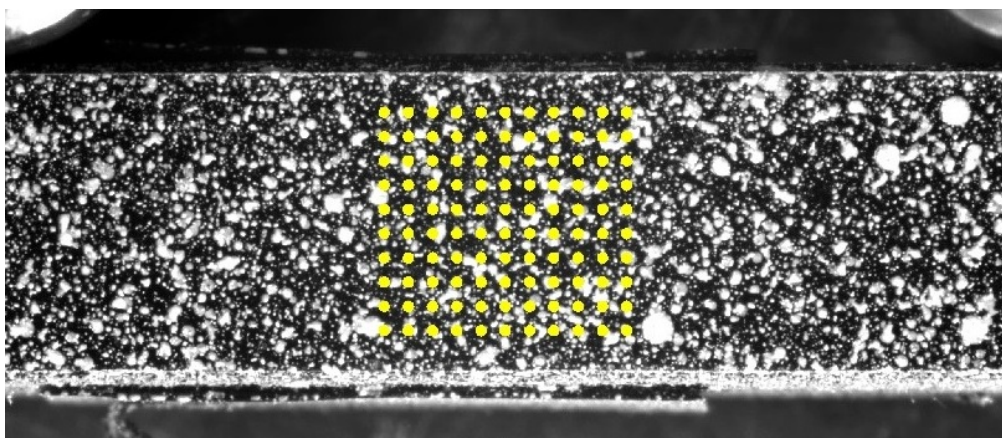
A videók beolvasását, kezelését a nyílt forráskódú OpenCV könyvtárból származó függvények végzik^[32]. Magas FPS-sel készült felvételek esetén hasznos lehet az az általam implementált funkció, amellyel csak minden n -edik képkocka kerül mentésre. Ez azért előnyös, mert az ilyen felvételeken sokszor nagyon kicsi változások történnek képről képre, amely sok felesleges számítást, ezáltal megnövelt futásidőt okozhat.

A főprogram saját bemenetre való adaptálása után a szoftver futtatása következik. A program elsőként az elmentett képkockákból összeállít egy videót, amit a felhasználó elér egy videólejátszó ablakban. Ezzel lehetőségünk van ellenőrizni a beolvasás helyességét, illetve azt is, hogy megfelelően választottuk-e meg az n értékét: nincsenek-e szemmel láthatóan túl nagy, avagy túl kicsi elmozdulások a képkockák között. A program elindítását követően végrehajtandó lépéseket az alábbi pontok foglalják össze.

1. Az 'esc' billentyű lenyomása a kimentett képkockák szekvenciájának elfogadására.
2. A következő felugró ablakban a vizsgálni kívánt terület kijelölése a bal egérgomb nyomva tartásával. A kijelölést többször újra lehet próbálni. A vizsgálandó terület kijelölését illusztrálja a 3.0.1 ábra.
3. A létrehozott pontrács ellenőrzése, a beállított osztásközök esetleges felülbírálnása. Ezen stádiumot mutatja be a 3.0.2 ábra.
4. A 'c' gomb lenyomása a számítások indítására.
5. A részeredmények és a program állapotának nyomon követése a konzolablakon.



3.0.1. ábra: Terület-kijelölés.



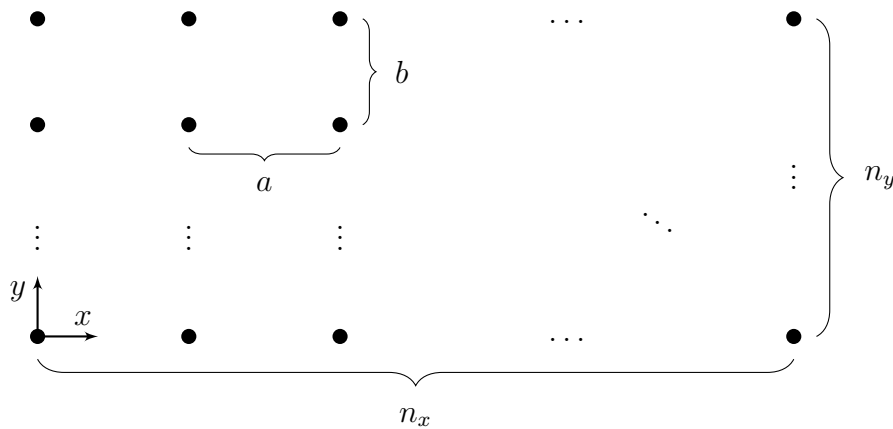
3.0.2. ábra: A végleges pontrács.

4. A mezőváltozók számítása, interpolációja

A rácspontok elmozdulását a 2. fejezetben tárgyalt módszerek egyikével immár kiszámoltuk. Ezt követően létrehozuk a végelem módszerben ismeretes diszkretizációt, ideértve annak adatstruktúráját is, mielőtt bármilyen számításba, interpolációba kezdünk.

4.1. Hálózás

A felhasználó által vizsgálni kívánt területre az előzetesen beállított osztásközökkel a program már elhelyezte a 4.1.1 ábrán látható pontrácsot, amely $n_x \cdot n_y$ számú pontból áll. A mezőváltozók interpolációjára a téglalap elrendezésből logikusan adódó négycsomópontos lineáris négyszögelemet választottam. Előzetes tanulmányaim alapján úgy gondolom, hogy ez a végelem megfelelő kompromisszumot biztosít a pontosság és a számítás bonyolultsága (futásideje) között.



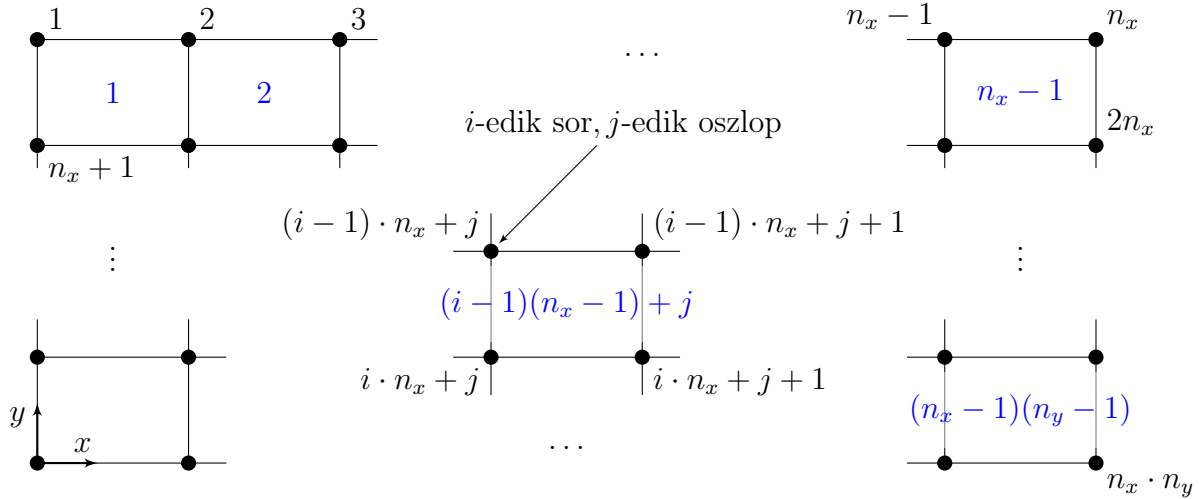
4.1.1. ábra: A követett pontok rácsa.

A 4.1.1 ábrán látszik, hogy az x irányú osztásköz értéke a pixel, y irányban pedig b pixel. Ez azt jelenti, hogy az összes végelem egy a és b oldalhosszúságokkal rendelkező, egybevágó téglalap. Megjegyzendő, hogy a téglalapokat az egyik átlójuk mentén kettéosztva könnyen folytathatnánk a munkát lineáris háromszögelemekkel is.

Az y tengely pozitív irányát immár felfelé mutatónak vesszük, hiszen az elmozdulások számításánál már korrigáltuk az itt felmerülő problémát. A végelemes diszkretizáció fontos lépéseként a rácspontokból csomópontok, a téglalapokból pedig elemek lesznek. A Python programnyelv lehetőséget ad arra, hogy mindkettő fogalmat osztályként, objektum-orientált módon kezeljük. A dolgozat mellékletében olvasható a `Node` és az `Element` osztály konstruktora, ahol látható, hogy mindkét objektumtípushoz a mechanikai mennyiségeken kívül hozzárendelünk globális sorszámot, illetve a megfelelő képkocka

sorszámát is. Továbbá, az `Element` osztály a `Node` osztály objektumaira építkezik a lokális csomópontok hozzárendelése során, vagyis osztálykompozíció történik.

A csomópontok globális sorszámozását illetően $i = 1 \dots n_y$ sorról beszélhetünk, oszlopokat tekintve pedig $j = 1 \dots n_x$. Az elemek globális sorszámát kékkkel jelöltem a 4.1.2 ábrán. Fontos megemlíteni, hogy mivel egy sorban $n_x - 1$ számú elem van, ezért az i -edik sor és j -edik oszlop csomópontjához tartozó elem sorszáma $(i - 1)(n_x - 1) + j$, hiszen az elemekre nézve $i = 1 \dots (n_y - 1)$ és $j = 1 \dots (n_x - 1)$.



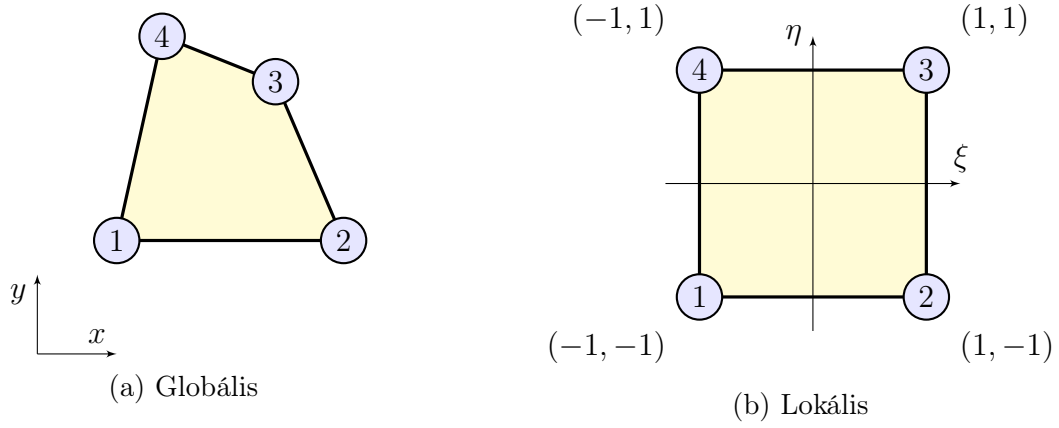
4.1.2. ábra: Globális számozások.

4.2. Az elmozdulásmező létrehozása

A globális elmozdulásmező képről-képre oly módon áll össze, hogy az egyes elemeken belüli interpoláció elvégzése után a kapott részeredményeket összeállítjuk a globális tartományra vonatkozóan. Az imént említett összeállításról az 5. fejezetben bővebben értekeznek. Először tekintsük az elemekre vonatkozó elmozdulásmező számításának módját.

4.2.1. Az elemenkénti elmozdulásmező interpolációja

Vegyünk egy, a globális $x - y$ koordináta-rendszerben tetszőleges helyen lévő általános négyszögelemet. A diszkrét csomóponti értékek interpolációjának alapja, hogy minden elemet leképezünk egy lokális $\xi - \eta$ koordináta-rendszerbe, amelyben a tetszőleges geometriájú négyszögelem egy szabályos négyzet alakot vesz fel, vagyis $\xi, \eta \in [-1, 1]$.



4.2.1. ábra: Koordináta-rendszerek.

A 4.2.1 ábrán az adott elem csúcsainál szereplő számok az elem lokális csomópontjait jelölik, melyek sorszámozása konvenció szerint jobbkéz-szabály szerint történik, azonban az ①-es csomópont megválasztása tetszőleges. A kettő említett koordináta-rendszer közötti leképezést az alábbi képletek írják le:

$$x = \sum_{i=1}^4 N_i \cdot x_i, \quad \text{és} \quad y = \sum_{i=1}^4 N_i \cdot y_i, \quad (4.2.1)$$

ahol x_i és y_i az adott elem csomóponti koordinátái a globális koordináta-rendszerben, N_i pedig a formafüggvényeket (interpolációs függvényeket) jelöli ($i = 1 \dots 4$), amelyek minden felhasznált elem esetén azonosak a $\xi - \eta$ koordináta-rendszerben:

$$N_1 = \frac{1}{4}(1 - \xi)(1 - \eta), \quad (4.2.2)$$

$$N_2 = \frac{1}{4}(1 + \xi)(1 - \eta), \quad (4.2.3)$$

$$N_3 = \frac{1}{4}(1 + \xi)(1 + \eta), \quad (4.2.4)$$

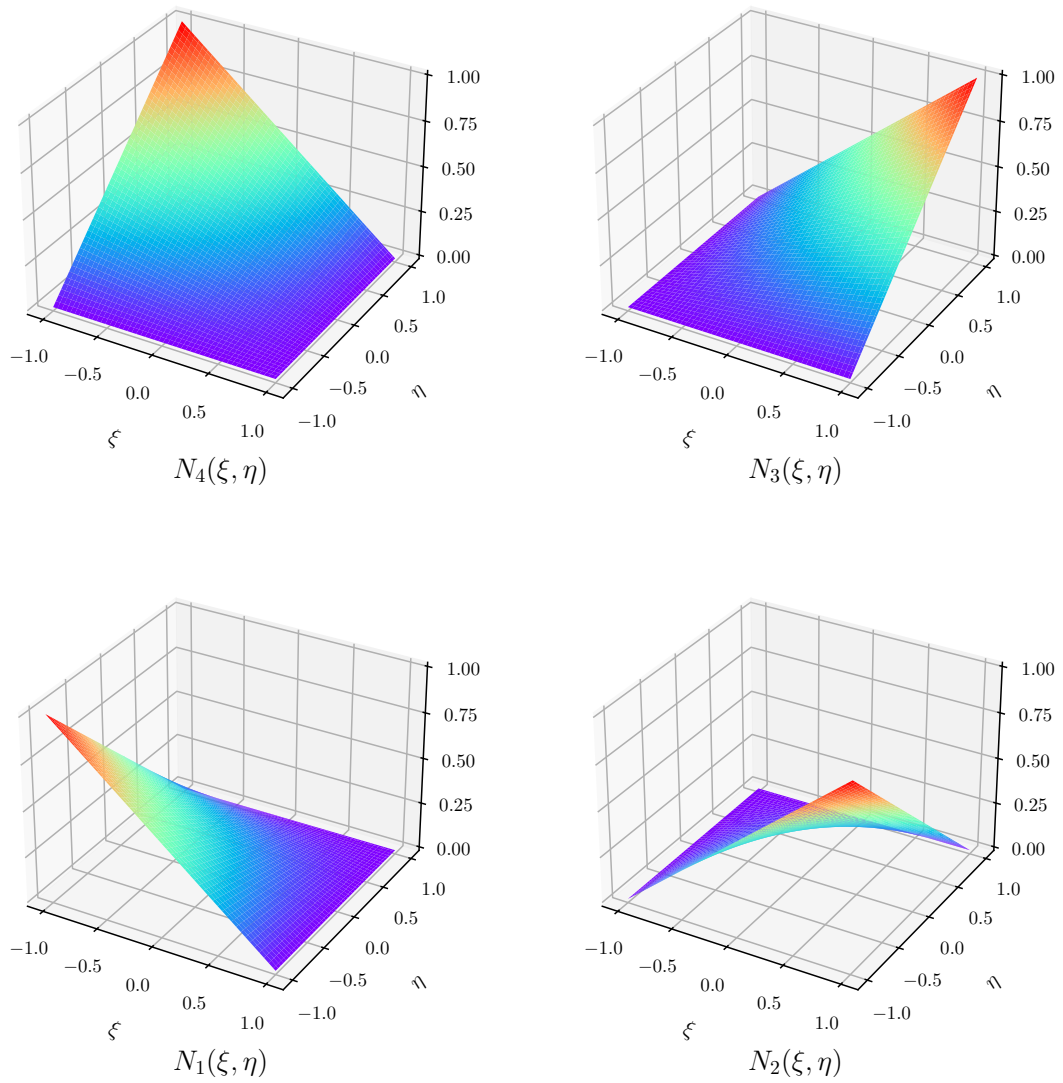
$$N_4 = \frac{1}{4}(1 - \xi)(1 + \eta). \quad (4.2.5)$$

Ezek a bilineáris függvények olyan tulajdonsággal rendelkeznek, hogy az N_i jelölésű kifejezés az i -edik lokális csomópont ξ_i és η_i koordinátáját behelyettesítve éppen 1-et ad eredményül, viszont a maradék három függvény ugyanezen a helyen nullát ad vissza, azaz:

$$N_i(\xi_i, \eta_i) = 1, \quad \text{de: } N_i(\xi_j, \eta_j) = 0, \quad \text{ahol: } j \neq i. \quad (4.2.6)$$

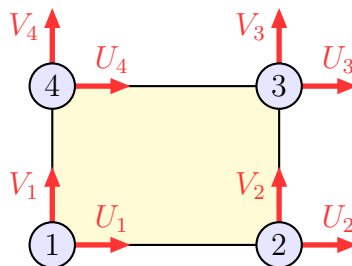
A 4.2.2 ábrán a formafüggvények grafikonjai láthatók a lokális csomópontoknak megfelelő

elrendezésben.



4.2.2. ábra: Formafüggvények.

Folytatva az elemen belüli elmozdulásmező felírását, a képkorrelációs algoritmus segítségével meghatározott elmozdulás-komponenseket jelölje $U_1 \dots U_4$ a vízszintes irány vonatkozásában, illetve $V_1 \dots V_4$ a függőleges irányban.



4.2.3. ábra: Elmozdulás-komponensek.

A 4.2.3 ábrán az egyes elmozdulás-komponensek indexelése az adott lokális csomópontra utal. Az interpolációra térve, a dolgozatomban izoparametrikus elemdefiníciót használok, amely azt jelenti, hogy a mezőváltozókat ugyanazokkal az interpolációs függvényekkel közelítem, mint a geometriát. Vagyis a vízszintes és függőleges irányú elmozdulások bármely tetszőleges (ξ, η) párra a lokális koordináta-rendszerben az alábbi módon írhatók fel:

$$u(\xi, \eta) = \sum_{i=1}^4 N_i \cdot U_i, \quad \text{és} \quad v(\xi, \eta) = \sum_{i=1}^4 N_i \cdot V_i. \quad (4.2.7)$$

Az elmozdulás-komponenseket a 8×1 -es ún. csomóponti elmozdulásvektorba rendezhetjük az alábbi módon:

$$\mathbf{U}^T = \begin{bmatrix} U_1 & V_1 & U_2 & V_2 & U_3 & V_3 & U_4 & V_4 \end{bmatrix}. \quad (4.2.8)$$

A jobb helykihasználás érdekében a kérdéses vektor transzponáltját írtam fel. E mennyiség felhasználásával az egy elemre vonatkozó elmozdulásmező tömörebb, mátrixos formában is felírható:

$$\mathbf{u}(\xi, \eta) = \begin{bmatrix} u(\xi, \eta) \\ v(\xi, \eta) \end{bmatrix} = \mathbf{N}(\xi, \eta) \cdot \mathbf{U}, \quad (4.2.9)$$

ahol a formafüggvényeket tartalmazó $\mathbf{N}$ mátrix:

$$\mathbf{N}(\xi, \eta) = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 \end{bmatrix}. \quad (4.2.10)$$

4.3. Az alakváltozási mező létrehozása

Az alakváltozási mező összeállítása során kulcsfontosságú lépés a csomóponti alakváltozások számításának módja, amely alapjául a csomóponti elmozdulások szolgálnak. Az alakváltozások számításánál feltételezzük, hogy képről-képre csak kicsi, szemmel nehezen látható alakváltozások lépnek fel. Ebből kiindulva azt is feltehetjük, hogy a kezdetben a és b oldalhosszúságokkal jellemzett téglalap elemek alakja a mozgás során nem változik, nem torzul. Ezáltal a teljes vizsgált terület is téglalap alakú marad.

4.3.1. A csomóponti alakváltozások és az elemenkénti alakváltozási mező

A kis alakváltozásokra érvényes alakváltozási tenzor mátrixa az alábbi módon írható fel az elmozdulásmező segítségével:

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\nabla \mathbf{u} + (\nabla \mathbf{u})^\top). \quad (4.3.1)$$

Az $x - y$ koordináta-rendszerben:

$$\boldsymbol{\varepsilon}(x, y) = \begin{bmatrix} \varepsilon_x & \varepsilon_{xy} \\ \varepsilon_{xy} & \varepsilon_y \end{bmatrix}, \quad (4.3.2)$$

$$\boldsymbol{\varepsilon}(x, y) = \frac{1}{2} \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} \\ \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{1}{2} \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \\ \frac{1}{2} \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) & \frac{\partial v}{\partial y} \end{bmatrix}. \quad (4.3.3)$$

A 2×2 -es tenzor szimmetrikus, és csupán három különböző elemet tartalmaz. A mellékátlóban szereplő mennyiségek kapcsolata a γ_{xy} fajlagos szögváltozással:

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}, \quad \text{tehát: } \varepsilon_{xy} = \frac{1}{2} \gamma_{xy}. \quad (4.3.4)$$

Innentől a mérnöki életben gyakrabban használt γ_{xy} értéket vesszük alapul a számításokhoz. A végeselemes analízis szempontjából célszerű a három különböző mennyiséget egy oszlopvektorba rendezni, amely egy adott pontban jellemzi a síkbeli alakváltozási állapotot.

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{bmatrix}. \quad (4.3.5)$$

Egy elem lokális koordináta-rendszerének bármely tetszőleges (ξ, η) pontjában interpolálhatók az u és v elmozdulásértékek a 4.2.7 egyenlet összefüggéseivel. Ezeket felhasználva, az alakváltozási mező interpolációját szintén a lokális koordináta-rendszerben végezzük a korábbi megfontolásoknak megfelelően. Mindez alapján egy tetszőleges (ξ, η)

helyen keresendő az

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \frac{\partial}{\partial x} \left(\sum_{i=1}^4 N_i(\xi, \eta) \cdot U_i \right) \\ \frac{\partial}{\partial y} \left(\sum_{i=1}^4 N_i(\xi, \eta) \cdot V_i \right) \\ \frac{\partial}{\partial y} \left(\sum_{i=1}^4 N_i(\xi, \eta) \cdot U_i \right) + \frac{\partial}{\partial x} \left(\sum_{i=1}^4 N_i(\xi, \eta) \cdot V_i \right) \end{bmatrix} \quad (4.3.6)$$

kifejezés, ahol N_i a 4.2.2 – 4.2.5 egyenletekben már bemutatott formafüggvényeket jelöli, U_i és V_i pedig a már ismert csomóponti elmozdulásokat jelöli rendre a vízszintes, illetve függőleges irányokban, a 4.2.3 ábra szerint. A számítás nehézségét az adja, hogy a formafüggvények ξ és η függvényében vannak definiálva, azonban a globális koordináták szerinti parciális deriváltjaikra van szükség az alakváltozás-komponensek felírásához.

A 4.2.8 egyenletben ismertetett $\mathbf{U}$ csomóponti elmozdulásvektor bevezetésével a 4.3.7 egyenletben olvasható alakra hozható az $\boldsymbol{\varepsilon}$ csomóponti alakváltozás-vektor. Ahogy azt a 2.5. fejezetben említettem, a csomóponti alakváltozások számításakor nem játszik szerepet az, hogy az elmozdulások számítása során a merev testszerű mozgások ki lettek-e szűrve vagy sem, hiszen a merev testszerű elmozdulások és rotációk alakváltozási tenzora azonosan nulla. Az U_i és V_i elmozdulásértékek konstansok, a differenciálás tagonként is elvégezhető. Tekintsük tehát az

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \sum_{i=1}^4 \frac{\partial N_i}{\partial x} U_i \\ \sum_{i=1}^4 \frac{\partial N_i}{\partial y} V_i \\ \sum_{i=1}^4 \frac{\partial N_i}{\partial y} U_i + \sum_{i=1}^4 \frac{\partial N_i}{\partial x} V_i \end{bmatrix} = \mathbf{B} \cdot \mathbf{U} \quad (4.3.7)$$

vektort, ahol

$$\mathbf{B} = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_3}{\partial x} & 0 & \frac{\partial N_4}{\partial x} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & 0 & \frac{\partial N_3}{\partial y} & 0 & \frac{\partial N_4}{\partial y} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial y} & \frac{\partial N_3}{\partial x} & \frac{\partial N_4}{\partial y} & \frac{\partial N_4}{\partial x} \end{bmatrix}. \quad (4.3.8)$$

Általánosságban, az i -edik formafüggvény x és y szerinti parciális deriváltjai felírhatók az alábbi módon:

$$\frac{\partial N_i}{\partial x} = \frac{\partial N_i}{\partial \xi} \cdot \frac{\partial \xi}{\partial x} + \frac{\partial N_i}{\partial \eta} \cdot \frac{\partial \eta}{\partial x}, \quad \text{és} \quad \frac{\partial N_i}{\partial y} = \frac{\partial N_i}{\partial \xi} \cdot \frac{\partial \xi}{\partial y} + \frac{\partial N_i}{\partial \eta} \cdot \frac{\partial \eta}{\partial y}. \quad (4.3.9)$$

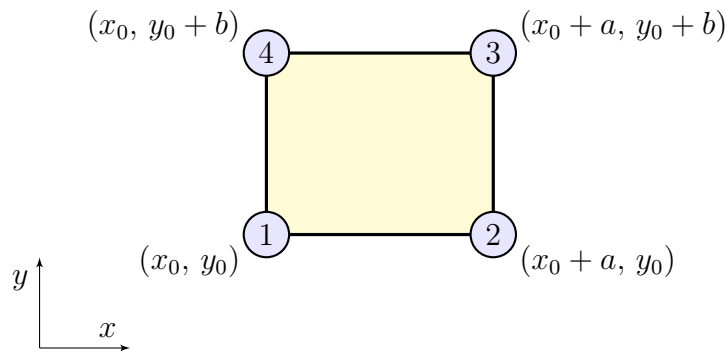
Az így kapott egyenletrendszer mátrixos formában is megadható.

$$\begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} \end{bmatrix}}_{\mathbf{J}^{-1}} \begin{bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{bmatrix}. \quad (4.3.10)$$

Ekkor azonban a $\mathbf{J}^{-1}$ együtthatómátrix közvetlen kiszámítása nehézkes, hiszen szükség lenne a 4.2.1 egyenletben tárgyalt leképzés invertálására. Azonban ha fordítva gondolkodunk, és előbb a formafüggvények lokális koordináta-rendszerbeli deriváltjait fejezzük ki hasonló módon, akkor egyszerűsödik a feladat. Tekintsük tehát a

$$\begin{bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix}}_{\mathbf{J}} \begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{bmatrix} \quad (4.3.11)$$

felírást. Ekkor a $\mathbf{J}$ mátrix könnyen számolható, hiszen a 4.2.1 egyenletben tárgyalt koordináta-transzformáció során éppen ξ és η segítségével írjuk fel x -et és y -t. A 4.3.11 egyenlet bal oldalán szereplő oszlopvektor szintén minden i -re könnyen számolható, hiszen $N_i = N_i(\xi, \eta)$. Így tehát a 4.3.10 egyenletben megfogalmazott lineáris egyenletrendszer $\mathbf{J}$ ismeretében könnyen megoldható.



4.3.1. ábra: Általános helyzetű, téglalap alakú végeelem.

Jelen dolgozat megfontolásai alapján vizsgáljunk meg egy általános helyzetű, a 4.3.1 ábrán látható koordinátákkal definiált, téglalap alakú végelelemet. Eme koordinátákra a 4.3.11 egyenletben szereplő $\mathbf{J}$ mátrixot felírva arra jutunk, hogy:

$$\mathbf{J} = \begin{bmatrix} \frac{a}{2} & 0 \\ 0 & \frac{b}{2} \end{bmatrix}, \quad (4.3.12)$$

(x_0, y_0) -tól független konstans. Így a 4.3.10 egyenlet szerint számolt $\partial N_i / \partial x$ és $\partial N_i / \partial y$ ($i = 1 \dots 4$) értékekből felépített $\mathbf{B}$ mátrix csak a téglalap dimenzióitól, illetve a ξ és η értékektől függ.

$$\mathbf{B} = \begin{bmatrix} \frac{\eta-1}{2a} & 0 & -\frac{\eta-1}{2a} & 0 & \frac{\eta+1}{2a} & 0 & -\frac{\eta+1}{2a} & 0 \\ 0 & \frac{\xi-1}{2b} & 0 & -\frac{\xi+1}{2b} & 0 & \frac{\xi+1}{2b} & 0 & -\frac{\xi-1}{2b} \\ \frac{\xi-1}{2b} & \frac{\eta-1}{2a} & -\frac{\xi+1}{2b} & -\frac{\eta-1}{2a} & \frac{\xi+1}{2b} & \frac{\eta+1}{2a} & -\frac{\xi-1}{2b} & -\frac{\eta+1}{2a} \end{bmatrix}. \quad (4.3.13)$$

A téglalapok a , b oldalhosszúságai ekkor már ismert konstansok, hiszen azokat a felhasználó adja meg a program indítása előtt. Eszerint egy adott elemen belüli alakváltozások immár könnyen számolhatók a

$$\boldsymbol{\varepsilon} = \mathbf{B}(\xi, \eta) \cdot \mathbf{U} \quad (4.3.14)$$

összefüggéssel. A csomóponti alakváltozás-vektorok a megfelelő (ξ, η) koordinátapárok behelyettesítésével szintén felírhatók, ahogy azt a 4.3.15 - 4.3.18 egyenletek mutatják. Kezdetben a szoftver a csomóponti értékeket számolja és menti, hiszen ezeknek kiemelt jelentősége van a 4.3.2 fejezetben tárgyalt átlagolás során. Eszerint

$$\boldsymbol{\varepsilon}^{(1)} = \mathbf{B}(-1, -1) \cdot \mathbf{U}, \quad (4.3.15)$$

$$\boldsymbol{\varepsilon}^{(2)} = \mathbf{B}(+1, -1) \cdot \mathbf{U}, \quad (4.3.16)$$

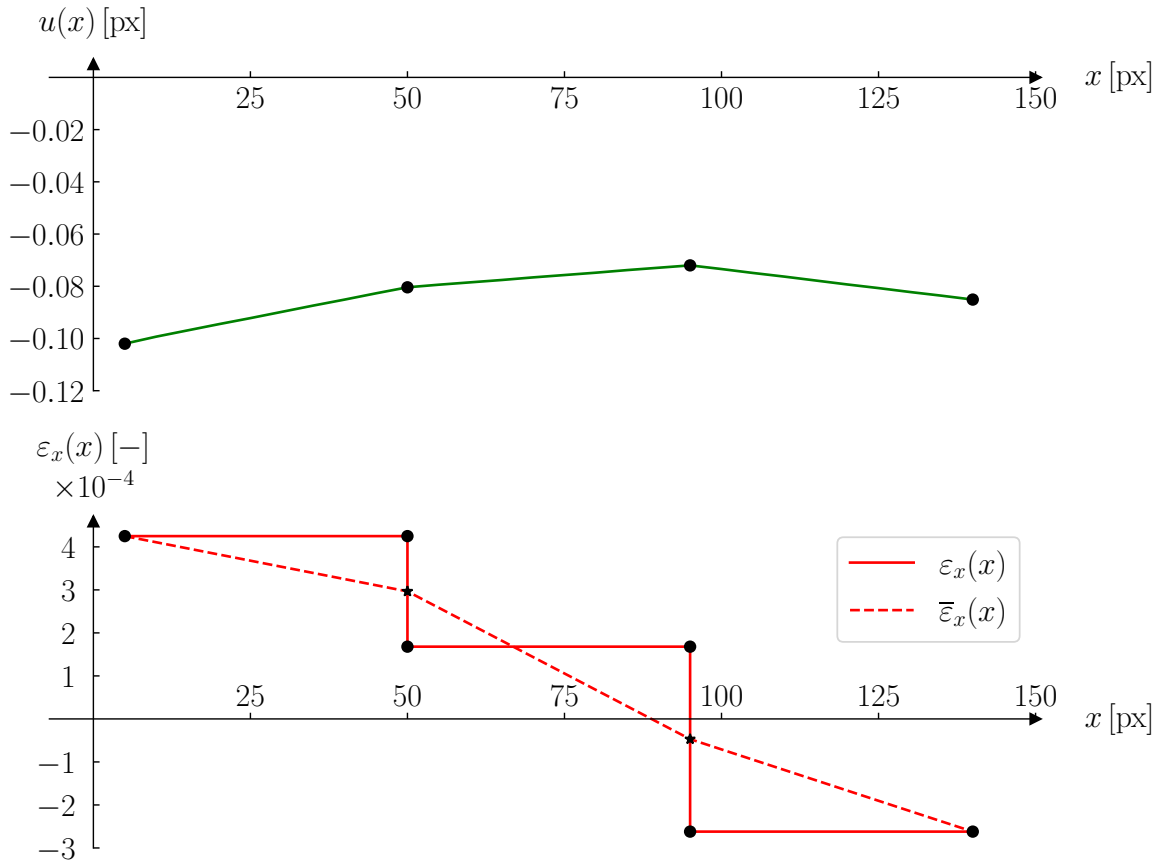
$$\boldsymbol{\varepsilon}^{(3)} = \mathbf{B}(+1, +1) \cdot \mathbf{U}, \quad (4.3.17)$$

$$\boldsymbol{\varepsilon}^{(4)} = \mathbf{B}(-1, +1) \cdot \mathbf{U}, \quad (4.3.18)$$

ahol az $\boldsymbol{\varepsilon}^{(i)}$ kifejezésben szereplő i érték az aktuális lokális csomópontot jelöli, $\mathbf{U}$ pedig az adott elemhez tartozó csomóponti elmozdulásvektor.

4.3.2. A csomóponti átlagolás kérdése

Az alakváltozási mező alapvetően nem folytonos, melynek oka, hogy a lineáris négy-szögelem a csomóponti elmozdulásértékek között lineáris interpolációval teremt kapcsolatot. Vagyis ha például η értékét rögzítjük a lokális koordináta-rendszerben, akkor $u(\xi)$ -re egy lineáris függvénykapcsolat adódik. Ez egyenesen következik a formafüggvények bilineáris jellegéből. Ha az $\varepsilon_\xi = \partial u / \partial \xi$ parciális deriváltat tekintjük, akkor az mindenképpen elemenként konstans lesz. Az említett mennyiségek globális koordináta-rendszerbe történő leképzése ezen a jelenségen nem sokat torzít abban az esetben, ha az elemek jó közelítéssel téglalap alakúak. Vagyis a példánál maradva, fix y -ra egy adott elemen belül az $u(x)$ függvény jó közelítéssel lineáris függvénykapcsolatot mutat, $\varepsilon_x(x)$ pedig közel konstans. Ezt hivatott szemléltetni a 4.3.2 ábrán látható mintapéllda, amelyen az is leolvasható, hogy az illusztráción egy elem ~ 45 [px] széles.



4.3.2. ábra: A csomóponti átlagolás hatása.

A végelelemes szoftverekben gyakran előforduló funkció (és sokszor alapértelmezett beállítás is), hogy a program a szomszédos elemek globális értelemben közös csomópontjainak alakváltozás- és feszültségkomponenseit az interpoláció előtt átlagolja, ezzel biztosítva

a mezőváltozók folytonosságát.

Ezt a funkciót a saját szoftveremben is implementáltam mind az alakváltozások, mind a feszültségek esetére. Az `avg_nodal_strain = True` parancs hatására az összes képkocka összes elemén végigfut egy ciklus, amely az azonos globális koordinátájú, de különböző elemekhez tartozó csomópontok megfelelő alakváltozás-értékeit átlagolja. E folyamatot hivatott illusztrálni a 4.3.3 ábra. Az imént tárgyalt eljárás miatt volt kulcsfontosságú lépés a csomóponti alakváltozások elmentése még az interpoláció előtt. Ezt követően az alakváltozás-értékeket a 4.2.2 – 4.2.5 egyenletekben már ismertetett $N_i(\xi, \eta)$, $i = 1 \dots 4$ formafüggvények segítségével interpolálom az elemek belsejében, az izoparametrikus elemdefiníciónak megfelelően. Az $\boldsymbol{\varepsilon} = \mathbf{B} \cdot \mathbf{U}$ típusú megközelítés jelen esetben nem lenne megfelelő, hiszen éppen ez a felírás okozza az alakváltozási mező szakadásait. Ezzel szemben az N_i formafüggvények használata garantálja az alakváltozási mező folytonosságát, éppen úgy, ahogy korábban az elmozdulásmezőét is. Vagyis legyen

$$\boldsymbol{\varepsilon}(\xi, \eta) = \begin{bmatrix} \varepsilon_x(\xi, \eta) \\ \varepsilon_y(\xi, \eta) \\ \gamma_{xy}(\xi, \eta) \end{bmatrix} = \mathbf{N}(\xi, \eta) \cdot \bar{\boldsymbol{\varepsilon}}, \quad (4.3.19)$$

ahol a formafüggvényeket tartalmazó $\mathbf{N}$ mátrix:

$$\mathbf{N}(\xi, \eta) = \begin{bmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 & N_4 & 0 & 0 \\ 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 & N_4 & 0 \\ 0 & 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 & N_4 \end{bmatrix}, \quad (4.3.20)$$

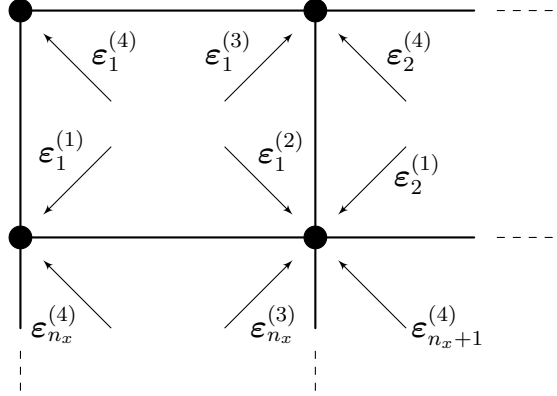
és a már átlagolt csomóponti alakváltozások vektora pedig:

$$\bar{\boldsymbol{\varepsilon}}^T = \begin{bmatrix} \bar{\varepsilon}_{x,1} & \bar{\varepsilon}_{y,1} & \bar{\gamma}_{xy,1} & \bar{\varepsilon}_{x,2} & \bar{\varepsilon}_{y,2} & \bar{\gamma}_{xy,2} & \bar{\varepsilon}_{x,3} & \bar{\varepsilon}_{y,3} & \bar{\gamma}_{xy,3} & \bar{\varepsilon}_{x,4} & \bar{\varepsilon}_{y,4} & \bar{\gamma}_{xy,4} \end{bmatrix}. \quad (4.3.21)$$

Az imént az $\bar{\boldsymbol{\varepsilon}}$ vektor transzponáltját írtam fel a helytakarékoság kedvéért, továbbá a kérdéses vektor elemeinek alsó indexében szereplő számok az adott lokális csomópont sorszámát jelölik.

A 4.3.2 ábra jól mutatja a csomóponti átlagolás hatását az átlagolás nélküli esettel való összevetésben. A mintapéldán jól kivehető az eredeti $\varepsilon_x(x)$ függvény szakaszonként konstans, illetve az átlagolt $\bar{\varepsilon}_x(x)$ függvény szakaszonként lineáris jellege. Természetesen az átlagolás nélküli állapothoz képesti különbség a jelen példán nagyon szembetűnő, az átlagolás utáni $\bar{\varepsilon}_x(x)$ függvény már-már irreális. Valós alkalmazások esetén, megfelelő

sűrűségű háló használatával az átlagolás nélküli, illetve az átlagolt mezőváltozók ábrája között elhanyagolhatóan kicsi különbség van.



4.3.3. ábra: A csomóponti alakváltozások átlagolása.

A 4.3.3 ábrán látható, hogy a rács sarokpontjainál nincs lehetőség átlagolásra, míg a vizsgált terület élein elhelyezkedő további csomópontok esetén kettő vektor számtani közepét képezzük az átlagolás során. A belső csomópontok esetén négy alakváltozási vektor találkozik egy helyen. Az említett ábra ϵ jelöléseit tekintve, alsó indexben szerepel az aktuális elem globális sorszáma, a felső indexben pedig a csomópont lokális sorszáma olvasható, az adott elem felől nézve. Az átlagolás programozása ideiglenes segédváltozók alkalmazásával valósul meg.

4.4. A feszültségmező létrehozása

A programban implementáltam a feszültségmező számításának lehetőségét. E funkció teljes mértékben opcionális, a főprogramban a `Stress_Calc = True` paranccsal lehet aktiválni. Jelen feladat kulcskérdése a csomóponti feszültségek számításának módja, hiszen onnantól az interpoláció és az egyéb ábrázolási eljárások az alakváltozási mezővel teljesen analóg módon történnek.

4.4.1. A csomóponti feszültségek és az elemenkénti feszültségmező

Az alakváltozási tenzor felírásánál kis alakváltozásokat feltételeztünk. Ennek megfelelően, a feszültségek számítására a Hooke-törvényt használom, amely kapcsolatot teremt az ϵ alakváltozási tenzor és a σ feszültségtenzor között. A modell lineárisan rugalmas, homogén, izotrop anyagi viselkedést feltételez:

$$\sigma = \frac{E}{1 + \nu} \left(\epsilon + \frac{\nu}{1 - 2\nu} \epsilon_I \cdot \mathbf{I} \right). \quad (4.4.1)$$

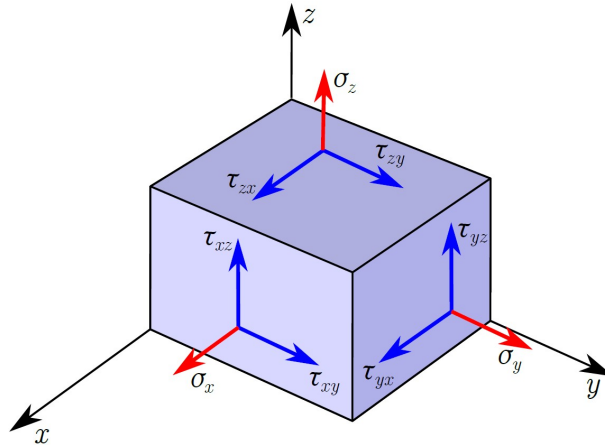
Kettő független anyagparaméterre van szükség a modell használatához, egyrészt az E [Pa] rugalmassági (Young) moduluszra, másrészt a ν [–] Poisson tényezőre. A Hooke-törvényben szereplő $\varepsilon_I = \text{tr } \varepsilon$ érték az alakváltozási tenzor első skalár invariánsát jelöli, ami megegyezik a főátlóbeli elemek összegével, vagyis a mátrix nyomával. Ezenfelül $\mathbf{I}$ jelöli a megfelelő dimenziójú egységmátrixot.

A probléma kétdimenziós, síkbeli jellegéből adódóan eddig nem volt szükségünk az $x - y$ síkra merőleges z irány tárgyalására. Azonban a z irányú kényszerek, illetve azok hiánya hatással van a feszültségek számítási módjára. Ezek alapján különféle feszültségi állapotokat definiálhatunk. Az általam írt program a *síkfeszültségi* állapotra van felkészítve, amelynek részleteiről a későbbiekben értekezek.

A feszültségi és alakváltozási tenzor szimmetriáját felhasználva, a 4.4.1 egyenletben megfogalmazott Hooke-törvény az alábbi alakot ölti:

$$\begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{xy} & \sigma_y & \tau_{yz} \\ \tau_{xz} & \tau_{yz} & \sigma_z \end{bmatrix} = \frac{E}{1 + \nu} \left(\begin{bmatrix} \varepsilon_x & \gamma_{xy}/2 & \gamma_{xz}/2 \\ \gamma_{xy}/2 & \varepsilon_y & \gamma_{yz}/2 \\ \gamma_{xz}/2 & \gamma_{yz}/2 & \varepsilon_z \end{bmatrix} + \frac{\nu}{1 - 2\nu} (\varepsilon_x + \varepsilon_y + \varepsilon_z) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right). \quad (4.4.2)$$

A feszültségkomponensek értelmezését mutatja a 4.4.1 ábra.



4.4.1. ábra: A feszültségi kiskocka^[33].

Síkfeszültségi állapot

A síkfeszültségi állapot feltételezi, hogy a z irányú feszültségkomponensek elhanyagolhatók, tehát $\sigma_z = \tau_{xz} = \tau_{yz} = 0$. Ez a modell z irányban kis vastagságú lemezek esetén ad megfelelő eredményeket, feltéve, hogy a terhelés kizárólag az $x - y$ síkban hat. Tekintve, hogy egy kamerával praktikusán ilyen jellegű kétdimenziós problémákról tudunk

felvételeket készíteni, célszerű volt a szoftverben ezt a feszültségi állapotot implementálni. Ekkor a feszültségi tenzor mátrixa:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x & \tau_{xy} & 0 \\ \tau_{xy} & \sigma_y & 0 \\ 0 & 0 & 0 \end{bmatrix}. \quad (4.4.3)$$

Célunk a tenzorban szereplő három különböző feszültségkomponens kifejezése az alakváltozási komponensek függvényében. Kezdetben a $\sigma_z = 0$ feltételt kihasználva, a Hooke-törvény 4.4.2 egyenletben olvasható formájából ε_z -re a következő összefüggést kapjuk:

$$\varepsilon_z = \frac{-\nu}{1-\nu}(\varepsilon_x + \varepsilon_y). \quad (4.4.4)$$

Ennek segítségével a három keresett feszültségkomponens könnyen kifejezhető a 4.4.2 egyenlet ismételt felhasználásával.

$$\sigma_x = \frac{E}{1-\nu^2} \cdot \varepsilon_x + \frac{\nu E}{1-\nu^2} \cdot \varepsilon_y, \quad (4.4.5)$$

$$\sigma_y = \frac{\nu E}{1-\nu^2} \cdot \varepsilon_x + \frac{E}{1-\nu^2} \cdot \varepsilon_y, \quad (4.4.6)$$

$$\tau_{xy} = \frac{E}{2(1+\nu)} \cdot \gamma_{xy} = \frac{E}{1-\nu^2} \cdot \frac{1-\nu}{2} \cdot \gamma_{xy}. \quad (4.4.7)$$

A három különböző feszültségkomponenst az alábbi oszlopvektorba rendezve,

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix}, \quad (4.4.8)$$

az alakváltozásokkal való kapcsolat mátrixos formába is rendezhető:

$$\boldsymbol{\sigma} = \mathbf{D} \cdot \boldsymbol{\varepsilon}. \quad (4.4.9)$$

A 4.4.5 - 4.4.7 egyenletek alapján a síkfeszültségi állapotra az alábbi együtthatómátrix érvényes:

$$\mathbf{D} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix}. \quad (4.4.10)$$

Az elemen belüli interpoláció alapértelmezés szerint a 4.3.1 fejezetben bevezetett $\mathbf{B}$ mátrixon alapul, hiszen

$$\boldsymbol{\sigma} = \mathbf{D} \cdot \boldsymbol{\varepsilon} = \mathbf{D} \cdot \mathbf{B}(\xi, \eta) \cdot \mathbf{U}, \quad (4.4.11)$$

ahol $\mathbf{D}$ konstans (csak az anyagparaméterektől függ), illetve az $\mathbf{U}$ csomóponti elmozdulásvektort a digitális képkorrelációs algoritmus már meghatározta.

Az interpoláció első lépéseként azonban a csomóponti feszültségkomponenseket határozzuk meg a 4.4.12 - 4.4.15 egyenletek szerint, elmentve őket minden végelem minden csomópontjához tartozóan. Ezzel megteremtjük a csomóponti átlagolás lehetőségét a feszültségmezőre vonatkozóan is.

$$\boldsymbol{\sigma}^{(1)} = \mathbf{D} \cdot \mathbf{B}(-1, -1) \cdot \mathbf{U}, \quad (4.4.12)$$

$$\boldsymbol{\sigma}^{(2)} = \mathbf{D} \cdot \mathbf{B}(+1, -1) \cdot \mathbf{U}, \quad (4.4.13)$$

$$\boldsymbol{\sigma}^{(3)} = \mathbf{D} \cdot \mathbf{B}(+1, +1) \cdot \mathbf{U}, \quad (4.4.14)$$

$$\boldsymbol{\sigma}^{(4)} = \mathbf{D} \cdot \mathbf{B}(-1, +1) \cdot \mathbf{U}, \quad (4.4.15)$$

4.4.2. Csomóponti átlagolás

Ahogy azt a 4.3.2 fejezetben említettem, a csomóponti átlagolás lehetőségét a feszültségértékekre is implementáltam. A Hooke-törvényből adódóan a különböző feszültségkomponensek az alakváltozások lineáris kombinációjaként felírhatók. Ebből következik, hogy az alakváltozási mezőhöz hasonlóan a feszültségmező sem folytonos. Mindez a négycsomópontos lineáris négyszögelem azon tulajdonságából következik, hogy az elmozdulásmező interpolációját a N_i , $i = 1 \dots 4$ bilineáris formafüggvények végzik (4.2.2 – 4.2.5), amelyek lokális koordináta-rendszerbeli parciális deriváltjai egy adott elemen belül konstansok. Ezért születhetnek két szomszédos elem egy közös csomópontjában eltérő alakváltozás-, illetve feszültségértékek, ahogy az a 4.3.2 ábrán is látható.

Az `avg_nodal_stress = True` parancs hatására egy ciklus az összes képkocka összes elemén végigfut, és az egyszerre több elemhez tartozó csomópontok feszültségértékeit átlagolja. Habár a 4.3.3 ábra az alakváltozások átlagolását mutatja be, a feszültségvektorok átlagolása teljesen analóg módon történik. Ezt követően a 4.4.11 egyenletben megfogalmazott interpolációs eljárás alkalmazása nem lenne megfelelő, hiszen az elmozdulásértékekre és a $\mathbf{B}$ mátrixra támaszkodva szakadásokkal teli feszültségmezőt kapnánk. Ehelyett az izoparametrikus elemdefiníciót használva, ismét az N_i formafüggvények segítségével interpoláljuk a már átlagolt csomóponti értékeket ($\bar{\boldsymbol{\sigma}}$), ahogy azt az alakváltozások esetén is

tettük, vagyis:

$$\boldsymbol{\sigma}(\xi, \eta) = \begin{bmatrix} \sigma_x(\xi, \eta) \\ \sigma_y(\xi, \eta) \\ \tau_{xy}(\xi, \eta) \end{bmatrix} = \mathbf{N}(\xi, \eta) \cdot \bar{\boldsymbol{\sigma}}, \quad (4.4.16)$$

ahol a formafüggvények mátrixa:

$$\mathbf{N}(\xi, \eta) = \begin{bmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 & N_4 & 0 & 0 \\ 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 & N_4 & 0 \\ 0 & 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 & N_4 \end{bmatrix}, \quad (4.4.17)$$

a már átlagolt feszültségkomponensek vektora pedig:

$$\bar{\boldsymbol{\sigma}}^T = \begin{bmatrix} \bar{\sigma}_{x,1} & \bar{\sigma}_{y,1} & \bar{\tau}_{xy,1} & \bar{\sigma}_{x,2} & \bar{\sigma}_{y,2} & \bar{\tau}_{xy,2} & \bar{\sigma}_{x,3} & \bar{\sigma}_{y,3} & \bar{\tau}_{xy,3} & \bar{\sigma}_{x,4} & \bar{\sigma}_{y,4} & \bar{\tau}_{xy,4} \end{bmatrix}. \quad (4.4.18)$$

A jobb helykihasználás érdekében a kérdéses oszlopvektor transzponáltját írtam fel az imént. Így tehát az elemeken belüli feszültségmező interpolációja csomóponti átlagolással (lásd 4.4.16 egyenlet), illetve anélkül is lehetséges a 4.4.11 egyenletben foglaltak szerint.

A feszültségértékek átlagolásának hatását jól összefoglalja a 4.3.2 ábra. A pirossal ábrázolt függvények ugyan ε_x értékeket jelenítenek meg, ám ez a lényegen nem változtat, hiszen a feszültségértékek a különböző alakváltozások lineáris kombinációjaként adódnak, tehát az ábrák jellegre helyesek a feszültségek szempontjából is.

4.4.3. A von Mises-féle egyenértékű feszültség számítása

A mérnöki életben, különösen a szilárdságtani méretezés során gyakran előáll az a kihívás, hogy a többtengelyű feszültségű állapotot (vagyis a többbelemű feszültségtenzort) szükséges egy úgynevezett σ_{meg} megengedett feszültséggel összehasonlítani. Ezen probléma feloldására születtek meg a különböző egyenértékű feszültségelméletek, amelyek a tenzor elemeiből egy skalármennyiséget állítanak elő, ezzel jellemezve a többtengelyű feszültségi állapotot.

Jelen dolgozatban a Huber-Mises-Hencky-elmélet szerinti egyenértékű feszültség számítását valósítottam meg. A módszer röviden HMH-elméletként is ismert, a külföldi szaknyelvben azonban a *von Mises equivalent stress*^[34] megnevezés terjedt el leginkább, utalva ezzel Richard von Mises osztrák matematikusra.

Egy adott pontra vonatkozó általános $\boldsymbol{\sigma}$ feszültségtenzor esetén a von Mises egyenér-

tékű feszültség a következőképpen számítható:

$$\sigma_{\text{vM}} = \sqrt{\frac{(\sigma_x - \sigma_y)^2 + (\sigma_y - \sigma_z)^2 + (\sigma_z - \sigma_x)^2 + 6(\tau_{xy}^2 + \tau_{xz}^2 + \tau_{yz}^2)}{2}}. \quad (4.4.19)$$

A síkfeszültségi állapot 4.4.3 egyenletben szereplő feszültségi tenzorát figyelembe véve, az egyenértékű feszültség kifejezése egyszerűbb alakra hozható:

$$\sigma_{\text{vM}} = \sqrt{\sigma_x^2 - \sigma_x \sigma_y + \sigma_y^2 + 3\tau_{xy}^2}. \quad (4.4.20)$$

A globális feszültségi mezők összeállítását követően a teljes vizsgált területre vonatkozó von Mises-féle egyenértékű feszültségmező a 4.4.20 egyenlettel könnyedén számítható.

Jelen fejezet eredményeképpen az elmozdulás-, az alakváltozás- és a feszültségkomponensek elemen belüli interpolációjának módszere immár ismert, ezen részeredmények globális mezőkké való összeállítását az 5. fejezetben részletezem.

5. A program kimenetei, eredményei

A szoftver többféleképpen tudja prezentálni a számítások eredményeit, részeredményeit. A program futása közben számos szövegfájl készül, emellett a mezőváltozókat ábrázoló illusztrációk is automatikusan mentésre kerülnek, az értelmezést segítő színskálákkal együtt. Ezek ráillesztése az eredeti videóra opcionális, vagyis a felhasználó választhatja ki azokat a mezőváltozókat, amelyekből videót kíván készíteni.

5.1. A globális mezők összeállítása, ábrázolása

A program által számított összes mezőváltozó adott elemen belüli interpolációja a 4. fejezetben bemutatásra került. Egy tetszőleges f mezőváltozó globális ábrája általános megközelítés szerint az egyes elemek felett értelmezett $f_k(x, y)$ összefüggések külön-külön létrehozott ábráinak az összeillesztéséből születik. Jelen esetben $k = 1 \dots (n_x - 1)(n_y - 1)$, az adott elem globális sorszámát jelöli az adott képkockán. A módszerrel kapcsolatban kettő kérdéskör merül fel. Először is, szükségünk van a 4.2.1 egyenletben leírt koordináta-transzformáció inverzére ahhoz, hogy a lokális koordináta-rendszerekben felírt $f_k(\xi, \eta)$ függvények használhatók legyenek x és y függvényében. A 4.3.1 ábrán látható, (x_0, y_0) bal alsó sarokponti koordinátákkal rendelkező, a, b oldalhosszúságú téglalap elem esetén:

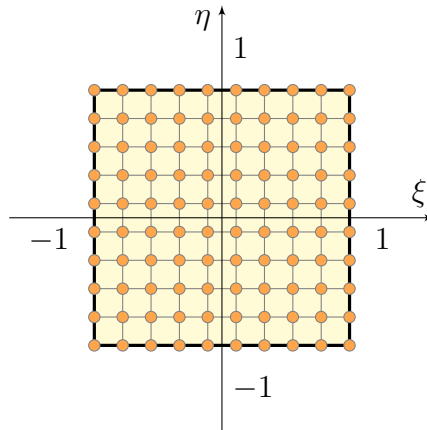
$$\xi = -\frac{a - 2x + 2x_0}{a} \quad \text{és} \quad \eta = -\frac{b - 2y + 2y_0}{b}. \quad (5.1.1)$$

A szabályos téglalap geometria okán az inverz transzformáció felírása nem okoz nagy problémát, így az $f_k(x, y)$ függvények már ábrázolhatók az egyes elemek felett. Sokkal nagyobb terhet (vagyis megnövekedett futásidőt) jelent az a megfontolás, miszerint az egyes elemek feletti ábrák összessége alkotja a mezőváltozó globális illusztrációját. Az ábrák külön-külön történő elkészítése, majd végül összeillesztése nagyon erőforrásigényes.

A saját programom az imént tárgyalt kettő kérdéskörre alternatív megoldásokat nyújt, amelyek célja a futásidő csökkentése. Mindenekelőtt, a $(\xi, \eta) \rightarrow (x, y)$ transzformációt mellőzi a szoftver, hiszen a szabályos téglalap elemek miatt egy tetszőleges $f_k(x, y)$ függvény képével megegyező illusztrációt kapunk eredményül abban az esetben is, hogyha a lokális koordináta-rendszer négyzet alakú értelmezési tartományán végezzük a kapcsolódó $f_k(\xi, \eta)$ ábrázolását, majd az eredményt a megfelelő dimenziójú téglalappá méretezzük át.

Másrészt, a mezőváltozók egyes elemek feletti külön-külön ábrázolása helyett a szoft-

ver egy adott sűrűségű mintavételezéssel minden elem felett kiszámítja a mezőváltozók értékét ξ és η függvényében. Tekintve, hogy a $\xi - \eta$ sík értelmezési tartománya a $[-1, 1] \times [-1, 1]$ négyzet, az interpolációra használt pontpárok kiválasztása egy $m \times m$ -es pontrács kialakításával célszerű, ahogy azt a 5.1.1 ábra mutatja. Ezen módszerrel minden elemre vonatkozóan kapunk egy interpolált mezőváltozót tartalmazó kisebb ($m \times m$ -es) mátrixot, amelyeket aztán egy nagyméretű mátrixban összeillesztünk, ez utóbbi pedig egy lépésben kerül ábrázolásra. Az ily módon összeállított globális mátrixok $m(n_y - 1) \times m(n_x - 1)$ méretűek.



5.1.1. ábra: Interpolációs rács $m = 10$ -es beosztással.

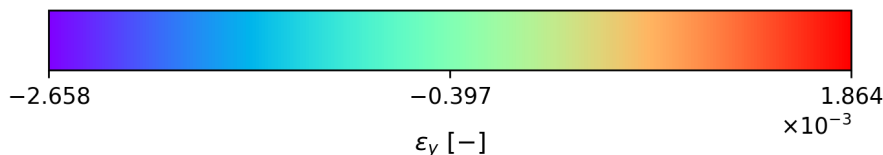
A globális mátrixokban szereplő adatokat a `Matplotlib`^[35] könyvtár `imshow` függvényével ábrázoljuk, elkészítve ezzel a bemeneti videóra illesztendő ábrákat.

A színes illusztrációk legfontosabb paraméterei közé tartoznak a színskála szélsőértékei. Ha az alapértelmezett beállításokkal végeznénk az ábrázolást, akkor a színskála szélsőértékei a mozgás során képről-képre változnának, mindig az aktuális képkockára vonatkozó minimumot-maximumot venné alapul az `imshow` függvény. Ezt a problémát úgy oldhatjuk meg, hogy még az interpolációs fázisban folyamatosan frissítjük az aktuális mezőváltozó szélsőértékeit, amelyeket egy-egy globális változóban tárolunk, később pedig ezeket használjuk az ábrázolás során. Így tehát a színskála szélsőértékeit valóban az adott mezőváltozó teljes mozgás során előforduló minimuma és maximuma adja, konzekvensen ezeket alkalmazzuk minden képkockán.

Egy másik fontos paraméter az úgynevezett α érték, amely a készített kép átlátszóságát kontrollálja. Tekintve, hogy a bemeneti videóra való ráillesztésnél előnyös, ha nem egy teljesen átlátszatlan ($\alpha = 1$) képpel fedjük el a vizsgált területet, így a felhasználó ezt a paramétert tetszés szerint módosíthatja. A munkám során úgy találtam, hogy az $\alpha = 0.75$ érték ideális eredményt nyújt, ezért a programomban ez vált alapértelmezetté.

5.2. A színes illusztrációk jelmagyarázata

Természetesen a mezőváltozók színes illusztrációi jelmagyarázat nélkül nem adnak teljes képet a vizsgált jelenségről. A színárnyalatokhoz tartozó értékeket az úgynevezett *colorbar* tartalmazza, jeleníti meg. Egy ilyen jelmagyarázatot mutat az 5.2.1 ábra.



5.2.1. ábra: Egy ε_y mezőhöz tartozó *colorbar*.

A fentihez hasonló jelmagyarázatok készítésénél ismét szerephez jutnak az épp aktuális mezőváltozó szélsőértékei, ám ezenkívül még fontos paraméter a vízszintes tengelyen megjelenő értékek száma is (*number of ticks*). A felhasználó ezt is tetszőlegesen módosíthatja.

Tekintve, hogy a jelmagyarázat ábrája is a bemeneti videóra illesztendő, így fontos szempont a generált kép mérete. A *colorbar* alapértelmezett szélességének a teljes képkocka szélességének harmadát választottam, a magasság esetén pedig a negyvennel való osztás vált be számomra, arányaiban. Természetesen a felhasználó számára biztosított egy **resize_factor** nevezetű skálázási paraméter, amellyel a jelmagyarázat méretét tudja arányosan módosítani a főprogramból.

A vizsgált videó jellegétől függ az is, hogy a képkocka mely részére érdemes elhelyezni a *colorbar* színskáláját. A felhasználó itt is saját belátása szerint dönthet a bal felső sarok koordinátáiról a *colorbar_x0* és *colorbar_y0* paraméterek használatával.

5.3. Videófelvételek létrehozása

Az előző alfejezetekben a mezőváltozók globális ábráiról, illetve a színárnyalatok jelenítését magyarázó *colorbarokról* esett szó. Egy adott kimeneti videó létrehozásakor mindezeket az eredeti videó képkockáira illesztjük. Az így létrejött képsorozatot az **OpenCV**^[36] könyvtár használatával videóvá összefűzve megkapjuk a program egyik fő eredményét. Mindemellett a felhasználó dönthet úgy, hogy némely mezőváltozó eredményeiből nem kíván videót készíteni.

A videófelvételek létrehozásánál fontos szempont, hogy a mezőváltozók ábráinak pozícióját, videón való elhelyezkedését éppen a mozgás miatt korrigálni kell. A végeselemes

háló bal felső csomópontjának itt kitüntetett szerepe van, hiszen a ráillesztés során az alul fekvő képkocka koordináta-rendszerében azt az (x_0, y_0) pontot kell megadni, ahova az illesztendő kép bal felső sarka kerül. Ez azt jelenti, hogy a bal felső, vagyis a 4.1.2 ábra szerinti globális ①-es csomópont (x, y) koordinátájának időfüggését figyelembe véve szükséges a színes illusztrációkat az eredeti képkockákra illeszteni.

5.4. Részeredményeket tartalmazó szövegfájlok

A szövegfájloknak kiemelt jelentősége van a részeredmények ellenőrzése során. A program minden főbb lépéséhez, funkciójához tartoznak ilyen jellegű ellenőrzési lehetőségek.

Így például vizsgálható a képkorrelációs algoritmus által megtalált pontok helyzete az idő függvényében, a kezdeti rácspontok koordinátáihoz viszonyítva. Ehhez hasonlóan a kiszámított elmozdulásértékek is megtekinthetők egy külön fájlban. Ezen részeredményekből következtethetünk a mozgáskövető algoritmus paramétereinek jóságára, avagy a próbatest mintázatának minőségére. Ha például egy 20-as osztásközű pontrácson egy adott pontot az algoritmus ~ 40 pixel távolságra talál meg a következő képkockán, akkor tudjuk, hogy a további eredmények helytelenek, tehát valamin változtatnunk kell.

5.4.1. táblázat: Egy csomóponti alakváltozásokat tartalmazó fájl első néhány sora.

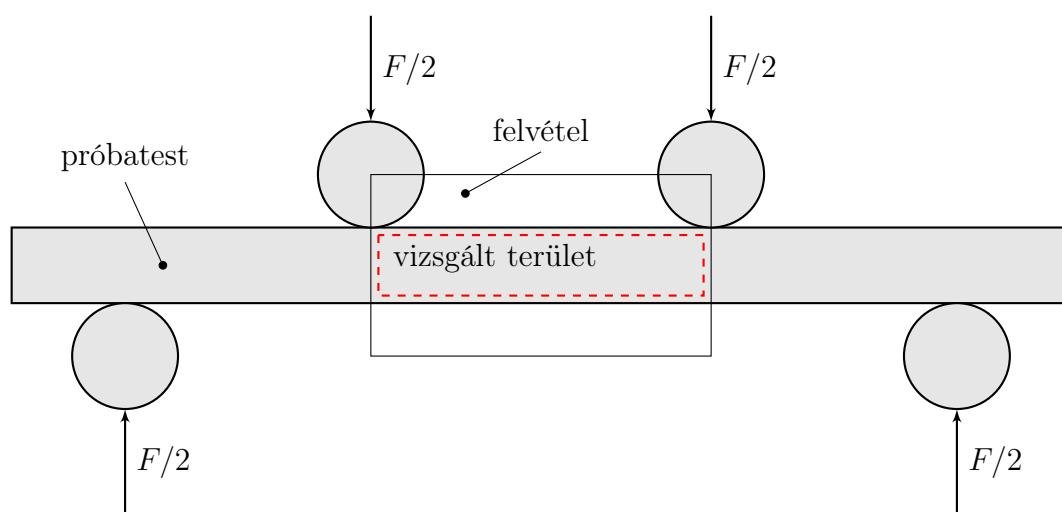
id	loc1	loc2	loc3	loc4	local1stn	local2stn	local3stn	local4stn
1	52	53	2	1	$\begin{bmatrix} -5.43 \cdot 10^{-4} \\ -7.99 \cdot 10^{-4} \\ -4.53 \cdot 10^{-3} \end{bmatrix}$	$\begin{bmatrix} -4.98 \cdot 10^{-4} \\ -1.60 \cdot 10^{-3} \\ -7.48 \cdot 10^{-3} \end{bmatrix}$	$\begin{bmatrix} -3.44 \cdot 10^{-4} \\ 2.58 \cdot 10^{-4} \\ -6.71 \cdot 10^{-3} \end{bmatrix}$	$\begin{bmatrix} -9.86 \cdot 10^{-4} \\ -1.84 \cdot 10^{-3} \\ -6.86 \cdot 10^{-3} \end{bmatrix}$
2	53	54	3	2	$\begin{bmatrix} -4.98 \cdot 10^{-4} \\ -1.60 \cdot 10^{-3} \\ -7.48 \cdot 10^{-3} \end{bmatrix}$	$\begin{bmatrix} -7.17 \cdot 10^{-4} \\ 8.95 \cdot 10^{-5} \\ -6.86 \cdot 10^{-3} \end{bmatrix}$	$\begin{bmatrix} -7.89 \cdot 10^{-4} \\ -5.53 \cdot 10^{-4} \\ -6.04 \cdot 10^{-3} \end{bmatrix}$	$\begin{bmatrix} -3.44 \cdot 10^{-4} \\ 2.58 \cdot 10^{-4} \\ -6.71 \cdot 10^{-3} \end{bmatrix}$
3	54	55	4	3	$\begin{bmatrix} -7.17 \cdot 10^{-4} \\ 8.95 \cdot 10^{-5} \\ -6.86 \cdot 10^{-3} \end{bmatrix}$	$\begin{bmatrix} -2.66 \cdot 10^{-4} \\ -1.09 \cdot 10^{-3} \\ -7.01 \cdot 10^{-3} \end{bmatrix}$	$\begin{bmatrix} -5.82 \cdot 10^{-4} \\ 1.43 \cdot 10^{-4} \\ -5.95 \cdot 10^{-3} \end{bmatrix}$	$\begin{bmatrix} -7.89 \cdot 10^{-4} \\ -5.53 \cdot 10^{-4} \\ -6.04 \cdot 10^{-3} \end{bmatrix}$
4	55	56	5	4	$\begin{bmatrix} -2.66 \cdot 10^{-4} \\ -1.09 \cdot 10^{-3} \\ -7.01 \cdot 10^{-3} \end{bmatrix}$	$\begin{bmatrix} -1.16 \cdot 10^{-3} \\ -5.58 \cdot 10^{-4} \\ -6.60 \cdot 10^{-3} \end{bmatrix}$	$\begin{bmatrix} -9.73 \cdot 10^{-4} \\ -3.15 \cdot 10^{-4} \\ -5.49 \cdot 10^{-3} \end{bmatrix}$	$\begin{bmatrix} -5.82 \cdot 10^{-4} \\ 1.43 \cdot 10^{-4} \\ -5.95 \cdot 10^{-3} \end{bmatrix}$
5	56	57	6	5	$\begin{bmatrix} -1.16 \cdot 10^{-3} \\ -5.58 \cdot 10^{-4} \\ -6.60 \cdot 10^{-3} \end{bmatrix}$	$\begin{bmatrix} -3.48 \cdot 10^{-4} \\ -1.11 \cdot 10^{-3} \\ -5.43 \cdot 10^{-3} \end{bmatrix}$	$\begin{bmatrix} -3.45 \cdot 10^{-4} \\ -2.15 \cdot 10^{-4} \\ -5.06 \cdot 10^{-3} \end{bmatrix}$	$\begin{bmatrix} -9.73 \cdot 10^{-4} \\ -3.15 \cdot 10^{-4} \\ -5.49 \cdot 10^{-3} \end{bmatrix}$

Az elmozdulás-, az alakváltozási- és a feszültségmező esetében a program képkockánként készít egy-egy olyan fájlt, amely a végelemekről tartalmaz különféle információkat. Egy ilyen jellegű fájl elrendezését mutatja az 5.4.1 táblázat. Az elemek a globális sorszá-

muk szerint következnek egymás után, valamint az adott elemhez tartozó lokális csomópontok globális sorszámai mellett az éppen aktuális mezőváltozók csomóponti értékei is olvashatók. Ez jól tükrözi az interpoláció előtti állapotokat, továbbá a 4.1.2 ábrán látható számozások teljesülése is visszaellenőrizhető. Az alakváltozási, illetve feszültségi mezők esetében pedig a csomóponti átlagolás megvalósulásáról is tájékozódhatunk.

5.5. A fejlesztéshez használt videófelvétel bemutatása

A program fejlesztése, tesztelése során a pydic^[14] szoftverhez tartozó egyik mintapéldát használtam fel. A példa egy négyponthas hajlítást ábrázoló képsorozat, amelyből a programom logikájának megfelelően egy videós anyagot készítettem bemenet gyanánt. A próbatest pettyes mintázatát kielégítőnek találtam, emellett megítélésem szerint a négyponthas hajlítás kellően összetett, de mégis jól jósolható eredményeket nyújt a szoftver teszteléséhez. A mérési összeállítást az 5.5.1 ábra mutatja. A felvétel 2560×1920 -as felbontású, ezen belül a vizsgált terület 2550×600 -es méretű az 5.6. fejezetben bemutatott eredmények esetében.



5.5.1. ábra: A négyponthas hajlítás mérési összeállítása, sematikus ábra.

5.6. A fő funkciók bemutatása

Ebben a fejezetben bemutatom a szoftver fő funkcióinak hatását a négyponthas hajlítás mintapéldáján. Az említett funkciók közé tartozik a merev testszerű mozgások kiszűrése, a csomóponti alakváltozások átlagolása, illetve a csomóponti feszültségek átlagolása. Ezen opciók logikai változókkal kapcsolhatók ki/be a főprogramban. Megjegyzendő, hogy a három változó bármilyen kombinációja működik, vagyis például az alakváltozások átlago-

lása nem vonja szükségszerűen maga után a feszültségek átlagolását. Jelen fejezetben nem cél a főprogramban beállítható további paraméterek vizsgálata, ideértve a Lucas-Kanade mozgáskövető algoritmus paramétereit is.

A futtatások során a végelelemes háló osztásköze mindkét irányban 50 [px] volt, így a háló 612 csomópontot és 550 elemet tartalmazott. Az elemeken belüli interpolációhoz $m = 10$ -es felbontást használtam. A vizsgálatok során használt optical flow paraméterek az 5.6.1 ábrán olvashatók.

```

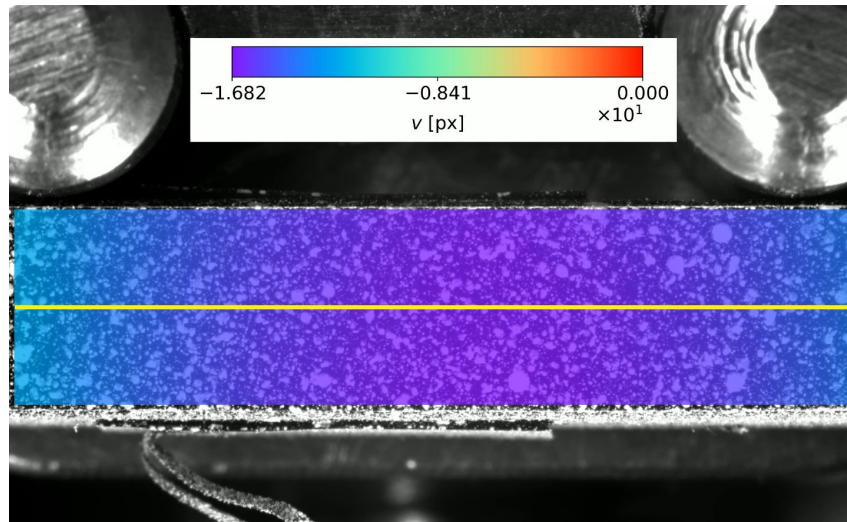
1      winSize = (80,80)
2      maxLevel = 5
3      criteria.epsilon = 0.03
4      criteria.maxCount = 10
5      minEigThreshold = None

```

5.6.1. ábra: A négyponthoz használt mozgáskövetési paraméterek.

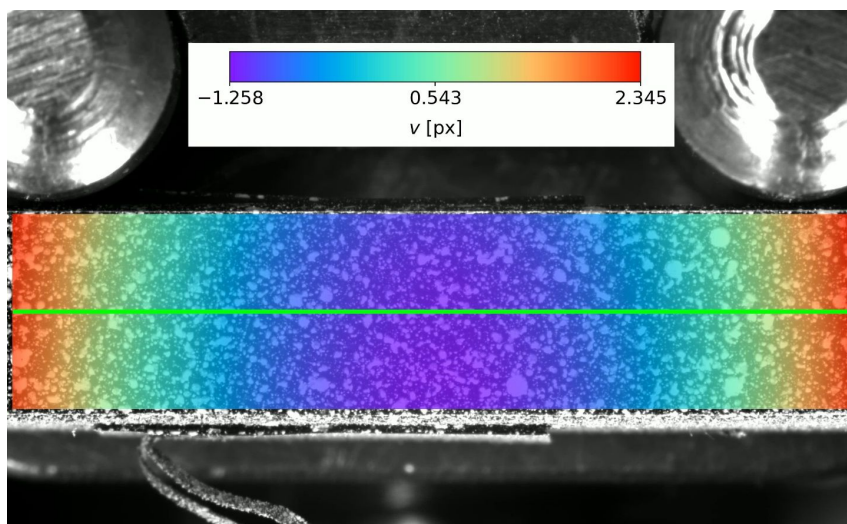
5.6.1. Merev testszerű mozgások kiszűrése

Az alábbi eredmények a négyponthoz használt mozgáskövetési paraméterek $v(x, y)$ elmozdulásmezőt mutatják be a merev testszerű mozgások figyelembevételével, illetve azok kiszűrésével.



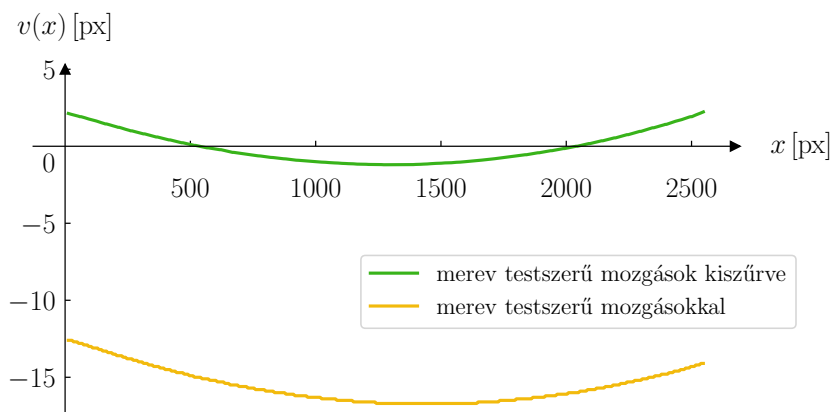
5.6.2. ábra: Négyponthoz használt $v(x, y)$ mezeje merev testszerű mozgásokkal.

Az 5.6.2 és az 5.6.3 ábrán látható színskálák különbözőségét az okozza, hogy a be-
meneti videófelvételen a próbatest a terhelés alatt jól láthatóan, szinte merev testszerűen
mozog $-y$ irányba. Ez a jelenség minden képkockán közel homogén v elmozdulásmezőt
eredményez, az apró részletek elvesznek. A színskála felső határa ebben az esetben az első
képkocka azonosan nulla elmozdulásmezejéből adódik.



5.6.3. ábra: Négypontos hajlítás $v(x, y)$ mezeje a merev testszerű mozgások kiszűrésével.

Ehhez képest, a merev testszerű mozgások kiszűrése után egyértelművé válik, hogy a próbatest bal és jobb szélső területei $+y$ irányba mozdulnak el, míg a középső tartomány ténylegesen negatív elmozdulásértékeket mutat a terhelés alatt. Mindkét említett ábra az adott kimeneti videó utolsó képkockáját mutatja be.

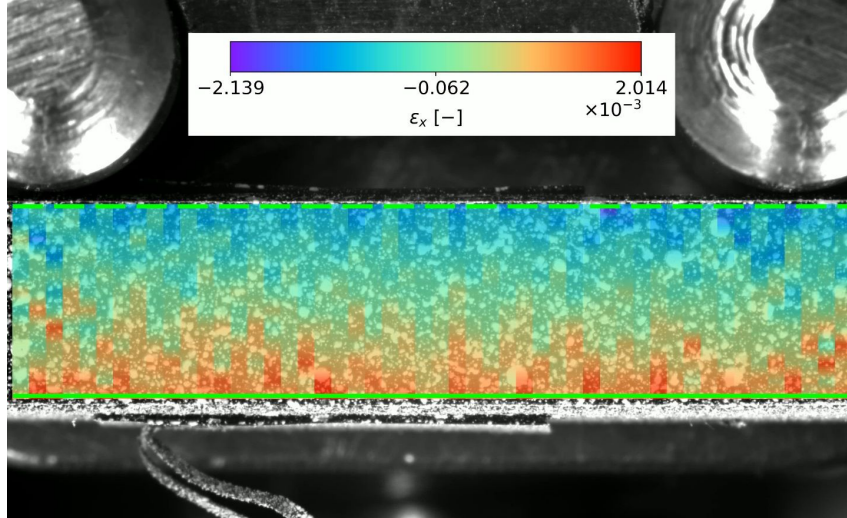


5.6.4. ábra: Négypontos hajlítás $v(x)$ mezeje fix y magasságon.

Az 5.6.4 ábrán könnyen összehasonlíthatók a függőleges elmozdulások értékei az 5.6.2 és az 5.6.3 ábrákon jelölt sárga, illetve zöld útvonalak mentén. Látható, hogy a próbatest a videófelvétel elejéhez képest, nagyjából egyenletesen -15 pixelt mozdult el függőlegesen, azonban a részletek jobban kiéleződnek, ha a merev testszerű mozgások kiszűrését választjuk.

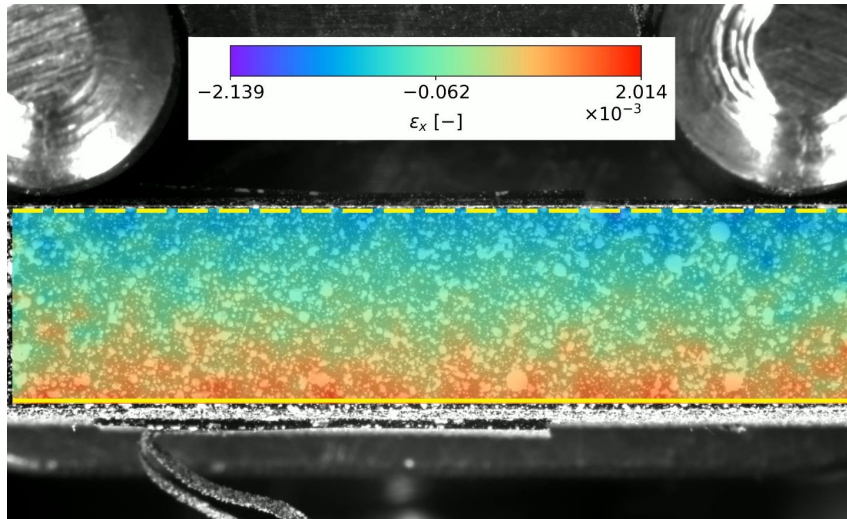
5.6.2. Csomóponti alakváltozások átlagolása

A következő eredmények a négy pontos hajlítás $\varepsilon_x(x, y)$ mezejét mutatják be csomóponti átlagolással, illetve anélkül. Az illusztrációk alapját ismét a kimeneti videók utolsó képkockái szolgáltatják.



5.6.5. ábra: Négy pontos hajlítás $\varepsilon_x(x, y)$ mezeje csomóponti átlagolás nélkül.

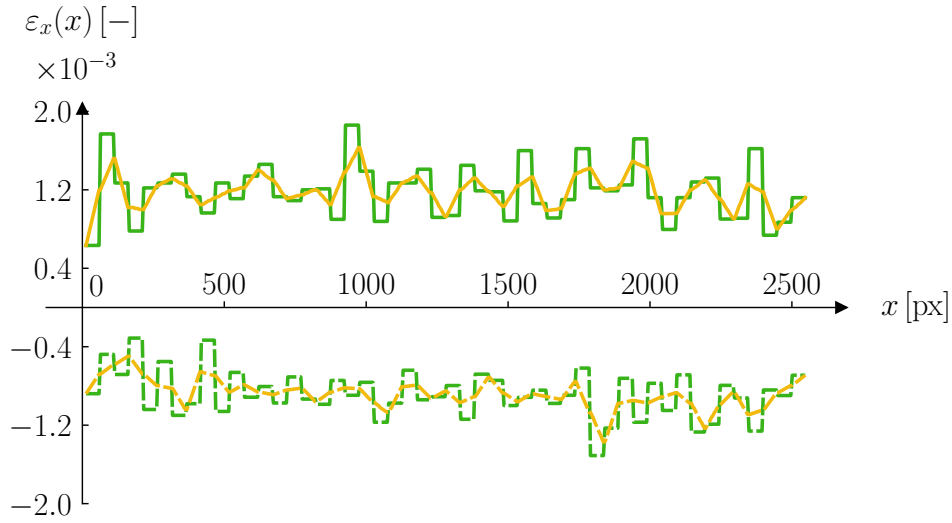
Az 5.6.5 és 5.6.6 ábrákon újfent elhelyeztem néhány útvonalat, amelyek mentén összehasonlíthatóvá válnak az x irányú alakváltozások. Mindkettő ábrán szaggatott vonalat helyeztem a nyomott oldalra, a húzott oldalra pedig összefüggő egyenest tettem. Az 5.6.7 diagramon a színek és a vonalak jellege az említett kettő illusztrációval koherens.



5.6.6. ábra: Négy pontos hajlítás $\varepsilon_x(x, y)$ mezeje csomóponti átlagolással.

Az 5.6.5 ábrán, vagyis az átlagolás nélküli esetben élesen kirajzolódnak az elemhatárok, ami természetesen a nem túl sűrű hálóból is következik. A rácspontok osztásközét

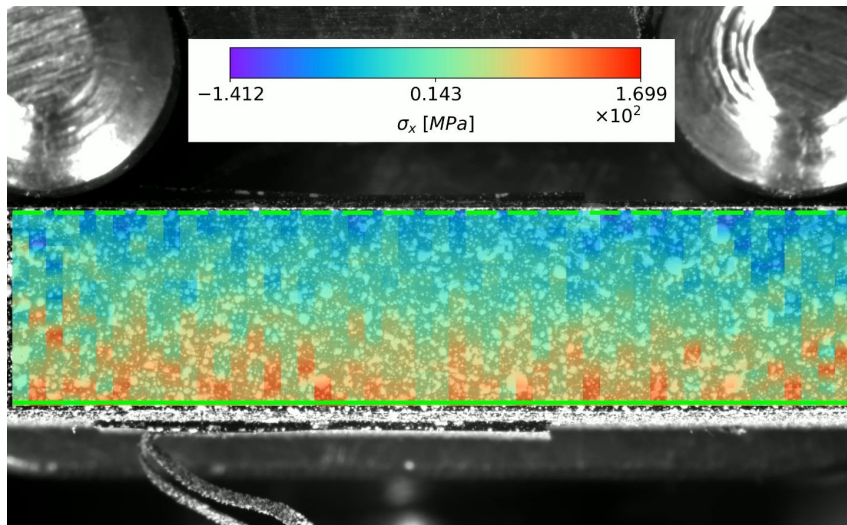
(50 [px] mindkét irányban) szándékosan úgy választottam meg, hogy az átlagolás hatása szabad szemmel is látható legyen. A könnyebb összehasonlítás érdekében a színskálák szélsőértékei azonosak a két felvételen.



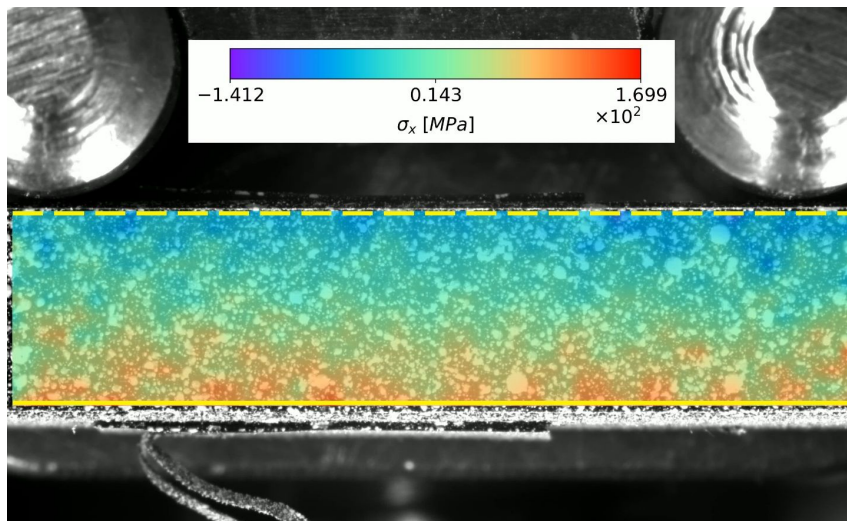
5.6.7. ábra: Az ε_x mező fix y magasságokon, átlagolással (sárga), és anélkül (zöld).

5.6.3. Csomóponti feszültségek átlagolása

A feszültségkomponensek esetében az átlagolás hatása hasonlóan jelentkezik, mint az alakváltozásoknál. A felvételek forrásából^[14] kiderül, hogy a próbatest alumíniumból készült, ezért a szükséges anyagparamétereket illetően $E = 70$ GPa rugalmassági moduluszt és $\nu = 0.32$ Poisson-tényezőt választottam. Ezen értékek csupán tájékoztató jellegű eredmények számítására lettek kiválasztva.

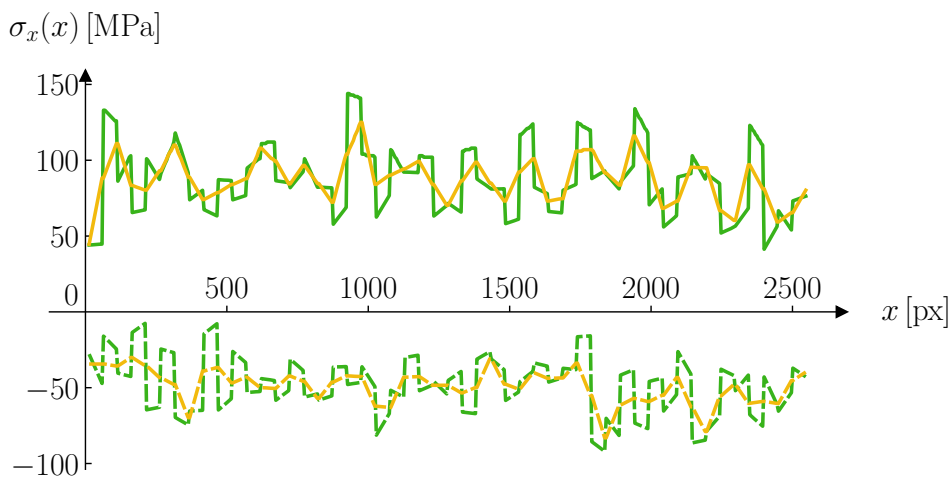


5.6.8. ábra: Négypontos hajlítás $\sigma_x(x, y)$ mezeje csomóponti átlagolás nélkül.



5.6.9. ábra: Négy pontos hajlítás $\sigma_x(x, y)$ mezeje csomóponti átlagolással.

Az 5.6.8 és 5.6.9 ábrákon szintén a videófelvétel utolsó képkockájára vonatkozó eredmények láthatók, azonos színskálával ellátva. A záró képkockához tartozó terhelési állapot az 5.5.1 ábra szerinti elrendezésben $F = 410 \text{ N}$. A $\sigma_x(x, y)$ mezőket bemutató ábrákon zölddel, illetve sárgával jelölt útvonalak mentén számított feszültségértékeket jeleníti meg az 5.6.10 ábra grafikonja. A diagramon felhasznált színek és vonalstílusok az 5.6.8 és 5.6.9 ábrákon lévővel koherensek.



5.6.10. ábra: A σ_x mező fix y magasságokon, átlagolással (sárga), és anélkül (zöld).

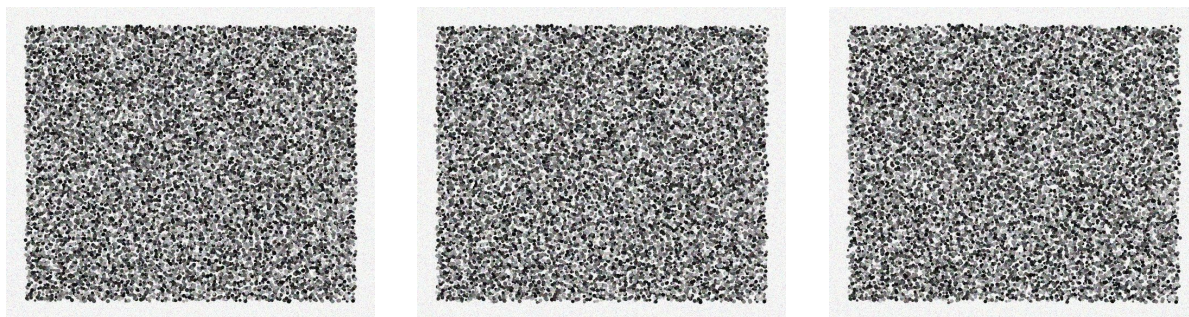
6. A szoftver tesztelése mintapéldákon keresztül

A projekt fő célja, hogy valós, általunk készített felvételeken végezhesünk vizsgálatokat, amelyek aztán segítenek a mérnöki problémamegoldásban. A program fejlesztése során a már említett négyponthoz hajlítás példájára támaszkodtam leginkább, azonban a dolgozatomban saját készítésű felvételeket is vizsgálok.

6.1. Virtuális tesztvideók használata

A programom fejlesztésében kitüntetett szerepet játszott egy sajátkezűleg készített, különálló kód, amely randomizált tesztvideók generálására képes. Az ilyen videók esetén a létrehozott elmozdulásmező bizonyos határokon belül véletlenszerű, ennek ellenére pontosan ismert minden képkockán. A virtuális tesztvideók generálását végző kódot elhelyeztem a függelékben.

Magát a tesztvideót alakíthatjuk úgy, hogy a saját próbatestünkön lévő pettyes mintázatot minél pontosabban reprezentálja. A program futtatása előtt testre szabható a pöttyök mérete, intenzitása, elhelyezkedése és még számos paraméter. A 6.1.1 ábrán látható, hogy a felvétel zajossága is modellezve van.

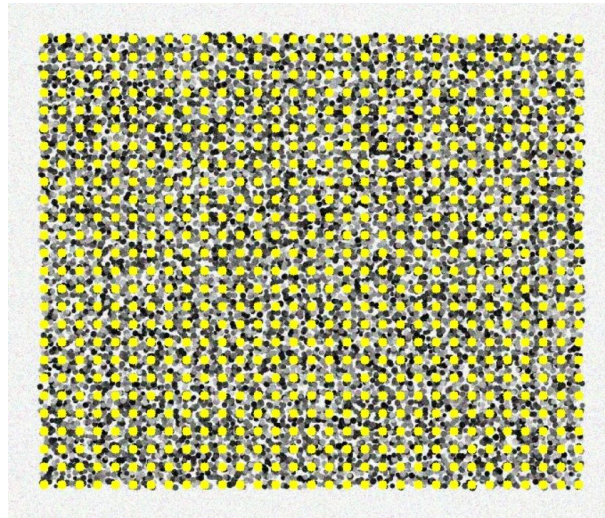


6.1.1. ábra: A tesztvideó első három képkockája.

Amint azt a 6.1.1 ábra mutatja, a virtuális videók elmozdulásmezeje nem kifejezetten látványos, nem szabályszerű, mindezt a véletlenszerűség jegyében alakítottam így. A pöttyök mozgása a gázokban és folyadékokban lévő részecskék *Brown-mozgásához* hasonlítható, emellett minden egyes generálás után eltérő.

A testreszabhatóságra visszatérve, még a virtuális videó elkészítése előtt szükséges beállítanunk azon pontok osztásközét, amelyek elmozdulásmezeje mentésre kerül a későbbi összehasonlítás céljából. Ez az érték jelen példában 20 [px] mindkét irányban, amelyhez a 6.1.2 ábrán szemléltetett rácsnak is igazodnia kell. Erre azért van szükség, mert a videón

szereplő összes pont elmozdulásának mentése megnehezítené a főprogram eredményeivel való összehasonlítást, ahol valamilyen praktikus választott osztásközzel dolgozunk.



6.1.2. ábra: A virtuális tesztvideóra illesztett rácspontok.

A különféle paraméterek beállítása után a tesztvideó legenerálása következik. Alapértelmezetten a videó felbontása 800×700 , azonban ezen is van lehetőség módosítani. A rövid kód futtatása során a beállított osztásközönként elhelyezkedő pontok időben változó elmozdulásmezeje mentésre kerül a `testdata_coordinates.dic` nevű fájlban. Ezután az elkészült videóra lefuttatjuk a `main.py` főprogramot, a teljes pöttyös terület kijelölésével, amelyre a korábban kiválasztott osztásközű hálót szükséges illeszteni a megfelelő összehasonlíthatóság érdekében. Az így létrejövő hálót mutatja a 6.1.2 ábra. A futtatást követően mentésre kerül a `testresults.dic` fájl, amely a mozgáskövető algoritmus által megtalált pontok koordinátáit tartalmazza.

A kettő említett fájl összevetésére egy külön kódrészlet szolgál, amely kiszámítja az ismert koordináták és a Lucas-Kanade algoritmus által nyújtott koordináták különbségét pixelben. Ezután tekintjük a k pixelnél nagyobb különbségek számát, tetszőleges k értékekre. Egy ilyen futtatás eredményét olvashatjuk a 6.1.3 ábrán. Az összehasonlítást végző rövid kódrészlet a dolgozat függelékében olvasható.

```
1      total points (nodes) tracked: 16120
2      difference above 3 px: 375
3      difference above 4 px: 55
4      difference above 5 px: 3
5      difference above 6 px: 0
6      difference above 7 px: 0
7      difference above 10 px: 0
```

6.1.3. ábra: Tesztvideó használata az optical flow paramétereinek finomhangolására.

Látható, hogy jelen példában az algoritmus legnagyobb hibája is ~ 5 [px]. Mivel a követett pontok osztásköze a kiindulási állapotban 20 [px] volt mindkét irányban, így feltételezhető, hogy nem volt olyan rácspont, amely koordinátájára egy szomszédos pont pozícióját adta volna eredményül a mozgáskövető algoritmus.

Összességében tehát a tesztvideó használata, illetve széleskörű konfigurálhatósága lehetőséget nyújt arra, hogy a mozgáskövető algoritmus által nyújtott eredményeket tüzetesen elemezzük. Ha kell, finomhangolhatjuk a főprogrambeli Lucas-Kanade módszer paramétereit, de akár a tesztvideó generálásán is módosíthatunk.

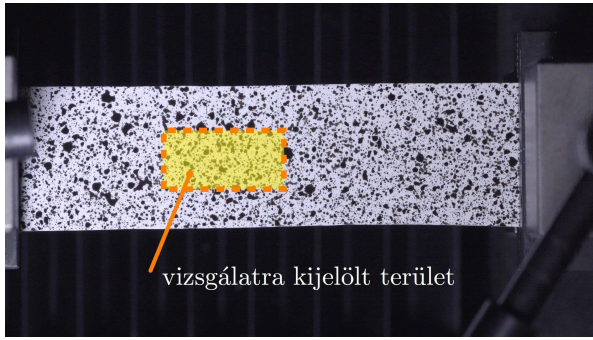
6.2. Tesztelés saját felvételekkel

A Tanszéki laborban készített videófelvételeken mikrocellás polietilén-tereftalát (MC-PET) mintadarabokat vetettünk egytengelyű húzás alá. A felvételek Fujifilm X-H2 típusú kamerával, illetve Fujifilm XF 16-80mm F/4 R OIS WR objektívvel készültek. Ezen felvételek közül most egyet vizsgálok részletesebben. A próbatestek mintázatai a 2. fejezetben megfogalmazott kívánalmak figyelembevételével készültek, festékszórással.

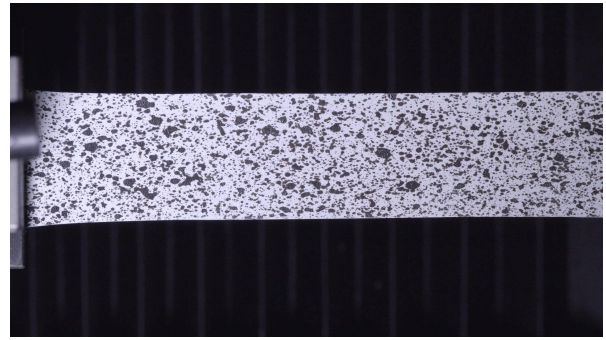
A Tanszéki laborban összeállított elrendezést mutatja a 6.2.1 ábra. A 6.2.2 ábrán látható a felvétel első képkockája, amelyen a vizsgálatra kiválasztott területet sárga téglalappal jelölöm. Az utolsó képkocka (6.2.3 ábra) a teljesen deformált alakot szemlélteti.



6.2.1. ábra: A saját mérés kivitelezése.

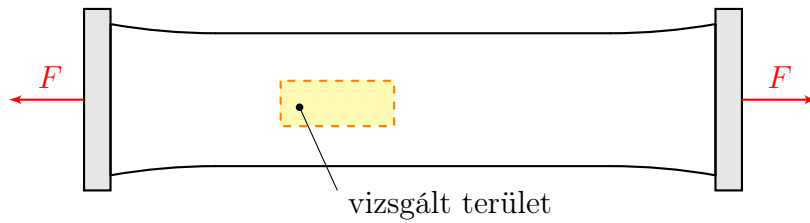


6.2.2. ábra: A felvétel első képkockája.



6.2.3. ábra: A felvétel utolsó képkockája.

A szoftveres vizsgálat sematikus összeállítását mutatja a 6.2.4 ábra. Az elemzett felvétel 8K, azaz 7680×4320 felbontású, illetve 30 FPS-sel készült. A továbbiakban bemutatott eredmények esetében a vizsgált terület következetesen 1520×720 méretű.



6.2.4. ábra: Az egytengelyű húzás mérési összeállítása, sematikus ábra.

A 6.2.4 ábrán sárga színnel jelölt terület a befogásoktól kellően távol esik, így azok hatása a Saint-Venant elv^[37] szerint csupán elenyésző mértékben érvényesül. Eszerint a kijelölt területen jó közelítéssel homogén eloszlású alakváltozási komponensekre számíthatunk. A vizsgálandó terület kiválasztásában, illetve a háló osztásközeinek meghatározásában nagy szerepet játszottak a pettyes mintázat adottságai.

A kijelölt területet néhány nagyobb (körülbelül $150 \text{ [px]} \times 150 \text{ [px]}$ befoglaló méretű) sötét folt határolja, amelyek a képp korreláció során gondot okoznak, amennyiben a megválasztott osztásközök ennél kisebbek. Ha ugyanis a mozgáskövetéshez használt pontrácson van például kettő szomszédos pont, amelyek környezete nagyon hasonló az intenzitásértékek szempontjából (vagyis ugyanazon sötét foltban belülré esnek), akkor az optical flow algoritmus könnyen fals eredményt szolgáltat már az elmozdulások szintjén. A háló ritkításával, vagyis az osztásközök növelésével ez a probléma elkerülhető, azonban én egy finomabb hálót kívántam alkalmazni.

A választott osztásközök értékének alsó korlátját a $\sim 80 \text{ [px]}$ szélességű (avagy magasságú) egybefüggő világos területek szabták meg, az előzőleg ismertetett jelenség okán. Így a háló osztásközeire mindkét irányban 80 pixelt választottam, amely 200 csomópont-

tot és 171 elemet eredményezett. Összefoglalva, a létrehozott mintázat legkedvezőtlenebb tulajdonsága a 2.0.1d ábrán illusztrált, túlságosan eltérő méretű pöttyök jelenléte.

6.2.1. A program futtatásához kapcsolódó beállítások

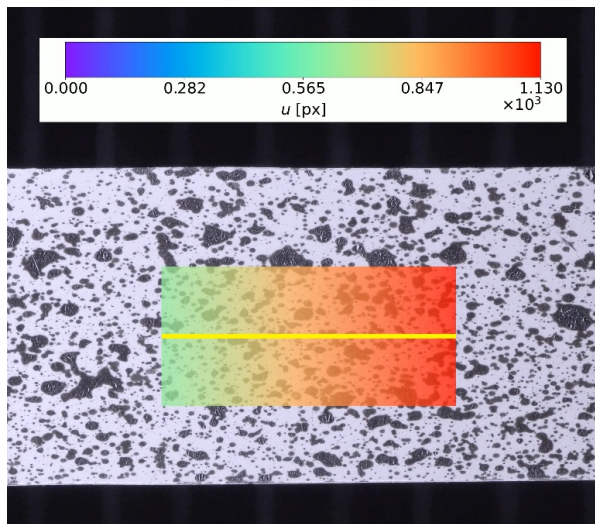
A pettyes mintázatot szem előtt tartva immár meghatároztuk a vizsgált területet és a háló osztásközeit, azonban ezeken túlmenően van még néhány paramétere a főprogramnak, amelyet szükséges módosítanunk a futtatás előtt. A 30 FPS-es felvétel áttekintése után úgy találtam, hogy elegendő csupán minden tizenötödik képkockát figyelembe venni képkorreláció szempontjából, ezzel párhuzamosan azonban érdemes a 2.3.1 táblázatban olvasható `winSize = (100,100)` paramétert beállítani, amellyel garantáljuk, hogy a követett pontok nem távoznak a keresési ablakból. Ezt a beállítást már csupán a felvétel extrém magas felbontása is indokolja. Az említett táblázatban szereplő `criteria.epsilon` paraméteren szigorítottam a korábbiakhoz képest, a pontosság érdekében immár `criteria.epsilon = 0.02`.

Az esztétikus megjelenítés szempontjából célszerű módosítani a colorbar méretén, helyzetén és a betűméretén is. Jelen vizsgálat során nem használom a szoftvert feszültség-számításra, vagyis `Stress_Calc = False`, hiszen a terhelés alatt képlékeny alakváltozást szenved a próbatest, ráadásul az anyagi viselkedés viszkoelasztikus és viszkoplasztikus jellemzőket is mutat. Ezen jelenségek külön-külön is kizárják a Hooke-törvény alkalmazásának lehetőségét.

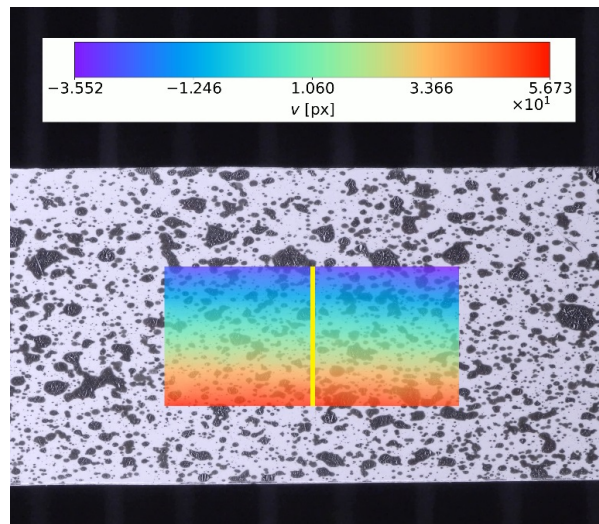
6.2.2. Az egytengelyű húzás elmozdulásmezeje

Az elmozdulásmező u és v komponensére kapott megoldásokat az egyes komponensekre vonatkozó színskálák használatával ábrázolom, amint azt a felmerülő nagyságrendek indokolják. A 6.2.5a és 6.2.5b ábrákon jelölt útvonalak mentén számított értékeket külön diagramokon is megjelenítem (6.2.5c és 6.2.5d) a könnyebb kiértékelés érdekében. Fontos megjegyezni, hogy mind most, mind pedig később, az alakváltozások tárgyalásánál a bemeneti videófelvétel utolsó képkockájára vonatkozó eredményeket ismertetem.

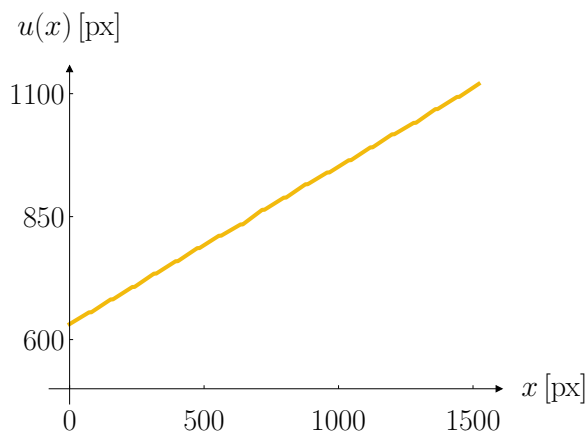
A 6.2.5 ábrán összességében jól látszik, hogy az $u(x, y)$ mező fix y magasságon lineáris jelleget mutat, hogyha pedig az x koordinátát rögzítenénk, akkor egy függőleges útvonal mentén nagyjából konstans $u(y)$ értékeket kapnánk. Hasonlóan, a $v(x, y)$ mező egy tetszőleges vertikális útvonal mentén jó közelítéssel lineáris összefüggést mutat, míg egy fix y magasságon hozzávetőlegesen konstans $v(x)$ eredmény születne.



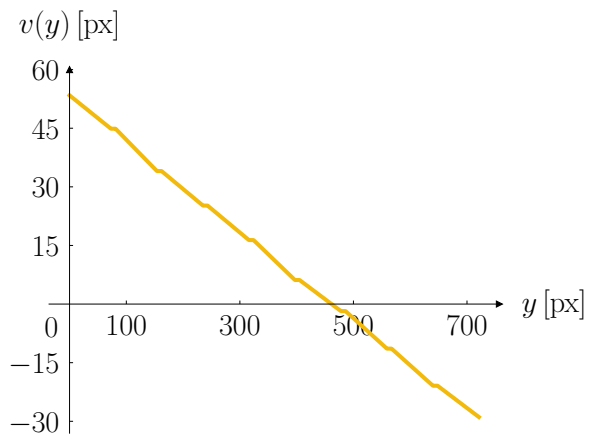
(a) Az $u(x, y)$ komponens.



(b) A $v(x, y)$ komponens.



(c) A fentebbi útvonal menti $u(x)$ eredmények.



(d) A fentebbi útvonal menti $v(y)$ eredmények.

6.2.5. ábra: Az egytengelyű húzóvizsgálat elmozdulásmezeje.

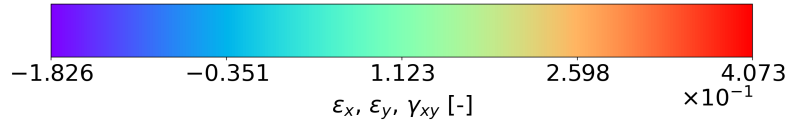
A 6.2.5 ábrán alapuló megfigyeléseink szerint, a kijelölt útvonalak mentén

$$\varepsilon_x = \frac{\partial u}{\partial x} \approx \text{konst.} \quad \text{és} \quad \varepsilon_y = \frac{\partial v}{\partial y} \approx \text{konst.}, \quad (6.2.1)$$

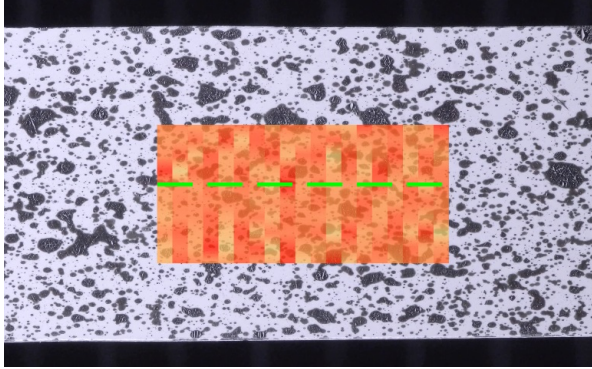
vagyis a nagyjából homogén eloszlású alakváltozás-komponensekre vonatkozó várakozásaink beteljesülésére jó esély mutatkozik.

6.2.3. Az egytengelyű húzás alakváltozási mezeje

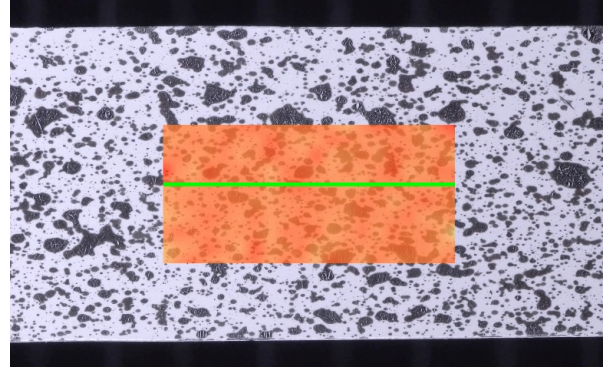
Az alakváltozások számítását csomóponti átlagolással, illetve anélkül is elvégeztem. Az utolsó képkockára vonatkozó eredményeket a 6.2.6 ábra foglalja össze. Mivel a különböző alakváltozás-komponensek nagyságrendileg összemérhetők, így a 6.2.6a ábrán látható egységes színskálát alkalmazom az összes mezőváltozóra vonatkozóan.



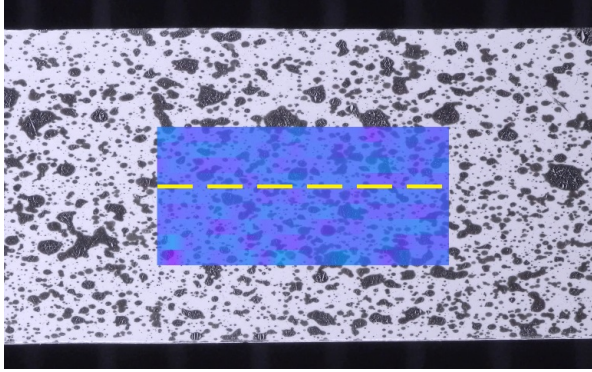
(a) Az összes alakváltozási komponensre érvényes színskála.



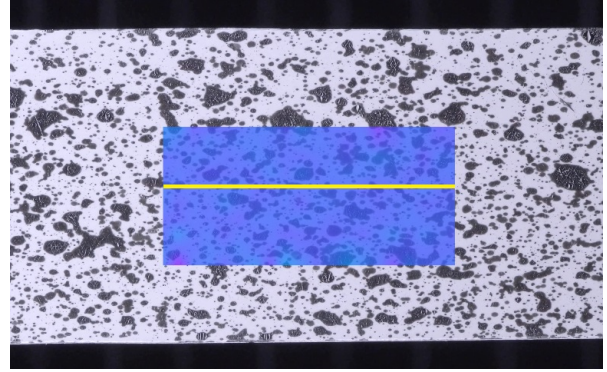
(b) Az $\varepsilon_x(x, y)$ komponens.



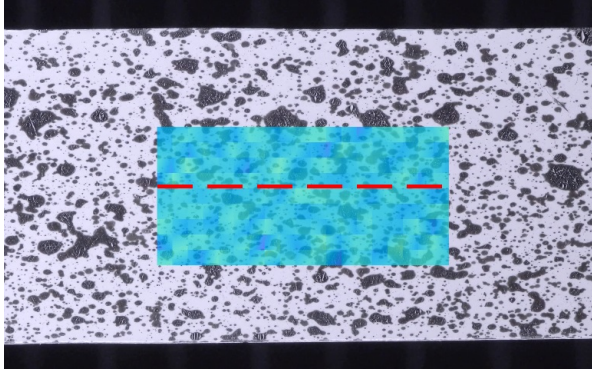
(c) Az $\varepsilon_x(x, y)$ komponens, átlagolással.



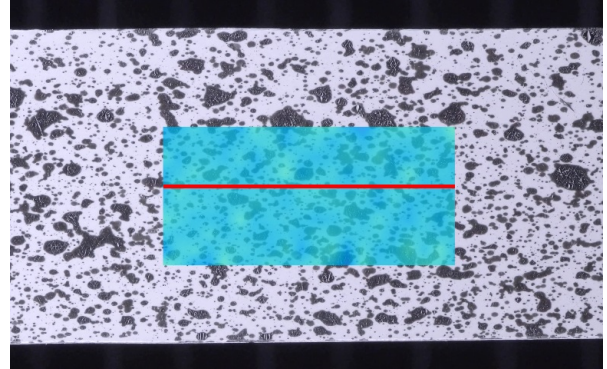
(d) Az $\varepsilon_y(x, y)$ komponens.



(e) Az $\varepsilon_y(x, y)$ komponens, átlagolással.



(f) A $\gamma_{xy}(x, y)$ komponens.

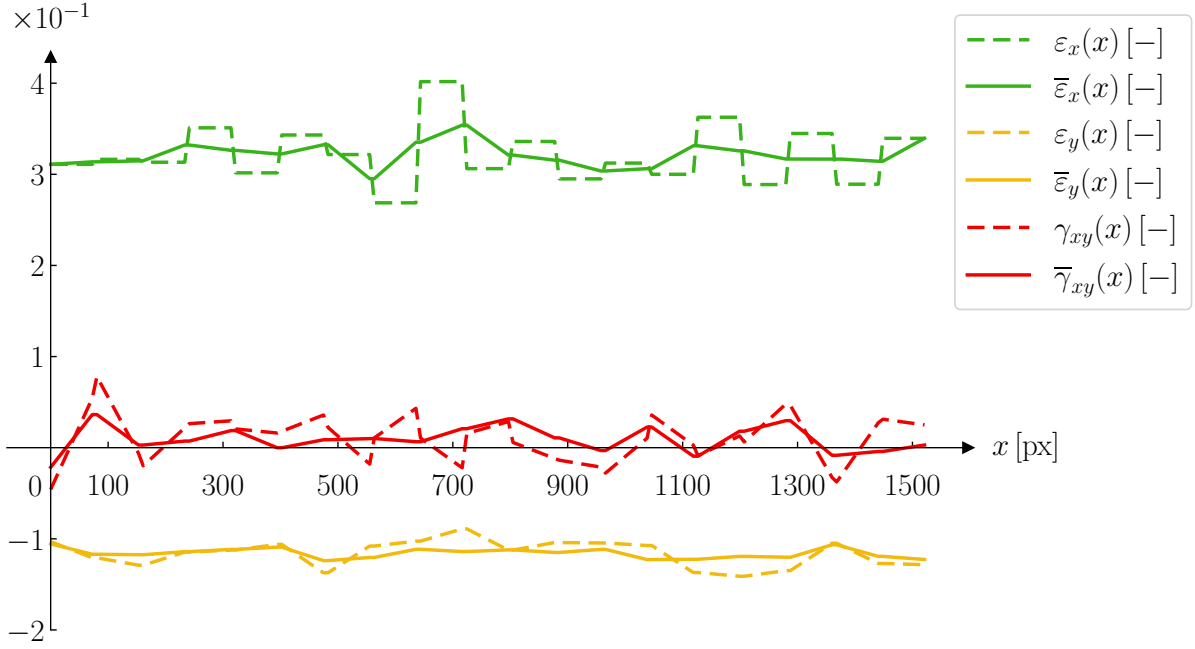


(g) A $\gamma_{xy}(x, y)$ komponens, átlagolással.

6.2.6. ábra: Az egytengelyű húzóvizsgálat alakváltozási mezeje.

A mezőváltozók illusztrációin ismét kijelöltem egy útvonalat, amely mentén a számítások eredményeit egy közös diagramon foglalom össze. A 6.2.6 ábrán látható egyenesek színei az alakváltozás-komponensek megkülönböztetését segítik a 6.2.7 grafikonon. Továbbá, folytonos vonal jelöli a csomóponti átlagolással kapott eredmény sorozatot, az átlagolás nélküli értékek pedig szaggatott vonallal szerepelnek, mind az eredmények ábráin, mind

pedig a diagramon.



6.2.7. ábra: Alakváltozás-értékek a kijelölt útvonal mentén.

A mezőváltozók ábráiról és a 6.2.7 grafikonról könnyen leolvasható, hogy az alakváltozási komponensek eredményei jól közelítik a kezdeti várakozásainkban megfogalmazott homogenitást. A kisebb-nagyobb mértékű eltérések több forrásból erednek. Egyrészt az anyagi inhomogenitást nem szabad elhanyagolnunk, másrészt pedig a pettyes mintázat sajátosságaiból fakadóan néhány helyen kissé kiugró eredmények születhetnek. Ez a jelenség a digitális képkorrelációs módszer legérzékenyebb aspektusa.

Az egytengelyű húzóvizsgálat segítségével egyszerűen meghatározható a próbatest Poisson-tényezője, amely definíció szerint kapcsolatot teremt a keresztirányú és a hosszirányú alakváltozások között:

$$\nu = \frac{-\varepsilon_{\text{kereszt}}}{\varepsilon_{\text{hossz}}} = \frac{-\varepsilon_y}{\varepsilon_x}, \quad (6.2.2)$$

ezáltal megmutatja, hogy egységnyi ε_x hatására mekkora ε_y keresztirányú összehúzódás történik. Egyes anyagok esetén előfordul, hogy $\nu < 0$, ekkor az anyag húzás hatására keresztirányban is tágul. Általánosságban a Poisson-tényező értéke -1 és 0.5 közé eshet^[33].

A kísérletem során mind az $\varepsilon_x(x, y)$, mind az $\varepsilon_y(x, y)$ mező esetén az utolsó képkockán számított értékek számtani közepét vettem alapul ν meghatározásához. Csomóponti átlagolással és anélkül is, az MC-PET mintadarab Poisson-tényezőjére $\nu = 0.3584$ adódott.

7. Összegzés, konklúzió

A munkám során egy olyan szoftvert készítettem, amely síkbeli deformációk mozgáselemzését hajtja végre nagyfelbontású videófelvevételek alkalmazásával. Fontos szempont volt a futási idő minimalizálása, amelyben nagy szerepet játszanak az ingyenes, teljesítményre optimalizált Python könyvtárak, mint például az OpenCV^[38] és a NumPy^[39].

A vizsgálat alapjául a digitális képkorrelációs módszer szolgál, amelyet a Lucas-Kanade optical flow algoritmusra^[21] építettem fel. Ezzel a módszerrel meghatározhatók az elmozdulásértékek egy általunk kijelölt, téglalap alakú területen elhelyezkedő ponthalmaz egyes pontjain. Ezen diszkrét pontokban alakváltozás-értékek számíthatók a programmal, kis alakváltozásokat feltételezve. Lineárisan rugalmas, homogén, és izotrop anyagi viselkedés esetén a Hooke-törvény segítségével opcionálisan feszültségek is számíthatók. A diszkrét csomóponti értékek interpolációja során a lineáris négycsomópontos négyszögelem elméleti hátterére támaszkodik a program. Folytonos alakváltozási-, illetve feszültségmezők létrehozására ad lehetőséget a kapcsolódó csomóponti értékek átlagolása.

A program fő eredménye a számított mezőváltozók bemeneti videófelvevételekre való vizuális ráillesztése. A megoldások helyességét részeredményeket tartalmazó szövegfájlokkal, míg a mozgáskövető algoritmus pontosságát pedig virtuális tesztvideókon végzett próba-futtatásokkal lehet ellenőrizni.

Összefoglalva, a jelen projekt során egy könnyen kezelhető és gyors program készült el optikai alapú mozgáselemzésre, amely nemcsak a Tanszéken folyó kutatások kapcsán, hanem a Tanszék falain kívül is hasznosnak bizonyulhat. A szoftver a hagyományos módszereknél (például felragasztott nyúlásmérő bélyegeknél) sokkal bővebb és kiterjedtebb adathalmazt szolgáltat. A dolgozatomban szereplő mozgáselemzés végrehajtásához azonban gondosabb előkészületek szükségesek. Elengedhetetlen lépés a próbatest megfelelő mintázattal való ellátása, de szintén fontos a megfelelő optikai eszközpark biztosítása is.

7.1. Korlátok, továbbfejlesztési lehetőségek

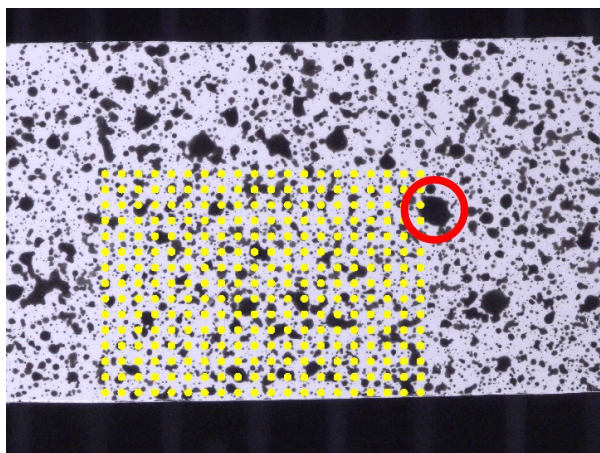
A program természetesen nem tekinthető befejezettnek, sok területen merülnek fel továbbfejlesztési lehetőségek. Fejlettebb DIC szoftverek esetén például a vizsgált terület nem kizárólag téglalap alakú, a lekövetett ponthalmaz tetszőleges alakot vehet fel. A felhasználóban felmerülhet az igény másfajta, a négycsomópontos lineáris négyszögelemtől eltérő végelemek használatára. Más végelemtípusok implementálásával az interpo-

lációs módszerek tárháza bővülne ki. Említettem, hogy a szoftver kis alakváltozások számítását képes elvégezni, e terület fejlesztése a jövőben kulcsfontosságú lehet. Távlati célként merül fel a jelenleg síkbeli mozgásokat kezelő szoftver 3D-s mozgáselemzésre való felkészítése is. A tudományos életben gyakoriak az ilyen irányú törekvések is^[40,41].

7.2. A nagy alakváltozások kezelésének első lépései

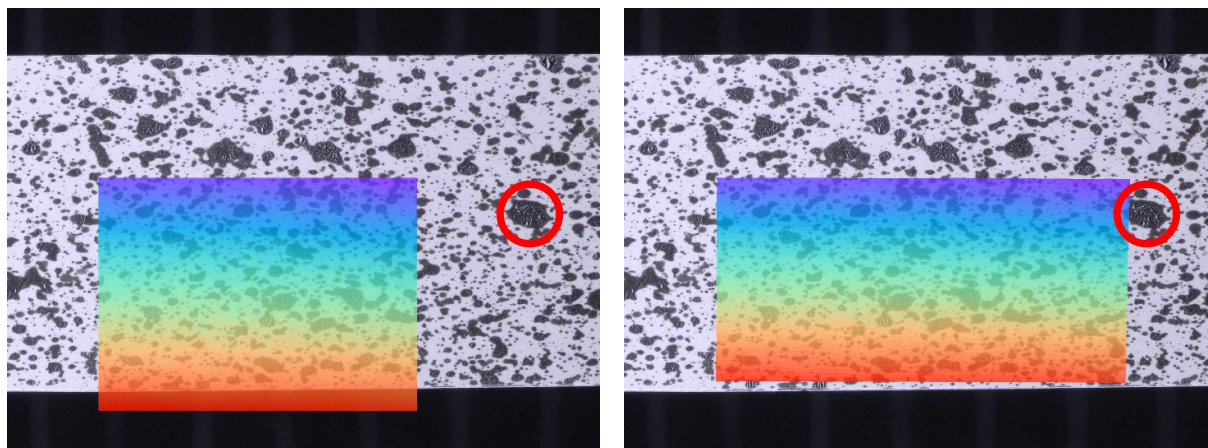
A programban már megvalósítottam néhány előkészítő lépést a *nagy* alakváltozások kezelésének érdekében. Először is, a 4. fejezetben feltételeztük, majd az 5. fejezetben a mezőváltozók ábrázolásánál kihasználtuk, hogy a kis alakváltozásokból adódóan az összes végelem minden egyes képkockán egy a, b oldalhosszúságokkal rendelkező téglalap, amint azt a 4.3.1 ábra mutatja. Nagy alakváltozások esetén ez a feltételezés nem tartható, a globális x, y koordináták interpolációjára is szükség van. Az 5.1.1 ábrán látható interpolációs rács felhasználásával, a ξ és η értékek függvényében a globális koordináták is interpolálhatók a koordináta-transzformáció (4.2.1 egyenlet) segítségével. A csomóponti értékeket a mozgáskövető algoritmus szolgáltatja.

A globális x, y koordináták ismeretében a különféle mezőváltozók ábrázolásánál figyelembe tudjuk venni az egyes elemek, így pedig a teljes vizsgált terület időfüggő deformálódását is. A 7.2.1 ábrán, az egytengelyű húzóvizsgálat videófelvételén szándékosan egy olyan területet jelöltem ki, amelynek alsó határa a próbatest széléhez közel húzódik. Az eredeti feltevés szerint a mezőváltozók ábrái a mozgás során végig a kezdetben kijelölt téglalap méreteit veszik fel. Ez a jelenség az utolsó képkockán már *vizuálisan* kifogásolható eredményt ad a próbatest keresztirányú zsugorodása, illetve a nagymértékű x irányú nyúlása miatt. Ugyanez a helyzet a 6.2.6 ábrán összefoglalt eredmények esetében is. Ennek ellenére a számított numerikus eredmények helyességét ez a jelenség nem befolyásolja.



7.2.1. ábra: A követett pontok halmaza, a mintadarab széléhez közel.

A 7.2.1 és a 7.2.2a ábrák összevetésében látható, hogy a felvétel elején a vizsgált terület határán elhelyezkedő foltok (például a pirossal bekarikázott is) a videó végére nagymértékben eltávolodtak az adott mezőváltozó illusztrációjának határától. Mindemellett az ábrázolás olyan területre is kiterjed, ahol már (a mozgás végére) nincs anyag.



(a) Konstans téglalapokkal ábrázolva.

(b) Az x, y koordináták interpolációjával.

7.2.2. ábra: Az egytengelyű húzóvizsgálat $v(x, y)$ mezeje a felvétel utolsó képkockáján.

A 7.2.2 ábrán kirajzolt mezőváltozó, annak felvett értékei, illetve a felhasznált háló adatai az ábrázolási szempontokat tekintve jelenleg nem relevánsak. A főprogramban az `assume_rectangles = False` parancs meghívásával figyelembe vehetjük a kijelölt terület deformálódását. Ezt hivatott illusztrálni a 7.2.2b ábra. Mindez azonban a *nagy* alakváltozások kezelésének csupán az első lépése volt. Nagyobb alakváltozások esetén ugyanis a 4.3.1 egyenlet nem érvényes, az alakváltozási tenzor formulációjára más elméletet kell alkalmaznunk. E téma tárgyalására jelen dolgozat nem tér ki.

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani a konzulensemnek, Dr. Kossa Attila egyetemi docensnek a dolgozatom megírásában nyújtott értékes szakmai támogatásáért. Hálával tartozom továbbá a Műszaki Mechanikai Tanszéknek a saját videófelvételek elkészítéséhez szükséges környezet és eszközök biztosításáért.

Jelen kutatás a Kulturális és Innovációs Minisztérium Nemzeti Kutatási, Fejlesztési és Innovációs Alapból nyújtott támogatásával valósult meg. Pályázati azonosító: FK 142457. Jelen TDK dolgozatban bemutatott tudományos eredmények az MTA Bolyai János Kutatási Ösztöndíj támogatásával készültek.

Hivatkozások

- [1] M. Kalaitzakis and N. Vitzilaios et al. Drone-based stereodic: System development, experimental validation and infrastructure application. *Experimental Mechanics*, 61: 981–996, 2021.
- [2] G. Lavezzi and M. Ciarcià et al. A dic-uav based displacement measurement technique for bridge field testing. *Engineering Structures*, 308:117951, 2024. ISSN 0141-0296. doi: <https://doi.org/10.1016/j.engstruct.2024.117951>. URL <https://www.sciencedirect.com/science/article/pii/S0141029624005133>.
- [3] V. Chandra and P. Chakraborty. Automated crack extension measurement method for fracture and fatigue analysis using digital image correlation. *Engineering Fracture Mechanics*, 305:110182, 2024. ISSN 0013-7944. doi: <https://doi.org/10.1016/j.engfracmech.2024.110182>. URL <https://www.sciencedirect.com/science/article/pii/S001379442400345X>.
- [4] J. Asswad and E. Ziade et al. Assessment of resistance to crack propagation of asphalt mixtures using two parts digital image correlation. *Theoretical and Applied Fracture Mechanics*, 133:104578, 2024. ISSN 0167-8442. doi: <https://doi.org/10.1016/j.tafmec.2024.104578>. URL <https://www.sciencedirect.com/science/article/pii/S0167844224003288>.
- [5] C. L. von Boyneburgk and J.-C. Zarges et al. Mechanical characterization and simulation of wood textile composites (wtc) supported by digital image correlation (dic). *Composites Part C: Open Access*, 11:100370, 2023. ISSN 2666-6820. doi: <https://doi.org/10.1016/j.jcomc.2023.100370>. URL <https://www.sciencedirect.com/science/article/pii/S2666682023000269>.
- [6] M. Mehrabian and R. Boukhili. 3d-dic strain field measurements in bolted and hybrid bolted-bonded joints of woven carbon-epoxy composites. *Composites Part B: Engineering*, 218:108875, 2021. ISSN 1359-8368. doi: <https://doi.org/10.1016/j.compositesb.2021.108875>. URL <https://www.sciencedirect.com/science/article/pii/S1359836821002663>.
- [7] Michael A Sutton, Jean Jose Orteu, and Hubert Schreier. *Image correlation for shape, motion and deformation measurements: basic concepts, theory and applications*. Springer Science & Business Media, 2009.
- [8] L. Wang and Z. Lei. Deep learning based speckle image super-resolution for digital image correlation measurement. *Optics & Laser Technology*, 181:111746, 2025. ISSN 0030-3992. doi: <https://doi.org/10.1016/j.optlastec.2024.111746>. URL <https://www.sciencedirect.com/science/article/pii/S0030399224012040>.
- [9] S. Boukhtache and K. Abdelouahab et al. A lightweight convolutional neural network as an alternative to dic to measure in-plane displacement fields. *Optics and Lasers in Engineering*, 161:107367, 2023. ISSN 0143-8166. doi: <https://doi.org/10.1016/j.optlaseng.2022.107367>. URL <https://www.sciencedirect.com/science/article/pii/S0143816622004183>.

- [10] X. Duan and H. Xu et al. Digital image correlation based on convolutional neural networks. *Optics and Lasers in Engineering*, 160:107234, 2023. ISSN 0143-8166. doi: <https://doi.org/10.1016/j.optlaseng.2022.107234>. URL <https://www.sciencedirect.com/science/article/pii/S0143816622002871>.
- [11] Y. Zhou and Q. Zuo et al. Transformer based deep learning for digital image correlation. *Optics and Lasers in Engineering*, 184:108568, 2025. ISSN 0143-8166. doi: <https://doi.org/10.1016/j.optlaseng.2024.108568>. URL <https://www.sciencedirect.com/science/article/pii/S0143816624005463>.
- [12] Advanced Industry And Education Equipment Co., Ltd (AIE). Gom correlate 2, 2024. URL <https://aie.com.vn/en/gom-correlate-2/>. Hozzáféres dátuma: 2024. 11. 01.
- [13] X-Sight s.r.o. X-sight 2d dic system, 2024. URL <https://www.xsight.eu/digital-image-correlation-optical-measuring-system-xsight-2d-dic/>. Hozzáféres dátuma: 2024. 11. 01.
- [14] Damien André. pydic. <https://gitlab.com/damien.andre/pydic>, 2024. Hozzáféres dátuma: 2024. 11. 01.
- [15] PolymerGuy. pdic: A toolkit for digital image correlation. <https://github.com/PolymerGuy/muDIC/tree/master>, 2024. Hozzáféres dátuma: 2024. 11. 01.
- [16] Correlated Solutions Inc. World-class 2d digital image correlation, 2024. URL <https://www.correlatedsolutions.com/vic-2d>. Hozzáféres dátuma: 2024. 11. 01.
- [17] Zhenyu Jiang. Opencorr: An open source library for research and development of digital image correlation. *Optics and Lasers in Engineering*, 165:107566, 2023. ISSN 0143-8166. doi: <https://doi.org/10.1016/j.optlaseng.2023.107566>. URL <https://www.sciencedirect.com/science/article/pii/S0143816623000957>.
- [18] N. Lovaas. Speckle pattern fundamentals. Technical report, Correlated Solutions Inc., 2020. URL <https://correlated.kayako.com/article/38-speckle-pattern-fundamentals>. Hozzáféres dátuma: 2024. 11. 01.
- [19] OpenCV Python Tutorials. Optical flow, 2016. URL https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html. Hozzáféres dátuma: 2024. 11. 01.
- [20] William LePage. A practical guide to dic., 2024. URL <https://digitalimagecorrelation.org/>. Hozzáféres dátuma: 2024. 11. 01.
- [21] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI'81: 7th international joint conference on Artificial intelligence*, volume 2, pages 674–679, 1981.
- [22] Jianbo Shi et al. Good features to track. In *1994 Proceedings of IEEE conference on computer vision and pattern recognition*, pages 593–600. IEEE, 1994.
- [23] Jean-Yves Bouguet et al. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel corporation*, 5(1-10):4, 2001.

- [24] M. Hofinger and S. R. Bulò et al. Improving optical flow on a pyramid level. In *European Conference on Computer Vision*, pages 770–786. Springer, 2020.
- [25] OpenCV Documentation 4.10.0. Object tracking, 2024. URL https://docs.opencv.org/4.10.0/dc/d6b/group__video__track.html. Hozzáférés dátuma: 2024. 11. 01.
- [26] Gábor Dr. Csernák. *Dinamika*. Akadémiai Kiadó, 2018.
- [27] Olga Sorkine-Hornung and Michael Rabinovich. Least-squares rigid motion using svd. *Computing*, 1(1):1–5, 2017.
- [28] Pál Rózsa. *Lineáris algebra és alkalmazásai*. Műszaki Könyvkiadó, 1974.
- [29] H. Gong and S. Cheng et al. An efficient digital twin based on machine learning svd autoencoder and generalised latent assimilation for nuclear reactor physics. *Annals of Nuclear Energy*, 179:109431, 2022. ISSN 0306-4549. doi: <https://doi.org/10.1016/j.anucene.2022.109431>. URL <https://www.sciencedirect.com/science/article/pii/S0306454922004613>.
- [30] M. Kumar and A. Vaish. An efficient encryption-then-compression technique for encrypted images using svd. *Digital Signal Processing*, 60:81–89, 2017. ISSN 1051-2004. doi: <https://doi.org/10.1016/j.dsp.2016.08.011>. URL <https://www.sciencedirect.com/science/article/pii/S1051200416301348>.
- [31] A. Bouhamidi and K. Jbilou et al. An extrapolated tsvd method for linear discrete ill-posed problems with kronecker structure. *Linear Algebra and its Applications*, 434(7):1677–1688, 2011. ISSN 0024-3795. doi: <https://doi.org/10.1016/j.laa.2010.06.001>. URL <https://www.sciencedirect.com/science/article/pii/S0024379510002971>. Special Issue: NIU.
- [32] OpenCV Documentation 4.10.0. Videocapture class reference, 2024. URL https://docs.opencv.org/4.10.0/d8/dfe/classcv_1_1VideoCapture.html. Hozzáférés dátuma: 2024. 11. 01.
- [33] Gábor Dr. Csernák. *Szilárdságtan*. Akadémiai Kiadó, 2021.
- [34] R. v. Mises. Mechanik der festen körper im plastisch- deformablen zustand. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, 1913:582–592, 1913. URL <http://eudml.org/doc/58894>.
- [35] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.
- [36] OpenCV Documentation 4.10.0. Creating a video with opencv, 2024. URL https://docs.opencv.org/4.10.0/d7/d9e/tutorial_video_write.html#autotoc_md228. Hozzáférés dátuma: 2024. 11. 01.
- [37] A.J.C.B. de Saint-Venant. *Memoire sur la torsion des prismes: avec des considerations sur leur flexion ainsi que sur l'équilibre intérieur des solides élastiques en général, et des formules pratiques pour le calcul de leur résistance à divers efforts s'exerçant simultanément*. S.l., 1853. URL <https://books.google.hu/books?id=wnjyxgEACAAJ>.

- [38] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.
- [39] C. R. Harris and K. J. Millman et al. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL <https://doi.org/10.1038/s41586-020-2649-2>.
- [40] X. Dai and J. Qi et al. Omnidirectional 3d-dic method for determining inner deformation in pipelines. *Measurement*, 216:112924, 2023. ISSN 0263-2241. doi: <https://doi.org/10.1016/j.measurement.2023.112924>. URL <https://www.sciencedirect.com/science/article/pii/S0263224123004888>.
- [41] Y. Huang and Z. Hui et al. Deformation ability of steel inner sleeve t-joint in modular gymnasium based on 3d-dic method. *Journal of Building Engineering*, 94:110002, 2024. ISSN 2352-7102. doi: <https://doi.org/10.1016/j.jobe.2024.110002>. URL <https://www.sciencedirect.com/science/article/pii/S2352710224015705>.