



TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

Rácsos szerkezetek topológiai optimálásának automatizálása:

Többszintű párhuzamos szűrőalapú szerkezet-detektáló algoritmus

Daróczy László

I. éves gépészmérnök (MSc) hallgató

Konzulens:

Dr. Jármái Károly (DSc) – egyetemi tanár
Anyagmozgatási és Logisztikai Tanszék

Miskolc, 2011

Tartalomjegyzék

1. Bevezetés	3
2. A tervezési folyamat új megközelítése	3
3. 2. lépés – BPPF	8
4. 3. lépés – Poszt-processzálas	15
5. BPPF implementálás 3D-s szerkezetekre	23
6. Összefoglalás	36
7. Köszönetnyilvánítás	37
8. Irodalomjegyzék	38
Függelék.....	39

1. Bevezetés

A számítógépek fejlődésével, a rendelkezésre álló számítási kapacitásoknak és a növekvő teljesítményeknek hála, valamint a véges-elem modellek (a továbbiakban VEM) fejlődésének és elterjedésének hála a rácsos tartók tervezése topológiai optimalás segítségével (a továbbiakban *TO – Topology Optimization*) egy megvalósítható tervezési alternatívává vált. A manapság elterjedt topológiai optimaló szoftverek képesek egyszerre több peremfeltétellel és akár több célfüggvényre vonatkozó megszorítással dolgozni egyszerre [12,13]. Ennek ellenére a topológiai optimalás eredményei nem használhatóak fel rögtön közvetlenül, hanem csupán egy kezdőpontként fog szolgálni a tervezést végző mérnök számára, mivel a topológiai optimalás végeredménye pl. nem I-tartókból van összeszerelve, és a rudak sem feltétlenül állandó keresztmetszetűek, illetve a csomópontok típusára sem lehet megkötéseket megadni. Bár több módszer is létezik optimális rácsos szerkezetek létrehozására, de a legtöbb ezek közül kötött számú rudat alkalmaz, amelyek csak előre meghatározott módon kapcsolódhatnak egymáshoz. [1,2].



1. ábra

Rácsos szerkezetű híd

Creative Commons, C. Hanchey ©

Forrás: <http://www.flickr.com/photos/21953562@N07/4529563149/>

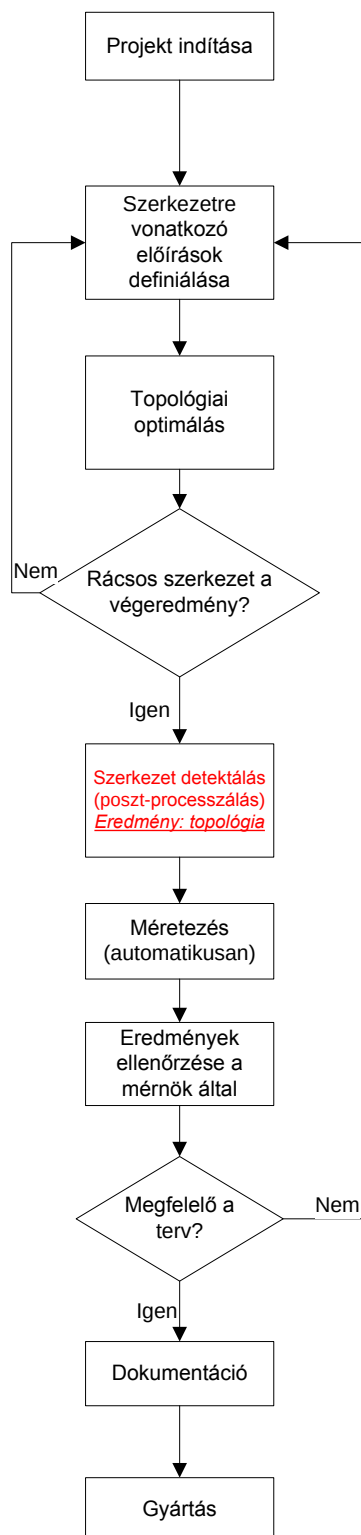
2. A tervezési folyamat új megközelítése

Az új megközelítés alapötlete az, hogy a topológiai optimalás eredményét kizárólag az optimális topológia meghatározásához használjuk (azaz a rudak és csomópontok száma, és a közöttük lévő kapcsolatok sorrendje). A program automatikusan detektálja a rácsos szerkezetet, és a végeredményt egy méretezési feladathoz használja fel bemenetként. Azzal a feltételezéssel élve, hogy a topológia maga optimális, a méretezés célja a csomópontok pontos helyzetének és a rudak keresztmetszetének meghatározására egyszerűsödik. Mivel a méretezési feladat során a változók száma jelentősen kisebb, mint a topológiai optimalás folyamán, ezért ez gyorsan és pontosan végrehajtható.

A tervezési folyamat javasolt folyamatábrája a **2. ábrán** található.

Bár több középvonala detektáló algoritmus is létezik, de ezek egyike sem specifikusan a topológiai optimalási feladatokhoz lett kifejlesztve, és a legtöbb messze van a 100%-os megbízhatóságtól. Az alábbi detektáló algoritmus a bitminta-primitív szűrők használatán alapul (a továbbiakban *BPPF – Bitpattern-primitive filters*, a későbbiekben definiálva). A szerkezet topológiáját (azaz a csomópontok

megközelítő helyzetét, és az azokat összekötő rudakat) a BPPF fogja visszaadni, ami méretoptimalás bemeneti adatait fogja szolgáltatni egyben.



2. ábra
Rácsos szerkezetek tervezésének
javasolt folyamata

A BPPF azon az ötleten alapul – mivel csak a topológiára van szükségünk –, hogy nem szükséges a számítási teljesítményt a pozíciók pontos meghatározására, vagy a rudak alakjának detektálására elpazarolni, mivel azt a cég erőforrásai és egyéb mérnöki szempontok fogják meghatározni (pl. I-tartó), ezért a csomópontok helyzete és a rudak vastagsága bemeneti adatként fog szolgálni inkább egy méretoptimalásnak. Ez utóbbi terület egy széles körben elterjedt eljárás a mérnöki tervezésben [3,4,5]. Az eredeti ötlet ellenére bemutatásra kerül az alábbi cikkben, hogy egy megfelelő BPPF és poszt-processzálas alkalmazásával akár a pontos pozíciók és pl. keresztmetszetek is meghatározhatóak többszámítások nélkül.

Későbbi használatra, az alábbi fogalmakat definiáljuk:

Topológiai szerkezet (TD – Topology Design): A topológiai optimalás végeredménye, ami tervezési tartományon belül értelmezett sűrűségértékekből áll (azaz az optimált változók értékeiből; SIMP esetén ezek a $[0,1]$ vagy $[x_{min}, 1]$ tartományban található értékek [6,7]; ESO/BESO esetén ezek a diszkrét 0,0/1,0 vagy $x_{min}/1,0$ értékek hard- illetve soft-kill eljárás esetén [8,9], [10]). A diszkretizáció történhet strukturált vagy strukturálatlan véges-elemes hálón, esetleg egyéb eljárásokkal is.

Csomópont (TD): Két vagy több rúd találkozása.

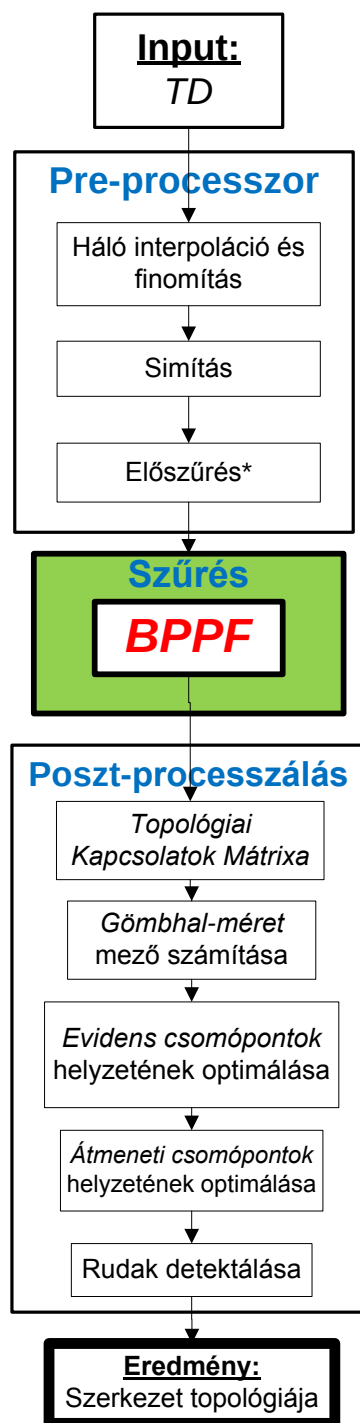
Végleges csomópont (TD): Olyan rudak kapcsolódási pontja, amelynek a végleges topológiában is léteznie kell.

Evidens csomópont (TD): Három vagy több rúd közötti csatlakozási pont. Egyben végleges csomópont is.

Átmeneti csomópont (TD): Pontosan két rúd közötti kapcsolódási pont. Ez a csomópont jelenthet egy átmeneti kapcsolatot a detektálás folyamatában, vagy egy végleges csomópontot amennyiben a rudak szöget bezáróan kapcsolódnak.

A detektáló algoritmus többszintű, és az alábbi lépésekre osztható fel (ld. 3. ábra):

0. lépés: Az algoritmus bemeneti adata egy TD.



3. ábra
Javasolt detektálási
algoritmus folyamatábrája

1. lépés: Pre-processor:

Ezen szint feladata a véges-elemes háló alapuló TD alapján a *BPPF* számára is értelmezhető adat létrehozása. Három fő lépésből áll:

a, Háló interpoláció és finomítás: Ezen lépés célja az *RTD* létrehozása, az alábbiak szerint definiálva:

Finomított Topológiai Szerkezet (RTD - Refined Topology Design): Strukturált diszkrétizált négyzetes (3D esetén kocka alapú) háló, mely a sűrűségértékeket tárolja. A *TD* értékeinek interpolációjával állítható elő.

Amennyiben az eredeti *TD* is ilyen hálóval került kiszámításra, akkor az *RTD* megegyezhet a *TD*-vel, ellenkező esetben két eljárás használata javasolt az interpolációhoz (ld. **4. ábra**).

- Egy nagyon finom strukturált négyzetes (2D)/kocka alakú (3D) elemekből álló háló létrehozása úgy, hogy az eredeti háló minden pontja az új hálón belül helyezkedjen el. Az egyes cellák által tárolt sűrűségértékek pedig megegyeznek majd azon eredeti cella értékével, amelyben az új cella középpontja elhelyezkedik. **Megjegyzés:** nagyon finom háló szükséges a megfelelő eredmény érdekében.

- Egy nagyon finom strukturált négyzetes (2D)/kocka alakú (3D) elemekből álló háló létrehozása úgy, hogy az eredeti háló minden pontja az új hálón belül helyezkedjen el. Az egyes cellák által tárolt sűrűségértékek úgy kerülnek kiszámításra, hogy első körben a cellánkénti sűrűségértékeket interpoláljuk csomópontonkéntira, majd második lépésben az eredeti véges-elem modell alakfüggvényeinek segítségével kiszámíthatóak az egyes sűrűségértékek bármely diszkrét pontban a sűrűségmező kiszámításának segítségével.

b, Simítás: Mivel az **1.a lépésben** különféle módszerek közül választhatunk, ezért olyan nagyfelbontású képekhez is eljuthatunk, ahol az eredeti cellák különálló diszkrét szerkezeti elemekként jelennek meg, ezért ajánlott a *TO*-ban alkalmazott hálófüggetlenségi szűrőhöz **[6,10,11]** hasonló szűrőt alkalmazni a szerkezetre $r_s = (0.5 - 1.0)r_{min}$ sugárral, ahol r_{min} a *TO*-ban alkalmazott szűrő sugara.

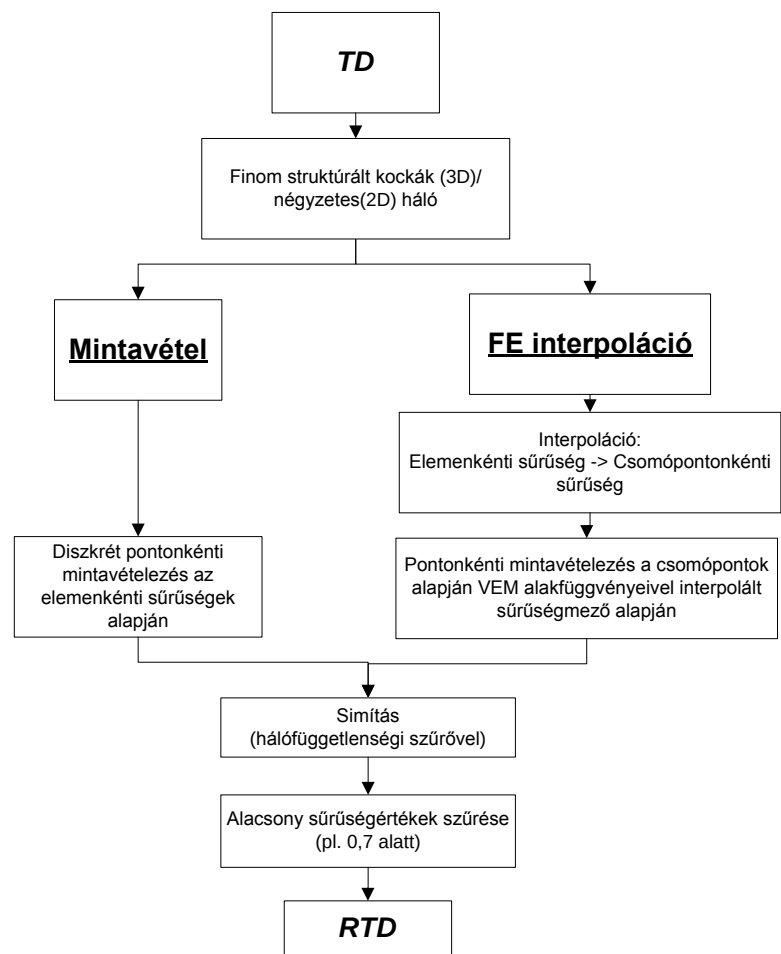
A hálófinomítást egy Ref_{mesh} faktorral fogjuk jelölni, ami azt mutatja meg, hogy hányszor finomabb az interpolált háló az eredetinél:

$$Ref_{mesh} = \sqrt[x]{\frac{\text{új háló elemeinek száma}}{\text{régi háló elemeinek száma}}} \quad \begin{array}{l} x = 2, \text{ ha } 2D \\ x = 3, \text{ ha } 3D \end{array}$$

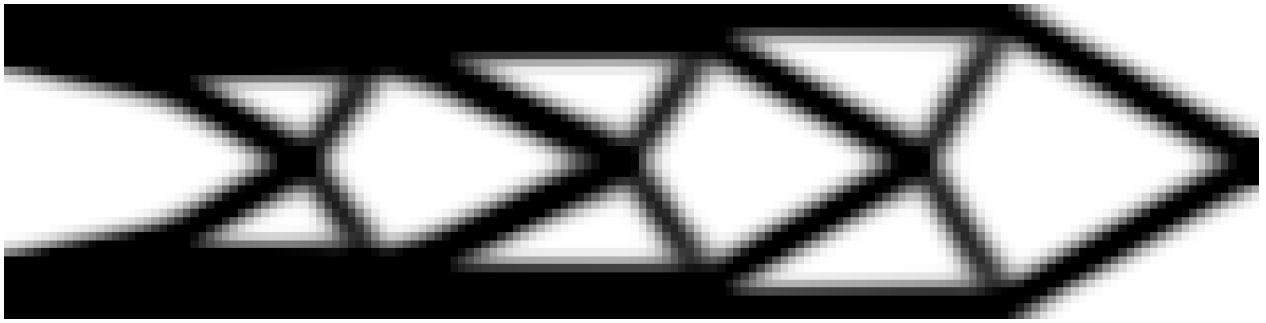
c, Előszűrés: Az előszűrés nem mást jelent, mint egy speciális *BPPF*, egészen pontosan a *JBSFTO* alkalmazását (*BPPF szűrőparancs szekvencia* pontos jelentése a **2. lépés**ben kerül bemutatásra, itt csak a teljesség kedvéért kerül említésre). Bár ez közvetlenül is hozzáadható lenne a **2. lépés**hez, de ennek ellenére ajánlott különválasztva tárgyalni őket. Ennek az oka nagyon egyszerű: a pre-processálás végén így kapott kép egy nagyfelbontású jó minőségű kép, mely akár más algoritmusokhoz is használható lenne bemenetként. Így az algoritmus ezen része önmagában is egy hasznos, erős algoritmust képvisel a TO finomítására.

Egy példa az **5. ábrán** kerül bemutatásra, a simítás és előszűrő alkalmazása nélküli eredmény a **6. ábrán**, míg a teljes algoritmus eredménye a **7. ábrán** látható.

3D-s esetre a **24./a ábrán** található példa a *TD*-re, simítás és előszűrés alkalmazása nélkül a **24./b**, míg a teljes algoritmus eredménye a **24./c ábrán** található.



4. ábra
1.a lépés



5. ábra

Eredeti TD: 160x40 felbontás

Megjegyzés: a szerkezet SIMP&OC eljárás segítségével készült



6. ábra

RTD: 960x160 felbontás, simítás és előszűrés nélkül



7. ábra

RTD: 960x160 felbontás (minden legalább 0,15 sűrűségű elem tömörnek van tekintve)

3. 2. lépés – BPPF

Ebben a részben részletesen ismertetésre kerül a bitminta-primitív szűrés alapötlete. Az alábbiakban megfogalmazott definíciók 2D-s esetre vonatkoznak. A 3D-s esetre vonatkozó definíciók a későbbiekben fognak ismertetésre kerülni.

Közbenső Topológiai Szerkezet (ITD - Intermediate Topology Design): Amennyiben bármilyen *szűrő* (sikeresen) alkalmazásra került az RTD-re, akkor az egy ITD lesz. Más néven ez lesz az *alap bitmező* (mivel minden cella 0 vagy 1 értéket tárol benne).

Pixel cella: Az RTD vagy ITD egyetlen cellája. Két dimenzió esetén ez egy kép egyetlen pixelét jelenti. Egy pixel jelölése $[k, l]$ (két dimenzióban), azaz ez a k . cella az x irányban, valamint az l . cella az y irányban.

Végleges Topológiai szerkezet (TDR - Topology Design Results): A szűrés folyamat végeredménye, ahol minden egyes pixel csak és kizárólagosan két *szomszédos* pixellel rendelkezik, amennyiben az nem egy *evidens csomópontot* képvisel.

Érték operátor: Egyetlen cella értékét mutatja meg: $\hat{f}[k, l]$. Üres cellák esetén ez 0, míg tömör cellák esetén 1 értéket ad vissza. További speciális értékek később kerülnek definiálásra.

Üres cella: A TDR vagy ITD egyetlen cellája, ha $\hat{f}[k, l] = 0$.

Tömör cella: A TDR vagy ITD egyetlen cellája, ha $\hat{f}[k, l] = 1$.

Általános szűrő: Bármely $r_x \times r_y$ méretű téglalap alakú terület.

Szimmetrikus szűrő: Bármely általános szűrő egyben szimmetrikus szűrő is, ha $r_x = r_y$ teljesül.

Szűrő cella: Egy szűrő egyetlen celláját azonosítja: $\langle i, j \rangle$, ha $1 \leq i \leq r_x$ és $1 \leq j \leq r_y$.

Érték operátor (szűrő esetén): Egyetlen szűrő cella értékét mutatja meg: $\hat{f}\langle i, j \rangle$.

A továbbiak az alábbi három-értékű reprezentáció kerül használatra.

Üres szűrő cella: Bármely általános szűrő bármely tetszőleges cellája, ha $\hat{f}\langle i, j \rangle = 0$.

Tömör szűrő cella: Bármely általános szűrő bármely tetszőleges cellája, ha $\hat{f}\langle i, j \rangle = 1$.

Detektor szűrő cella: Bármely általános szűrő bármely tetszőleges cellája, ha $\hat{f}\langle i, j \rangle = -1$.

Egyfókuszú szűrő: Bármely általános szűrő, amelyre teljesül, hogy $\hat{f}\langle i_1, j_1 \rangle = \hat{f}\langle i_2, j_2 \rangle = -1$ akkor és csakis akkor, ha $i_1 = i_2, j_1 = j_2$, azaz egyetlen detektor szűrő cellát tartalmaz.

Aktív szűrő: Bármely szűrő aktív szűrő a $[k, l]$ pozíciónál, ha $\hat{f}[k + i, l + j] = 0$ akkor és csakis akkor igaz, ha $\hat{f}\langle i, j \rangle = 0, 1 \leq i \leq r_x, 1 \leq j \leq r_y$. Az aktív szűrőket az aktiválási operátor 1 értékkel jelzi: $\hat{A}_{filter}[k, l] = 1$ (egyébként 0 értéket ad vissza).

Szűrők hatása az alap bitmezőre: Egy szűrőnek akkor és csakis akkor van hatása egy ITD-re, ha létezik olyan k, l egész számpár, melyre $\hat{A}_{filter}[k, l] = 1$. Amennyiben bármely szűrő esetén adott k, l -re $\hat{A}_{filter}[k, l] = 1$ teljesül, akkor a szűrő hatása a bitmezőre $f[k + i, l + j] := 0$ bármely $1 \leq i \leq r_x$, $1 \leq j \leq r_y$, -re, ahol $\hat{f}[i, j] = -1$ teljesül.

Szomszédossági operátor: Bármely két szűrő cella ($\langle i_1, j_1 \rangle$ és $\langle i_2, j_2 \rangle$), $1 \leq i_1, i_2 \leq r_x$, $1 \leq j_1, j_2 \leq r_y$ szomszédos akkor és csakis akkor, ha $\hat{f}[i_1, j_1] \neq 0$, $\hat{f}[i_2, j_2] \neq 0$, és $|i_1 - i_2| + |j_1 - j_2| = 1$. Ez az állítás egyenértékű a következővel: Két szűrő cella szomszédos, amennyiben nem üres cellák, és van egy közös oldaluk.

Ezt az állapotot a szomszédossági operátor 1-es értékkel fogja jelezni: $\hat{N}[i_1, j_1][i_2, j_2] = 1$ (egyébként 0 értéket ad vissza).

Megjegyzés: a szomszédossági operátor definiálható lenne az élek helyett a csúcsokon előírt kapcsolódással is, de az alábbi megközelítés nagyobb egyértelműséget és stabilitást garantál.

Kapcsolódási operátor: Bármely két cella ($\langle i_1, j_1 \rangle$ és $\langle i_n, j_n \rangle$), $1 \leq i_1, i_n \leq r_x$, $1 \leq j_1, j_n \leq r_y$ kapcsolatban áll egymással akkor és csakis akkor, ha $\hat{f}[i_1, j_1] \neq 0$, $\hat{f}[i_n, j_n] \neq 0$, és egy megfelelő $\langle i_1, j_1 \rangle, \langle i_2, j_2 \rangle, \dots, \langle i_n, j_n \rangle$ sorozat létezik, melyre $\hat{f}[i_m, j_m] \neq 0$ és $\hat{N}[i_m, j_m][i_{m+1}, j_{m+1}] = 1$ is teljesül $m = 1, 2 \dots (n - 1)$ esetén.

Ez az állítás egyenértékű a következővel: Két cella kapcsolatban áll egymással egy úton keresztül, ha egy olyan megfelelő indexpárokból álló sorozat létezik, melyben minden indexpár által reprezentált cella nem üres cella, és az egymás utáni indexek szomszédos cellákat jelölnek.

A kapcsolódást a kapcsolódási operátor 1-es értékkel fogja jelezni: $\hat{C}[i_1, j_1][i_2, j_2] = 1$ (egyébként 0 értéket ad vissza).

Az előző két definíció minden további kiegészítés nélkül alkalmazható az alap bitmezőre is, de a rövideg érdekében nem kerülnek külön bemutatásra.

Megfelelő Közbenső Topológiai Szerkezet (AITD - Appropriate Intermediate Topology Design): Bármely olyan ITD vagy ITR, amelyben ugyanolyan számú csomópont és azok közötti kapcsolódás szerepel, mint a TD esetében.

Az RTD-ből kiindulva lépésről lépésre finomítani fogjuk az IT-t közben ügyelve annak megfelelőségére – azaz addig törölünk pixeleket, míg el nem érjük a TDR-t.

Érvényes detektálási szűrő: Bármely olyan szűrő, amely egyszer alkalmazva az AITD egyetlen pixelére továbbra is AITD-t eredményez.

Az AITD fenntartásához definiálnunk kell az érvényes detektáló szűrőkre érvényes feltételeket is. Csak azok a szűrők megengedettek, amelyek nem tudnak letörölni egyetlen olyan pixelt sem, amelyek **lehetnének** az egyszerűsített szerkezet vázvonalaának részei (azaz a detektáló szűrő cella nem helyezkedhet el a szűrő szélén, illetve csak olyan pixeleket törölhetők, amelyek tömör pixeleket olyan csoportjait kötik össze, melyek a detektáló szűrő cellát nem tartalmazó úton is kapcsolódnak

egymáshoz. Ezeket a megfogalmazásokat összefoglalva az érvényes detektáló szűrőknek az alábbi feltételeket kell kielégíteniük:

- Ha $\hat{f}(i, j) = -1$, akkor $i \neq 1, i \neq r_x, j \neq 1$ és $j \neq r_y$
- Bármely két cellára, amelyek kielégítik, hogy $\hat{f}(i_1, j_1) = \hat{f}(i_2, j_2) = 1$, és $\hat{C}(i_1, j_1)(i, j) = 1$, $\hat{C}(i, j)(i_2, j_2) = 1$, ahol $f(i, j) = -1$; a $\hat{C}(i_1, j_1)(i_2, j_2) = 1$ állításnak is teljesülnie kell, ha a detektáló szűrő cellát figyelmen kívül hagyjuk.

Ezt az állapotot az *érvényességi operátor* 1-es értéke fogja jelezni $\check{V}[\text{filtername}] = 1$.¹ Ellenkező esetben 0 értéket ad vissza.

Ha megpróbáljuk meghatározni az összes lehetséges *érvényes detektáló szűrőt*, akkor egy erősen korlátozott tagszámú csoportot kapunk – amelyek forgatással vagy tükrözéssel nem transzformálhatóak egymásba-, ezt a csoportot fogjuk *bitminta-primitíveknek* nevezni.

Stabil detektálási szűrő: A *stabil szűrők* olyan *érvényes detektálási szűrők*, amelyek bármilyen tetszőleges kombinációban alkalmazhatóak anélkül, hogy olyan *TDR*-t vagy *ITD*-t eredményeznének, amely nem *AITD*.

Instabil detektálási szűrő: Olyan *szűrők*, amelyek – amennyiben többször, egymás után alkalmazásra kerülnek egy *ITD*-re – eredményezhetnek nem megfelelő *TDR*-t vagy *ITD*-t. Példának tekinthetünk olyan *szűrőket*, amelyek az eredeti *TD* egyes fontos részleteit letörölnék.

Érzékeny detektálási szűrő: Olyan *szűrő*, amelyeket ha más *szűrőkkel* nem megfelelő sorrendben alkalmazunk a szerkezetre, nem megfelelő *TDR*-t vagy *ITD*-t eredményezhet. Gyakori példa lehet egy olyan *szűrő*, amely megnöveli a *csomópontok* számát az eredeti szerkezethez képest, vagy letörölnék szükséges részleteket.

Bár az *érzékeny szűrők* alkalmazása megkérdőjelezhető, de sokszor ennek ellenére szükségesek (pl. egy *I-szűrő*, ld. **9. ábra**).

A továbbiakban a szűrési folyamat *3x3, szimmetrikus, egyfókuszú szűrők* segítségével lesz bemutatva *SIMP* eljárással készített topológiai optimálások eredményein prezentálva.

A teljes szűrési folyamat megértése érdekében néhány további koncepciót definiálunk az alábbiak szerint:

Szűrő szekvencia: Bármely olyan (értelmes) sorrendje a *szűrőknek* és *szekvencia operátoroknak*, amelyek egy *TDR*-t eredményeznek. Egy karakterlánc (sztring) segítségével reprezentáljuk.

Szűrő-azonosító: Egy karakter-kód, amely egy *bitminta-primitívet* azonosít.

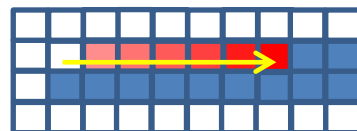
Szekvencia operátor: Olyan előre definiált operátor a *szűrő szekvenciában*, mely az egyes szűrők *ITD*-n való alkalmazásának módját határozza meg. Az alábbi implementációban a következő operátorok kerültek alkalmazásra:

¹ Ez a szükséges és elégséges feltétel ahhoz, hogy ne töröljünk egyetlen *csomópontot* illetve rudat sem. Ennek ellenére ez nem elégséges feltétel annak biztosításához, hogy a rudak és *csomópontok* számát ne növeljük. Ezért szükséges a *szűrő szekvenciák* alkalmazása.

- $\tilde{Z}$ – zéró operátor (zero operator): nincs hatása (csak lezárja bármely előző operátor hatását)
 - $\tilde{B}$ – Képhatár operátor (border operator): az *ITD* szélén található minden pixelt letöröl (így a különféle szűrők a kép szélén is alkalmazhatóak).
 - $\tilde{F}$ – Teljes-mező operátor (F-cycle, Full field operator): Az operátor az utána található összes szűrőt az összes lehetséges struktúrált irányban alkalmazni fogja az *ITD*-re. Programozási oldalról tekintve ez azt jelenti, hogy a *szűrő-szekvencia* minden pixelre két egymásba ágyazott for ciklus segítségével alkalmazva lesz mind a négy lehetséges struktúrált irányba: $+x + y$, $+x - y$, $-x + y$, $-x - y$. Bár első ránézésre ez az erőforrások pazarlásának tűnhet, de közelebbről megvizsgálva egy érdekes jelenséget fedezhetünk fel: amikor a különböző szűrők struktúrált irányokban kerülnek alkalmazásra, akkor sok esetben képesek akár egyetlen lépésben is képesek nagyszámú felesleges pixel eltávolítani. Ez a hatás kerül bemutatásra a **8. ábrán**. Egy $\tilde{S}$, $\tilde{Z}$, $\tilde{M}$, $\tilde{F}$ vagy $\tilde{D}$ operátor fogja lezárni *szekvencia* ezen részletét.
 - $\tilde{M}$ – Festett teljes-mező operátor (M-cycle, Marked full field operator): Az *M-ciklus* az *F-ciklus* egy speciális formájának tekinthető. Bár az *F-ciklus* nagyon gyors, de a szerkezet vázának meghatározása folyamán hajlamos a megmaradó, végső pixeleket az eredeti szerkezet határán megadni (ld. **10. ábra**). Ezért került ez az operátor definiálásra Egy $\tilde{S}$, $\tilde{Z}$, $\tilde{M}$, $\tilde{F}$ vagy $\tilde{D}$ operátor fogja lezárni ezt a *szekvenciát*. Az *M-ciklus* három fő lépésre osztható fel:
 - o Azon, pixelek megjelölése (az *ITD* módosítása nélkül) amelyek törölhetőek lennének. A festett kifejezés a katonaságban célpontok megjelölésére alkalmazott „megfestés” kifejezésből származik.
 - o A szűrő-szekvencia alkalmazása kizárólag a megjelölt pixelekre (**Megjegyzés:** ez nem jelenti azt, hogy ténylegesen minden bejelölt pixel törlésre is kerül, ebben az esetben ugyanis pl. a **8. ábra** esetén a teljes rúd törlésre kerülne).
 - o A megfestés megszüntetése.
- Számítástechnikailag ez nagyon könnyen megoldható, például az egyes cellákban tárolt (sűrűség)értékek egy előre meghatározott, kellően nagy számmal történő megszorzásával. A korábban meghatározott definíciók így érvényesek maradnak (azzal a feltételezéssel, hogy az 1 értékeket „nem 0”-ra cseréljük), és a jelölés megszüntetése egyetlen osztással elvégezhető.
- $\tilde{S}$ – Komplex teljes-mező operátor (S-cycle, Complex full field operator): A *komplex teljes-mező operátor* addig ismétli az utána található *szekvenciát*, amíg bármilyen változtatást tud eredményezni az *ITD*-n. Akár több *F*- vagy *M*- *ciklust* is tartalmazhat. Csak egy másik $\tilde{S}$, $\tilde{Z}$ vagy $\tilde{D}$ operátor fogja lezárni a *szekvencia* ezen részletét.
 - $\tilde{D}$ – Két-szintű teljes-mező operátor (D-cycle, Double-level Complex full field operator): A *két-szintű teljes-mező operátor* addig ismétli az utána található *szekvenciát*, amíg bármilyen változtatást tud eredményezni az *ITD*-n. Akár több *F*- vagy *S*- vagy *M*- *ciklust* is tartalmazhat. Csak egy másik $\tilde{D}$ vagy $\tilde{Z}$ operátor fogja lezárni ezt a *szekvenciát*.



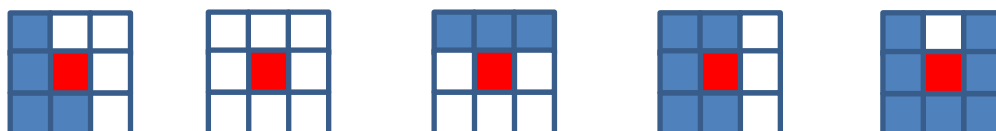
8/a
3x3 sík szűrő



8/b ábra
Egy szűrő speciális irányban történő ismételt alkalmazása
(piros szín – törölt pixelek, a szín intenzitása a sorrendet határozza meg)

Ezen tanulmány során a következőkben definiált 3x3 síkbeli *bitminta-szűrők* kerültek alkalmazásra. A bemutatásra kerülő mintapéldák során bizonyításra jut majd, hogy ez a megközelítés elégséges a feladat 2D-ben történő megoldásához. 3D-ban ez a megközelítés nem megfelelő, mivel a 3x3x3-as szűrők lehetséges száma 2^{26} nagyságrendjébe esik (beleértve a különböző *bitminta-primitívek* elforgatásait és tükörképeit is). Ezért egy hasonló megközelítés – az összes lehetséges szűrő elnevezése, és azok megfelelő sorrendjének megválasztása – nagyon hosszadalmas fejlesztési és előkészítési időt igényelne. A későbbiekben ezért egy eltérő megközelítés is bemutatásra kerül majd a 3D-s feladatokhoz.

A tesztekhez az alábbi 3x3-as szűrők kerültek elnevezésre a használat során:



9./a ábra

L-, N (zaj)-, T-, C-, U- szűrők

Sorrendben: 8, 1, 4, 4, 4 variáció különböző orientációkban

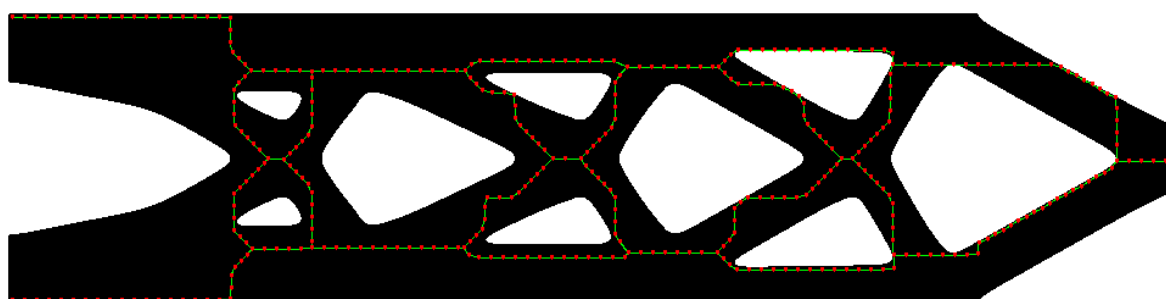


9./b ábra

O-, V-, I-, P- szűrők

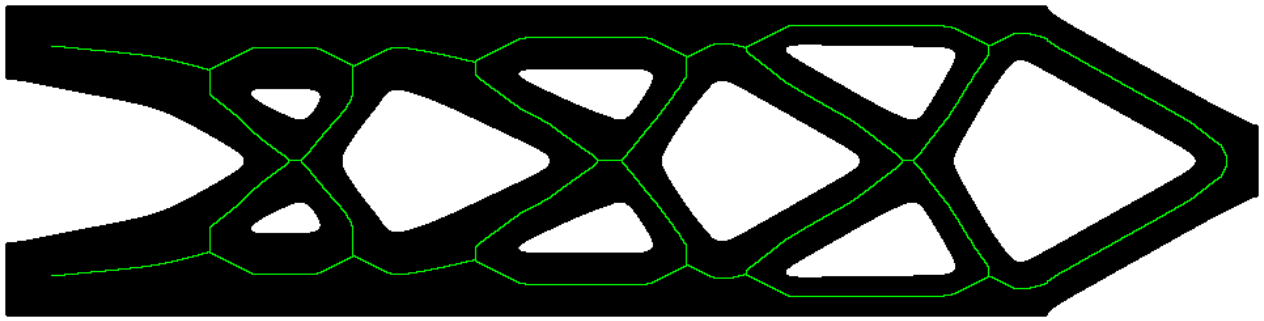
Sorrendben: 4, 4, 4, 4 variáció különböző orientációkban

A topológiai optimalás végeredménye a *BPPF* alkalmazásával együtt a **10.**, **11.**, **12.** ábrán láthatóak. Megfigyelhető, hogy az $\tilde{M}$ operátor alkalmazása általában jobb eredményeket ad, de lassabb is általánosságban.



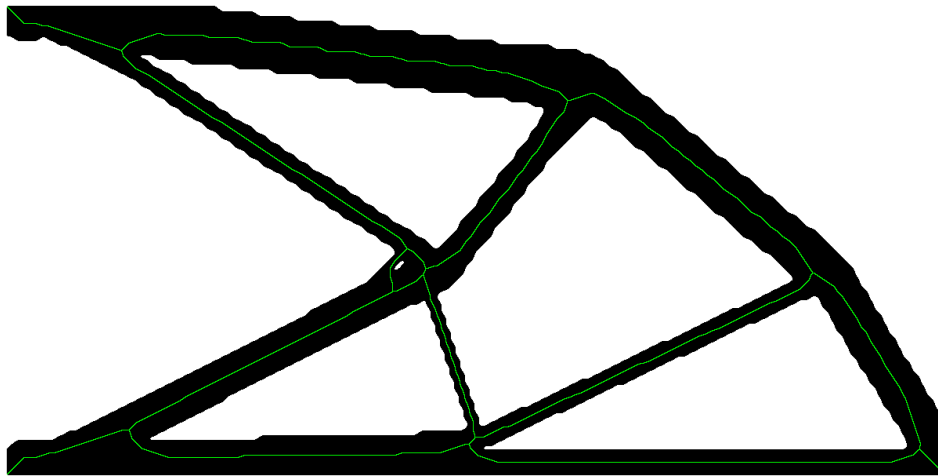
10. ábra

Szűrő-szekvencia: $\tilde{J} \tilde{B} \tilde{F} \tilde{N} \tilde{S} \tilde{F} \tilde{T} \tilde{P} \tilde{O} \tilde{S} \tilde{F} \tilde{T} \tilde{O} \tilde{P} \tilde{L} \tilde{V} \tilde{D} \tilde{F} \tilde{I} \tilde{F} \tilde{C} \tilde{S} \tilde{F} \tilde{T} \tilde{O} \tilde{P} \tilde{L} \tilde{V}$ (peremfeltétel kényszerítéssel)



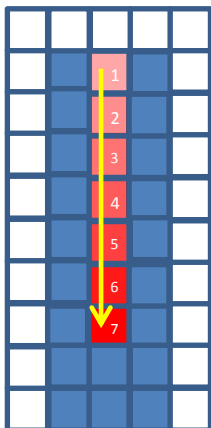
11. ábra

Szűrő-szekvencia: $\tilde{J} \tilde{B} \tilde{F} \tilde{N} \tilde{D} \tilde{M} \tilde{T} \tilde{O} \tilde{P} \tilde{L} \tilde{V} \tilde{C} \tilde{M} \tilde{L} \tilde{M} \tilde{P}$



12. ábra

Szűrő szekvencia: $\tilde{J} \tilde{B} \tilde{F} \tilde{N} \tilde{D} \tilde{M} \tilde{T} \tilde{O} \tilde{P} \tilde{L} \tilde{V} \tilde{C}$ (peremfeltétel kényszerítéssel)



13. ábra

U-szűrő ismételt alkalmazásának a hatása

Megjegyzés: a tesztek folyamán kiderült, hogy az *U-szűrők instabil szűrők*. Ez egy egyértelmű tény, tekintve, hogy például egy 3 pixel széles, 10 pixel hosszú pixelcsoportot a szűrő ismételt alkalmazása két rúddá alakítana át (ld. **13. ábra**). Ez a tény nagyon fontos lesz a későbbiekben a 3D-s szűrők esetében (*bemetszésre tiltott szűrők*)

Ezenfelül az *I-szűrők is instabil szűrők*, mivel képesek teljes rudak letörlésére. Ennek ellenére sok esetben használhatóak, mivel a mérnöki szerkezetek ritkán tartalmaznak olyan rudakat, amelyek egyik végükön nem csatlakoznak semmihez sem (kivétel, ha terhelés, vagy megfogás található a rúd végén). Ezen okból az *I-szűrők* is alkalmazhatóak a szűrési folyamat során, azzal a kitétellel, hogy a peremfeltételhez kapcsolódó pixelek nem törölhetők. Erre a továbbiakban *peremfeltétel kényszerítés* néven fogunk hivatkozni. A **10.** és **11. ábra** esetében a peremfeltétel manuálisan a teljes bal oldali falról a bal felső és alsó pontra lett cserélve.

Bár elsőre nehéz megérteni, hogy miért egy adott szűrősorrendet választunk, de –ahogy sok más területen is – itt is ez csak megfelelő tapasztalat, kísérletezés és időráfordítás kérdése.

A fejlesztés mostani fázisában a $\tilde{J} \tilde{B} \tilde{F} \tilde{N} \tilde{D} \tilde{M} \tilde{T} \tilde{O} \tilde{P} \tilde{L} \tilde{V} \tilde{C} \tilde{M} \tilde{L} \tilde{M} \tilde{P}$ szűrő-szekvencia majdnem 100%-os pontosságot produkált, (azaz egyetlen olyan topológiát sem találtunk, amelyre nem működött volna, ez önmagában egy figyelemreméltó tény).

Megjegyzés: Ennek ellenére a szűrő szekvencia megválasztása nem egy evidens feladat. Bár egyetlen hiba sem merült fel, ennek ellenére lehetséges, hogy létezik jobb (gyorsabb és pontosabb) szűrő-szekvencia (például ezen szűrő alkalmazása előtt több olyan szűrő is tesztelésre került, ami csak a sokadik, speciális esetben hibázott). Az algoritmus sebességét is a szekvencia határozza meg.

Annak bizonyítása, hogy maga a detektálási folyamat sem evidens, nagyon egyszerű. Tekintsünk egy egyszerű példát: adott egy egyenes rúd, aminek felületén egy kicsiny felületi hiba található. Hol találjuk azt a határt, ahonnan ezt egy különálló mérnöki részletnek (pl. bordának) tekintjük, vagy amikor már el kell tekintenünk tőle?

Ezt a határt pontosan a szűrő szekvencia fogja meghatározni. Például a **Függelékben** megtalálható példák közül számos a $\tilde{J} \tilde{B} \tilde{F} \tilde{N} \tilde{D} \tilde{M} \tilde{T} \tilde{O} \tilde{P} \tilde{L} \tilde{V} \tilde{C} \tilde{F} \tilde{T} \tilde{P} \tilde{O}$ vagy $\tilde{J} \tilde{B} \tilde{F} \tilde{N} \tilde{D} \tilde{M} \tilde{T} \tilde{O} \tilde{P} \tilde{L} \tilde{V} \tilde{C}$ szekvenciával készült el, amelyek hajlamosak kis rudakat létrehozni a rudak sarkainál egyes esetekben (bár ezt a későbbiekben tárgyalt *LLPO* kijavítja), míg a most ismertetett szekvencia hajlamos inkább a vastag rudak orientációját a horizontális vagy vertikális irányhoz közelíteni (de azok száma és topológiája nem sérül).

Emellett fontos meglátni azt, hogy végfelhasználói, kereskedelmi termékben a szűrő szekvencia megválasztása nem szerepel, ez a fejlesztő feladata lesz, hogy kiválasszon egy megfelelő sorrendet. Ennek ellenére – pl. igény esetén – nyitva hagyhatja a lehetőséget, hogy a felhasználó felülbírálja megfelelő szaktudás esetén a beépített sorrendet. A felhasználó egyetlen feladata az, hogy meghatározza a szerkezetre vonatkozó kritériumokat a poszt-processzáláshoz (ld. **3. lépés**).

4. 3. lépés – Poszt-processzálas

A poszt-processzálas feladata nem más, mint a legkisebb olyan *TCM* (lásd később) megalkotása, mely megfelelően le tudja írni a szerkezet topológiáját. Bár a szerkezet 'csontváza' sikeresen detektálva van, de ennek ellenére az eredmények ebben a formában még nem használhatóak. Az *evidens csomópontok* azonnal felismerhetők, de azok a *végleges csomópontok*, amelyek szög alatt csatlakozó rudakat kötnek össze, még nem. Abban az esetben viszont, amennyiben a szerkezet nem tartalmaz ilyen rudakat (pl. a szerkezet kizárólag egyenes rudakból áll), a folyamat gyakorlatilag készen is van, mivel az *evidens csomópontok* illetve a peremfeltételnél található *csomópontok* lesznek a *végleges csomópontok*, így ebben az esetben nem szükséges a *poszt-processzálas* végrehajtani. Amennyiben tovább szeretnénk növelni a rudak helyzetének pontosságát, illetve kiterjeszteni az eljárást pl. ívelt rudak esetére is, akkor néhány további definíció szükséges:

Topológiai Kapcsolatok Mátrixa (a továbbiakban TCM - Topology Connectivity Matrix): Egy olyan mátrix, amely a csomópontok számát, és azok közötti kapcsolatokat (rudak) határozzák meg. Amennyiben a csomópontok számát n -nel jelöljük, akkor ez egy $n \times n$ méretű mátrix.

Megjegyzés: Amennyiben az i . csomópont kapcsolódik a j . csomópontához, akkor ez fordítva is igaz, azaz:

$$t_{ij} = t_{ji} = 1.$$

,tehát a mátrix szimmetrikus:

$$\mathbf{T} = \mathbf{T}^T$$

Egyenes középvonalú rudak esetén t_{ij} nem nulla értéke jelzi a kapcsolatot, egyébként pedig ez az érték határozza meg a rúd típusát.

Erősen ajánlott számítástechnikai okokból egy 'sparse' mátrixban tárolni az adatokat, mivel ez a mátrix egy ritka mátrix. A mátrixot $\mathbf{T}$ -vel jelöljük, míg az egyes *csomópontok* pozícióját a $\mathbf{p}$ -vel jelölt *pozíció vektor* fogja tárolni. Egyetlen csomópont pozíciója a $\mathbf{p}_i$ helyen található, és az egyes komponenseket pedig $p_{i,x}$, $p_{i,y}$ jelöli.

Gömbhal operátor: A gömbhal operátor elnevezés egy egyszerű hasonlatból származik: A szerkezet minden egyes pontjába gömbhalakat helyezünk el, majd gonosz módon megijesztjük azokat. Ekkor a halak akkorára fúvódnak fel, amíg a szerkezet falához nem érnek. A gömbhal operátor mérete tulajdonképpen a gömbhal területét (síkbeli halakat képzelünk el), azaz egy adott pontba beírható legnagyobb olyan kör területét jelenti, melynek kerülete nem metszi a szerkezet határvonalát. Matematikailag is megfogalmazva:

$$\hat{P}[k, l] = \max(r^2 \pi), \text{ minden } r \text{ értékre, mely kielégíti } \hat{f}[k + r' \cos \alpha, l + r' \sin \alpha] = 1 - t$$

$$\text{ha } 0 \leq r' \leq r, 0 \leq \alpha \leq 2\pi$$

A gömbhal operátor által meghatározott skalármező értéke a **14. ábrán** látható. Megfigyelhető, hogy a függvény szélsőértékei jelölik a szerkezet középvonalát (ez is felhasználható detektálásra, de a diszkretizációs hibák miatt egy másik megközelítés szükséges a detektáláshoz).

olyan irányokba, ami közel van egy másik, létező kapcsolat irányához, azaz megtiltunk minden olyan mozgást, amely egy már létező kötés irányában történne.

Matematikailag az *LLPO* az alábbi formában fogalmazható meg:

$$\text{Maximáljuk: } \hat{P} \left[p_{i,x} + d_x, p_{i,y} + d_y \right] \rightarrow \max$$

$$\text{Úgy, hogy } \theta_j \leq \alpha_{\max}, \text{ ahol } \theta_j = \arccos \frac{(d_x e_x + d_y e_y) \cdot (p_i - p_j)}{|d_x e_x + d_y e_y| \cdot |p_i - p_j|} \text{ ha } t_{ij} \neq 0.$$

(A lehetséges mozgás iránya és a csomópontot egy másik csomóponttal összekötő irány által bezárt szög).

$$\text{ahol, } 0 \leq d_x, d_y \leq D_{\max}$$

Az új pozíció pedig $p_i^{\text{new}} = p_i^{\text{old}} + d_x e_x + d_y e_y$ lesz.

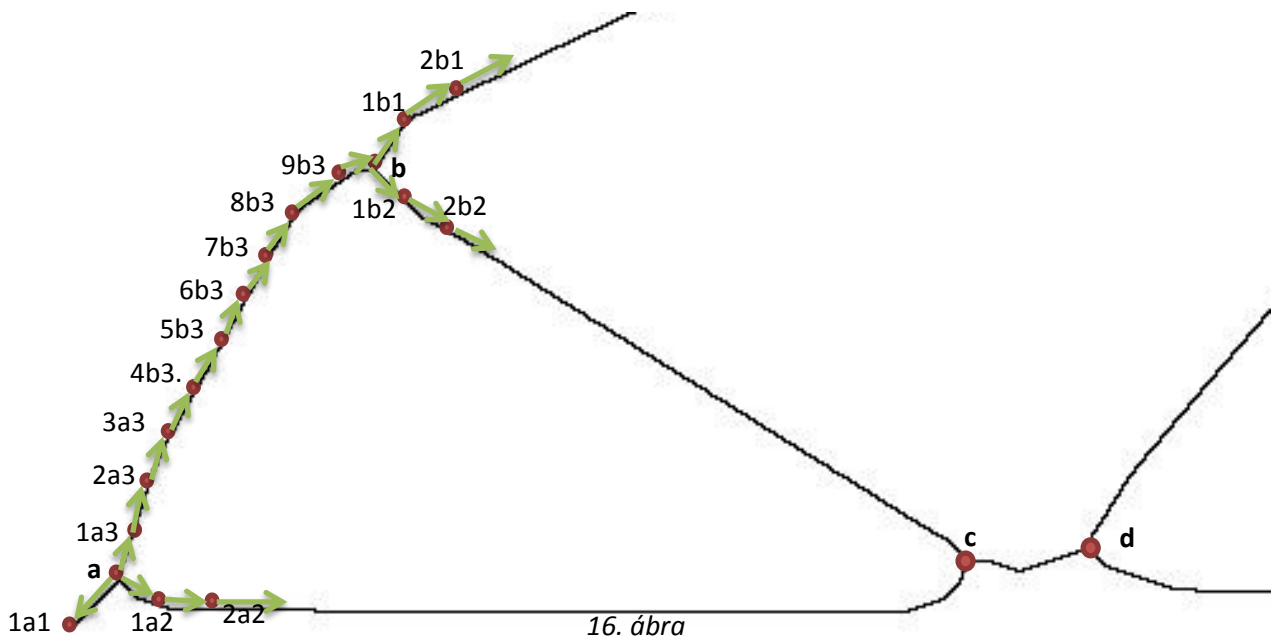
Általában $D_{\max} = (0.8 - 1.1) \cdot Ref_{\text{mesh}}$ érték egy jó választás a folyamathoz. A javasolt paraméterek a szögekre $\alpha_{\max} = 6^\circ$ az *evidens csomópontokra*, és $\alpha_{\max} = 12^\circ$ az *átmeneti csomópontokra*. A teljes optimalizációs folyamatot addig ismétljük, amíg a pozíciók nem konvergálnak.

A fenti koncepciók felhasználásával már befejezhető az algoritmus részletes leírása.

3./a lépés: Egy nagy felbontású TCM (Topology Connectivity Matrix) létrehozása

Ezen lépés célja a megmaradó, vázvonalat alkotó pixelekből egy kellően részletes *TCM* létrehozása úgy, hogy kellően nagyszámú *csomópont* és rúd alkossa azt egy részletes felbontás érdekében a további feldolgozáshoz. Az eljárás maga egy meglepően egyszerű rekurzív algoritmus.

Az algoritmus az *evidens csomópontokból* (olyan átmeneti csomópontok, amelyek 1,3,4.. kapcsolattal rendelkeznek) indul, és ezen pontokból kiindulva az egymáshoz kapcsolódó pixelek során keresztül egyszerre, párhuzamosan végighalad a szerkezet „csontvázán”. Összesen $Step_{\text{mov}}$ számú pixelen történő áthaladás után pedig létrehoz egy új *átmeneti csomópontot*, és összeköti azt az előzővel. Amennyiben pedig *evidens csomópont*hoz ér, akkor befejezi a ciklust, és összeköti a két *csomópontot*. Miközben az algoritmus halad előre, automatikusan letörli a már feldolgozott pixeleket, így megakadályozva egy útvonal többszöri detektálását. Az algoritmus működésének elve a **16. ábrán** látható.



3./d lépés közbeni eredménye

XYZ – X=Hányadik lépés ez útvonalon; Y=azt az evidens csomópontot azonosító karakter, ahonnan az adott ciklus elindult; Z – közös evidens csomópontból induló utak azonosítója

Ajánlott érték az algoritmushoz: $Step_{mov} = (0.4 - 0.7) \cdot Ref_{mesh} \cdot r_{min}$.

3./b lépés: Gömbhal-méret mező kiszámítása

A korábban definiálásra került $\hat{P}$ operátor alkalmazását jelenti a teljes *alap bitmezőre*, mivel ez az érték szükséges lesz a következő lépések során.

3./c lépés: Evidens csomópontok helyzetének optimalizálása

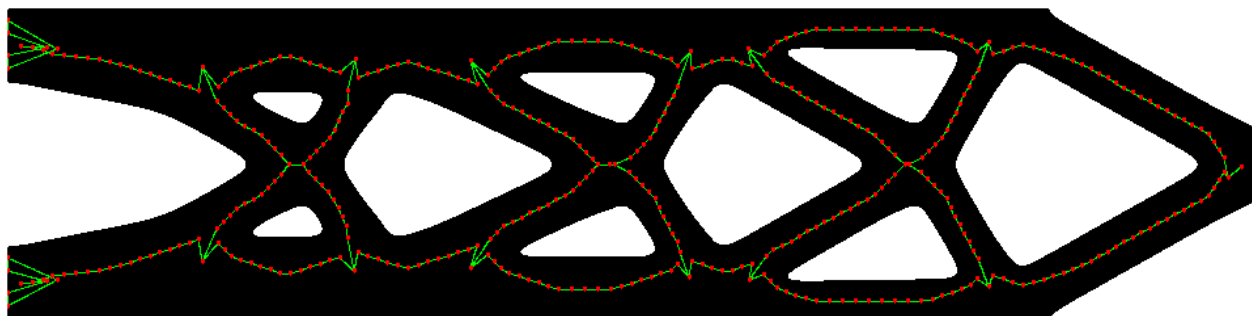
Bár nem magától értetődő, de a tapasztalatok fényében hasznos, amennyiben csak az *evidens csomópontok helyzetét* a *tiltott szögtartományra* vonatkozóan egy kevésbé szigorú megkötéssel (azaz nagyobb paraméterrel) optimalizáljuk, így azok az optimális pozíciókba mozoghatnak még azelőtt, hogy más *csomópontok* helyzetét korrigálnánk.

Bár a TCM kezelésére a fentebb említett szubrutinok elégségesek, de további kisegítő szubrutinok beiktatásával jelentősen növelhető az eljárás hatékonysága. Ilyen szubrutin például az egymáshoz közel elhelyezkedő *csomópontok* egyesítése, azaz amennyiben bármely rúd esetén, melyre igaz, hogy $|\mathbf{p}_i - \mathbf{p}_j| < Limit_{merge}$, (ha $t_{ij} = 1$), akkor a rúd egyetlen *csomóponttá* kerül összeolvasztásra. A rudat alkotó eredeti *csomópontok* kapcsolatai az új *csomópont*éhoz lesznek adva, és az új *csomópont* helyzete annak a csomópontnak a helyzetével fog megegyezni, amelyik nagyobb számú vagy pontosan 1 kapcsolattal rendelkezett, egyéb esetekben (megegyező számú kapcsolat esetén) pedig az eredeti *csomópontok* átlaga adja meg az új pozíciót.

Így az optimalizálás folyamán az egymáshoz túl közel elhelyezkedő *csomópontok* egyesítésre kerülnek, és az algoritmus ellenőrzi a kisméretű zárt hurkokat, illetve szabadon lévő rúdvégeket is (bár ez csak

biztonságot hivatott növelni, mivel a *BPPF* algoritmus nem hajlamos ilyen szerkezeteket létrehozni, csak ritka, speciális esetekben). Így a kiegészített szubrutinokkal együtt az algoritmus nagyon hatékonyan képes mind a *csomópontok* számának csökkentésére (az egyesítések révén), mind a pozíciók helyesbítésére (mivel a csökkenő számú *csomóponttal* csökken a szögekre vonatkozó megkötések száma is).

A javasolt értékek erre vonatkozóan $Limit_{merge} = (0.6 - 0.8) \cdot Ref_{mesh}$.

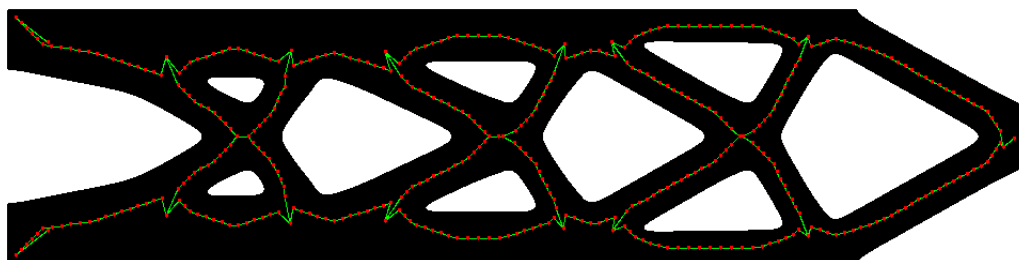


17./a ábra
3./c lépés eredménye²

3./d lépés: Átmeneti csomópontok helyzetének optimalizálása

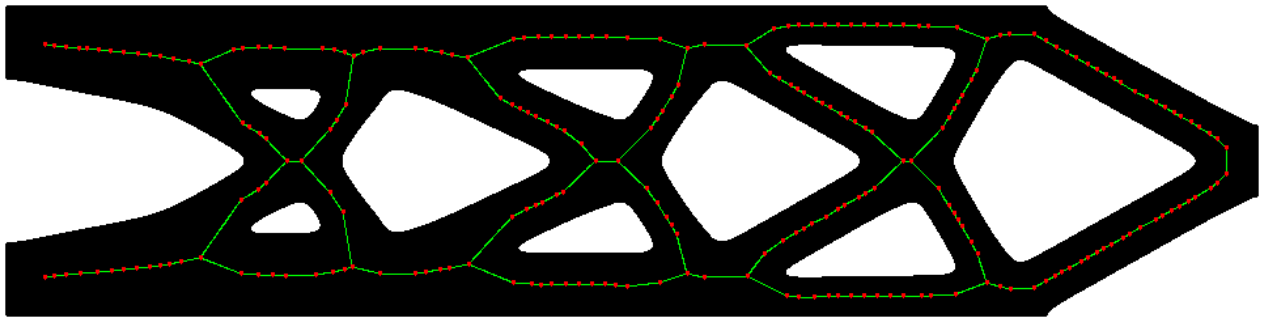
A következő lépésben az *LLPO* eljárás mind az *evidens*, mind az *átmeneti csomópontokra* alkalmazásra kerül, azaz az összes pozíciót optimaljuk. Ennek segítségével drámaian növelhető a végeredmény pontossága. A 3./d lépés eredménye látható egy mintafeladaton a **18. ábrán**.

Az így kapott *TCM* már alkalmazható lenne egy méretezési feladathoz – a 3./e lépés kihagyásával is akár – azonban még így is viszonylag nagy számú változót tartalmazna. Pontosán ezért ajánlott a 3./e lépés (ld. később) végrehajtása a csomópontok és rudak számának csökkentése érdekében.



17./b ábra
3./c lépés eredménye (peremfeltétel kényszerítés kizárólag a bal oldali sarkokra)

² Megjegyzés: a falnál a peremfeltétel nem két pontra vonatkozott, hanem a teljes fal befalazására, innen ered az érdekes, meglepő viselkedés. Mivel a peremfeltétel kényszerítés a teszt során aktív volt, így az algoritmus a teljes falat detektálta. Ennek ellenére ez nem okozhat problémákat – bár személyes kételyek esetén nyugodtan helyettesíthetjük a teljes falra vonatkozó peremfeltételt a bal felső és alsó pontra vonatkozó előírással-, mivel az *LLPO* folyamat javítja ezt a “hibát”, ahogy az a **18. ábrán** is látható.



18. ábra
3./d lépés eredménye

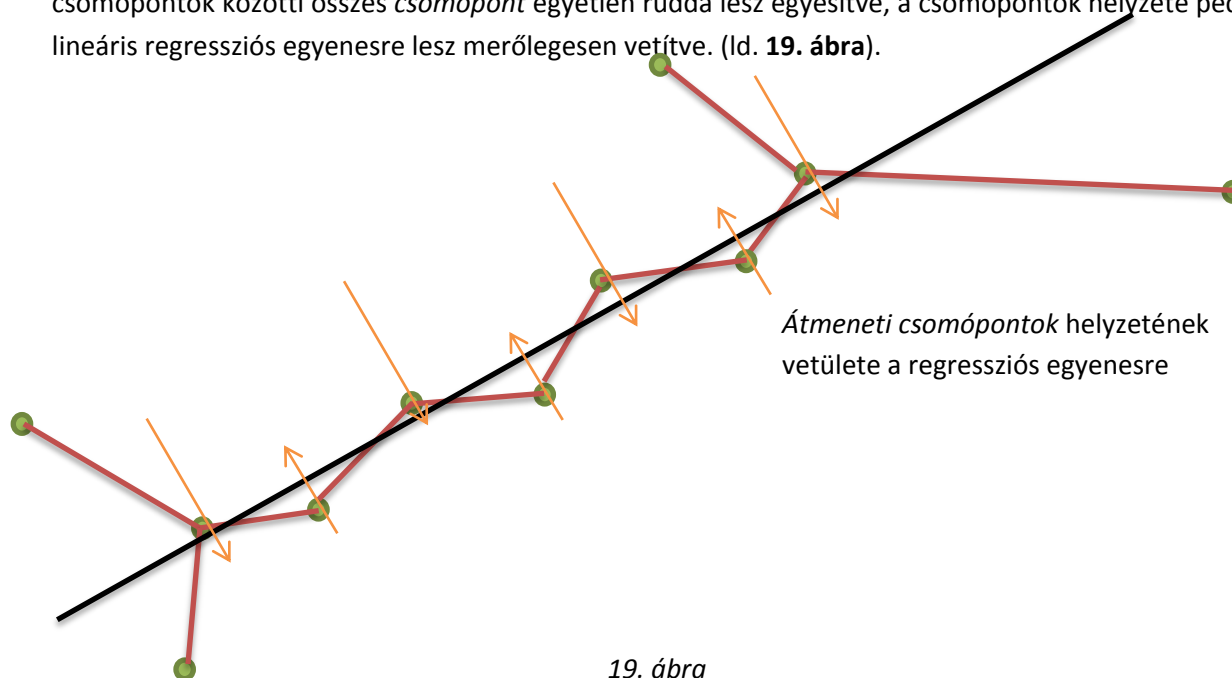
Fontos megjegyezni, hogy a tesztek folyamán az *LLPO* eljárás hajlamos volt az *evidens csomópontok* közelében az *átmeneti csomópontok* számát jelentősen csökkenteni (ld. **18. ábra**). Bár egyik oldalról nézve ez helytelennek tűnhet, mivel így csökken a *TCM* felbontása az *evidens csomópontok* lokális környezetében, azonban a rudak helyzetének pontosabb meghatározását ez pontosan elősegíti, mivel az – akár még mindig pontatlan helyzetű - *evidens csomópontok* így kisebb súllyal kerülnek figyelembe vételre a rudak detektálása folyamán.

Ennek ellenére a későbbiekben egy fejlettebb (még csak egyes részleteiben kész) *LLPO* eljárás kerül majd alkalmazásra a szoftverben, amely már megkötéseket fog tartalmazni nem csak a rudak közötti minimális és maximális távolságokra, de az egy *csomópontból* kiinduló rudak hosszára is a nagyobb pontosság érdekében.

3./e lépés: Rudak detektálása

A konkrét gépészeti részletek detektálása nem alkotja az algoritmus szerves részét, hanem a tervezést vezető mérnöknek kell meghatározni az ehhez szükséges feltételeket, hogy pontosan milyen rudak (pl. egyenes középvonalú, vagy ívelt), milyen csomópontok (pl. K-kötés) megengedettek a tervezés folyamán, illetve milyen méretű a szerkezet, és mi a legkisebb méret, amit még megkülönböztetünk (pl. egy híd esetén nyilvánvalóan nem fogunk a 1 mm-es mérettartománnyal foglalkozni). A méretezési feladat azonban már ekkor is elvégezhető lenne további egyszerűsítések nélkül. A teljesség kedvéért bemutatásra kerül egy egyszerű, de meglepően hatékony eljárás egyenes rudak detektálására.

A, Amennyiben két *evidens csomópont* közötti összes *átmeneti csomópont*ra teljesül, hogy azok lineáris regressziója kisebb, mint egy előre definiált tűrés (mely a rúdvastagság nagyságrendjébe esik), illetve az *átmeneti csomópontok* által meghatározott szomszédos rudak által bezárt szögek, illetve azok előjeles összege is kisebb egy előre meghatározott határnál, akkor a kérdéses *evidens csomópontok* közötti összes *csomópont* egyetlen rúddá lesz egyesítve, a csomópontok helyzete pedig lineáris regressziós egyenesre lesz merőlegesen vetítve. (ld. **19. ábra**).



19. ábra
Átmeneti csomópontok lineáris regressziója

Az ajánlott értékek:

$$R_{max} = (0.8 - 1.2) \cdot Ref_{mesh} \text{ és } \beta_{max} = 15 - 25^\circ$$

Amennyiben az ismertetett feltételek teljesülnek, akkor az *átmeneti csomópontok* egyetlen rúddá lesznek egyesítve.

B, Az *evidens csomópontokból* kiindulva minden lehetséges irányban kiválasztjuk azon lehető legnagyobb számú *átmeneti csomópontot*, amíg azokra még teljesül az **A**-ban megfogalmazott

kritérium. Azaz amennyiben a feltétel még igaz az első j átmeneti csomópontra, de nem igaz $j + 1$ átmeneti csomópontra, akkor az első j átmeneti csomópont egyesítve lesz egyetlen rúddá.

C, Az **A** kritérium tesztelve lesz bármely két átmeneti csomópont közötti legrövidebb útra – amely nem tartalmaz evidens csomópontot –, és amennyiben teljesül, akkor azok egyesítve lesznek egy rúddá.

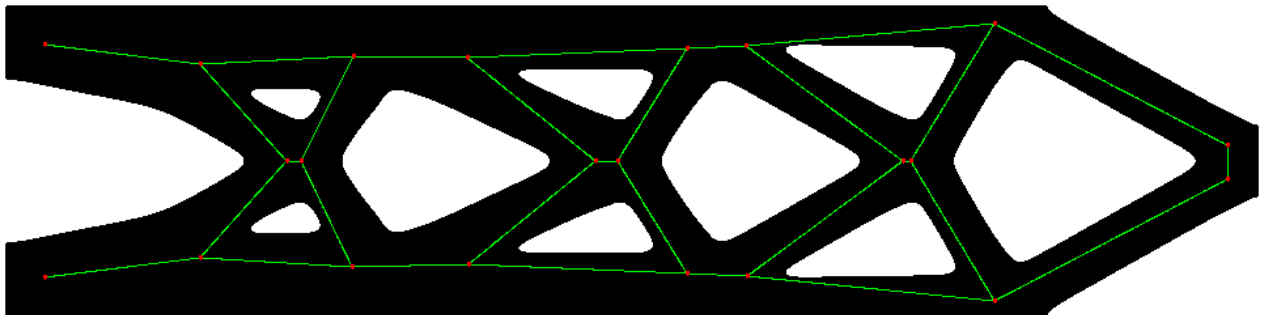
Bár megkérdőjelezhető az a megközelítésmód, hogy miért nem alkalmazunk előre definiált kritériumokat, az ok ennek ellenére nagyon egyszerű: ahhoz, hogy egy szerkezetet pontosan detektálhassunk, szükséges információkkal rendelkezünk annak céljáról, méreteiről, és gyártástechnológiájáról egyaránt.

Például előfordulhat, hogy az optimalás végeredménye egy ívelt tartó, de amennyiben az adott cég csak egyenes középvonalú rudakkal dolgozik, attól még továbbra is helytelen lenne azt sok, kis méretű rúdból kiszerkeszteni. Fordítva ez még nagyobb hibának tűnne: azaz csak egyenes rudakat használni akkor, mikor megfelelő, ívelt rudak is rendelkezésre állnak.

Ezen okok miatt a leginkább reális megközelítésnek az tűnne, amennyiben kiegészítenénk a szoftvert egy adatbázissal, mely nagyszámú, különféle gerendákra és kötésekre vonatkozó feltételt tartalmazni, és a tervező egyszerűen kiválaszthatná, mely szerkezeti elemekkel szeretne dolgozni a *poszt-processzálas* folyamán.

4. lépés:

A teljes számítás végeredménye a *TCM*, valamint a hozzá kapcsolódó pozíció vektor. Az algoritmus végeredménye a **20. ábrán** látható.



20. ábra

A detektálás végeredménye – detektált topológia
Piros pontok=csomópontok, zöld vonalak=rudak

A mellékelt **függelékben** az olvasó további példákat találhat az algoritmus alkalmazására. Ezekhez a topológiák egy része BESO, más részük SIMP módszerrel készült.

5. BPPF implementálása 3D-s szerkezetekre

Ahhoz, hogy a BPPF ötlete alkalmazható legyen 3D-s esetekre is, néhány fogalmat definiálnunk kell megint:

Közbenső Topológiai Szerkezet (ITD - Intermediate Topology Design): Amennyiben bármilyen *szűrő* (sikeresen) alkalmazásra került az RTD-re, akkor az egy ITD lesz. Más néven ez lesz az *alap voxelmező* (mivel minden cella 0 vagy 1 értéket tárol benne).

Voxel cella: Az RTD vagy ITD egyetlen cellája. Egy voxel jelölése $[l, m, n]$ (három dimenzióban), azaz ez a l . cella az x irányban, valamint az m . cella az y irányban illetve az n . cella az z irányban.

Végleges Topológiai szerkezet (TDR - Topology Design Results): A szűrési folyamat végeredménye, ahol minden egyes *voxel* csak és kizárólagosan két *szomszédos voxel*rel rendelkezik, amennyiben az nem egy *evidens csomópontot* képvisel.

Érték operátor: Egyetlen cella értékét mutatja meg: $\hat{f}[l, m, n]$. Üres cellák esetén ez 0, míg *tömör cellák* esetén 1 értéket ad vissza. További speciális értékek később kerülnek definiálásra.

Üres cella: A TDR vagy ITD egyetlen cellája, ha $\hat{f}[l, m, n] = 0$

Tömör cella: A TDR vagy ITD egyetlen cellája, ha $\hat{f}[l, m, n] = 1$

Általános szűrő: Bármely $r_x \times r_y \times r_z$ méretű téglalap alakú terület.

Szimmetrikus szűrő: Bármely *általános szűrő* egyben *szimmetrikus szűrő* is, ha $r_x = r_y = r_z$ teljesül.

Szűrő cella: Egy *szűrő* egyetlen celláját azonosítja: $\langle i, j, k \rangle$, ha $1 \leq i \leq r_x$ és $1 \leq j \leq r_y$, $1 \leq k \leq r_z$.

Érték operátor (szűrő esetén): Egyetlen *szűrő cella* értékét mutatja meg: $\hat{f}\langle i, j, k \rangle$

A továbbiakban az alábbi három-értékű reprezentáció kerül használatra.

Üres szűrő cella: Bármely *általános szűrő* bármely tetszőleges *cellája*, ha $\hat{f}\langle i, j, k \rangle = 0$.

Tömör szűrő cella: Bármely *általános szűrő* bármely tetszőleges *cellája*, ha $\hat{f}\langle i, j, k \rangle = 1$.

Detektor szűrő cella: Bármely *általános szűrő* bármely tetszőleges *cellája*, ha $\hat{f}\langle i, j, k \rangle = -1$.

Egyfókuszú szűrő: Bármely *általános szűrő*, amelyre teljesül, hogy $\hat{f}\langle i_1, j_1, k_1 \rangle = \hat{f}\langle i_2, j_2, k_2 \rangle = -1$ akkor és csakis akkor, ha $i_1 = i_2$, $j_1 = j_2$, és $k_1 = k_2$, azaz egyetlen *detektor szűrő cellát* tartalmaz.

Aktív szűrő: Bármely *szűrő* *aktív szűrő* a $[l, m, n]$ pozíciónál, ha $\hat{f}[l + i, m + j, n + k] = 0$ akkor és csakis akkor igaz, ha $\hat{f}\langle i, j, k \rangle = 0$, $1 \leq i \leq r_x$, $1 \leq j \leq r_y$, $1 \leq k \leq r_z$. Az *aktív szűrőket* az *aktiválási operátor* 1 értékkel jelzi: $\hat{A}_{filter}[l, m, n] = 1$ (egyébként 0 értéket ad vissza).

Szűrők hatása az alap voxelmezőre: Egy szűrőnek akkor és csakis akkor van hatása egy ITD-re, ha léteznek olyan l, m, n egész számok, melyre $\hat{A}_{filter}[l, m, n] = 1$. Amennyiben bármely szűrő esetén

bármely l, m, n -re $\hat{A}_{filter}[l, m, n] = 1$ teljesül, akkor a szűrő hatása a bitmezőre $f[l + i, m + j, n + k] := 0$ bármely $1 \leq i \leq r_x, 1 \leq j \leq r_y, 1 \leq k \leq r_z$, -re, ahol $\hat{f}\langle i, j, k \rangle = -1$ teljesül.

Szomszédossági operátor: Bármely két szűrő cella $\langle i_1, j_1, k_1 \rangle$ és $\langle i_2, j_2, k_2 \rangle$, $1 \leq i_1, i_2 \leq r_x, 1 \leq j_1, j_2 \leq r_y, 1 \leq k_1, k_2 \leq r_z$ szomszédos akkor és csakis akkor, ha $\hat{f}\langle i_1, j_1, k_1 \rangle \neq 0$, $\hat{f}\langle i_2, j_2, k_2 \rangle \neq 0$, és $|i_1 - i_2| + |j_1 - j_2| + |k_1 - k_2| = 1$. Ez az állítás egyenértékű a következővel: Két szűrő cella szomszédos, amennyiben nem üres cellák, és van egy közös lapjuk. Ez gyakran lesz közvetlen kapcsolatként is megnevezve.

Ezt az állapotot a szomszédossági operátor 1-es értékkel fogja jelezni: $\hat{N}\langle i_1, j_1, k_1 \rangle \langle i_2, j_2, k_2 \rangle = 1$ (egyébként 0 értéket ad vissza).

Megjegyzés: a szomszédossági operátor definiálható lenne a lapok helyett a csúcsokon vagy éleken előírt kapcsolódással is, de az utóbbi megközelítés nagyfokú instabilitást okozna 3D-ben.

Kapcsolódási operátor: Bármely két cella $\langle i_1, j_1, k_1 \rangle$ és $\langle i_n, j_n, k_n \rangle$, $1 \leq i_1, i_n \leq r_x, 1 \leq j_1, j_n \leq r_y, 1 \leq k_1, k_n \leq r_z$ kapcsolatban áll egymással akkor és csakis akkor, ha $\hat{f}\langle i_1, j_1, k_1 \rangle \neq 0$, $\hat{f}\langle i_n, j_n, k_n \rangle \neq 0$, és egy megfelelő $\langle i_1, j_1, k_1 \rangle, \langle i_2, j_2, k_2 \rangle, \dots, \langle i_n, j_n, k_n \rangle$ sorozat létezik, melyre $\hat{f}\langle i_m, j_m, k_m \rangle \neq 0$ és $\hat{N}\langle i_m, j_m, k_m \rangle \langle i_{m+1}, j_{m+1}, k_{m+1} \rangle = 1$ is teljesül $m = 1, 2 \dots (n - 1)$ esetén.

Ez az állítás egyenértékű a következővel: Két cella kapcsolatban áll egymással egy úton keresztül, ha olyan megfelelő indexpárokból álló sorozat létezik, melyben minden indexpár által reprezentált cella nem üres cella, és az egymás utáni indexek szomszédos cellákat jelölnek.

A kapcsolódást a kapcsolódási operátor 1-es értékkel fogja jelezni: $\hat{C}\langle i_1, j_1, k_1 \rangle \langle i_2, j_2, k_2 \rangle = 1$ (egyébként 0 értéket ad vissza).

Az előző két definíció minden további kiegészítés nélkül alkalmazható az alap voxelmezőre is, de a rövideg érdekében nem kerülnek külön bemutatásra itt.

Érvényes detektálási szűrő: Bármely olyan szűrő, amely egyszer alkalmazva az AITD egyetlen pixelére továbbra is AITD-t eredményez.

Az AITD fenntartásához definiálnunk az érvényes detektáló szűrőkre érvényes kritériumokat. Csak azok a szűrők megengedettek, amelyek nem tudnak letörölni egyetlen olyan pixelt sem, amelyek **lehetnének** az egyszerűsített szerkezet vázvonalaának részei (azaz a detektáló szűrő cella nem lehet a szűrő szélén, illetve csak olyan pixelek törölhetőek, amelyek tömör pixelek olyan csoportjait kötik össze, melyek a detektáló szűrő cellát nem tartalmazó úton is kapcsolódnak egymáshoz. Ezeket a megfogalmazásokat összefoglalva az érvényes detektáló szűrőknek az alábbi feltételeket kell kielégíteniük:

- Ha $\hat{f}\langle i, j, k \rangle = -1$, akkor $i \neq 1, i \neq r_x, j \neq 1$ és $j \neq r_y$, illetve $k \neq 1, k \neq r_z$
- Bármely két cellára, amelyek kielégítik, hogy $\hat{f}\langle i_1, j_1, k_1 \rangle = \hat{f}\langle i_2, j_2, k_2 \rangle = 1$, és $\hat{C}\langle i_1, j_1, k_1 \rangle \langle i, j, k \rangle = 1, \hat{C}\langle i, j, k \rangle \langle i_2, j_2, k_2 \rangle = 1$, ahol $\hat{f}\langle i, j, k \rangle = -1$; a $\hat{C}\langle i_1, j_1, k_1 \rangle \langle i_2, j_2, k_2 \rangle = 1$ állításnak is teljesülnie kell, ha a detektáló szűrő cellát figyelmen kívül hagyjuk.

Ezt az állapotot az *érvényességi operátor* 1-es értéke fogja jelezni $\check{V}[\text{filtername}] = 1$ (egyébként 0 értéket ad vissza). Ez a szükséges és elégséges feltétel ahhoz, hogy **ne csökkentsük** a *csomópontok* és *rudak* számát a topológiában.

Ha megpróbáljuk meghatározni az összes lehetséges *érvényes detektáló szűrőt*, akkor egy erősen korlátozott tagszámú csoportot kapunk – amelyek forgatással vagy tükrözéssel nem transzformálhatóak egymásba-, ezt a csoportot fogjuk *bitminta-primitíveknek* nevezni.

A *bitminta-primitívek* alkalmazása egy 3D-s szerkezet esetében azonban jóval nagyobb kihívást jelent, mivel a 2D-hez hasonló megközelítés alkalmazása – azaz minden lehetséges szűrő elnevezése hatalmas munkát és precizitást igényelne, másrészt a *szűrő szekvenciák* túl hosszúak lennének, és az algoritmus is lassú lenne. Ezért egy teljesen eltérő megközelítést fogunk alkalmazni, de a **3. ábrán** javasolt folyamatábra változatlan marad (annyi kiegészítéssel, hogy az előszűrés $\check{J}$ $\check{B}$ -re módosul).

Az alábbiakban a 3x3x3-as, *egyfókuszú szimmetrikus szűrők* használatát fogjuk bemutatni (de használhatnánk nyugodtan magasabb rendű *szűrőket* is). Ezen *szűrők* esetében a *tömör cellák* lehetséges száma a $0 \div (3^3 - 1) = 0 - 26$ tartományban helyezkedik el. Ebben az esetben a *szűrők* definíciója változatlan maradt, azonban a *szűrő szekvenciákban* az egyes *bitminta-primitívek* megnevezése helyett inkább **bitminta primitív csoportot** fogunk megnevezni. Míg egyes *szűrőket* a $[\text{filter}]$ -el fogunk jelölni, addig egy teljes csoport jelölése a $\{\text{bitpattern} - \text{primitive group}\}$ alakú lesz. Minden egyes *bitminta primitív csoportot* egy vagy néhány karakterrel (specifikus tulajdonságok megjelölésére szolgáló markerek), és egy, a *szűrőben* található *tömör cellák* számát megadó karakterek kombinációjával fogunk jelölni. Továbbá, mivel nem akarunk kiszámolni és tárolni minden *szűrőt* egyesével, az *alap voxelmezőn* történő tesztelést is megváltoztatjuk. A *voxelmező* minden 3x3x3-as térfogatára létrehozuk azt a *szűrőt*, amit *lehetne* aktiválni azon. Ezután pedig egyszerűen leteszteljük, hogy teljesíti-e az *érvényességi operátor* által meghatározott feltételt, illetve a markerek által előírt tulajdonságokra vonatkozó specifikus tulajdonságokat. Amennyiben minden feltétel teljesül, akkor a *szűrő* ténylegesen aktiválásra is kerül. :

- N – semmilyen feltétel (**no criteria**)
- I^3 - bemetszés megtiltva (“incut” excluded) – ha $\hat{f}\langle i, j, m \rangle \neq 0$ teljesül $\langle 2,2,1 \rangle, \langle 2,2,2 \rangle, \langle 2,2,3 \rangle$, vagy $\langle 2,1,2 \rangle, \langle 2,2,2 \rangle, \langle 2,3,2 \rangle$, vagy $\langle 1,2,2 \rangle, \langle 2,2,2 \rangle, \langle 3,2,2 \rangle$ esetén, akkor egy *szűrő* nem tartozik ebbe a csoportba.
- L – alacsony bemetszés – (“Low-incut” included) – ha $\hat{f}\langle i, j, m \rangle \neq 0$ csak egy esetben teljesül a $\langle 2,2,1 \rangle, \langle 2,2,2 \rangle, \langle 2,2,3 \rangle$, vagy $\langle 2,1,2 \rangle, \langle 2,2,2 \rangle, \langle 2,3,2 \rangle$, vagy $\langle 1,2,2 \rangle, \langle 2,2,2 \rangle, \langle 3,2,2 \rangle$ indexek közül.

A fenti jelöléseket kiterjesztve a 2D-re is, belátható, hogy $[L] = \{I4\}$; $[N] = \{N0\}$; $\{[T], [O]\} = \{I3\}$; $[U] = \{L7\}$; $[C] \subset \{N5\}$; $[I] \subset \{N1\}$; $[P] = \{I2\}$ és $[V] = \{I5\}$.

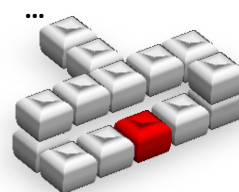
A *szekvencia operátorok* definíciója változatlan marad 3D-ben is, azzal a kiegészítéssel, hogy az *F-ciklus* esetében a keresési irányok $+x + y + z$, $+x - y + z$, $-x + y + z$, $-x - y + z$, $+x + y - z$, $+x - y - z$, $-x + y - z$, $-x - y - z$ struktúrált irányokra módosulnak.

³ Megjegyzés: az itt használt “I” egy bitminta primitív csoportot jelöl. Azért került bevezetésre az egyes *szűrők* $[I]$ -vel, illetve a teljes csoportok $\{I\}$ -vel történő jelölése, hogy a két koncepció ne legyen összekeverhető.

Sajnos amennyiben az eddigiekben ismertetett implementációt beprogramozzuk és kipróbáljuk, a végeredmény rengeteg apró hibát, kicsiny, zárt hurkokat, és egyéb zavaró jelenségeket fog tartalmazni, mivel az algoritmusnak a 2D-s esethez képest kisebb irányítása van a teljes folyamat felett. Ez abból – a már korábban is említésre került – tényből származik, hogy az alkalmazott *értékyességi operátor* csak a rudak számának csökkentését akadályozza meg, a növelését nem. Mivel a *szűrők* elnevezése nem egy megoldható megközelítés, így meg kell fogalmaznunk annak is a szükséges és elégséges feltételét, hogy egy *szűrő* ne növelhesse a rudak és *csomópontok* számát. Az alábbi definíciók szükségesek ehhez:

Konzisztens ITD: Egy olyan *ITD*, amelyben nem léteznek olyan élek vagy csúcsok, amelyek csak két olyan *voxel* cellához tartoznának, amelyek nem *szomszédos cellák*.

A kezdeti lépésben (feltételezve, hogy megfelelő módon alkalmaztuk a simítást), ez a feltétel teljesülni fog. A konzisztens *ITD* szükségességének feltétele a *szomszédosság* definíciójából ered, mivel azt a lapon való érintkezésként definiáltuk. Bár elsőre nem evidens, de tesztek segítségével belátható, hogy ez tényleg szükséges feltétel az *ITD* megfelelő állapotának fenntartásához.



21. ábra
Helyi hibák

Amennyiben a konzisztenciát nem biztosítjuk az egymást követő lépésekben, akkor olyan helyi hibák és hurkok jelennek meg a szerkezetben, amelyek a továbbiak nem törölhetőek le, pl. a pirossal jelölt cellák a **21. ábrán** nem törölhetőek 3x3x3-as szűrők segítségével (mivel a részlet mérete sokkal nagyobb, mint a szűrő mérete). Ez egyben arra is rávilágít, hogy miért stabilabb a *szomszédosság* alkalmazott definíciója. A jelenlegi definíciónál egy *direkt kapcsolathoz* csak két *voxel cella* tartozik. Amennyiben közös lap helyett közös él vagy közös csúcs lett volna a definíció, akkor 4 illetve 8 cella tartozna egyetlen *direkt kapcsolathoz*, ez pedig jelentős határozatlanságot okozna.

A feltétel a szűrőkre a **konzisztens értékyességi operátor**: $\widetilde{V}^c[\text{filtername}]$ segítségével alkalmazható, amely 1-es értékkel fogja jelezni, amennyiben a szűrő nem tud olyan csúcsot vagy élt létrehozni, ami csak két nem szomszédos *voxel cellához* tartozna. Azokat a *szűrőket*, amelyek kielégítik ezt a feltételt, **konzisztens szűrőknek** fogjuk nevezni. Matematikailag megfogalmazva:

- $\widetilde{V}^c[\text{filtername}] = 1$ akkor és csakis akkor, ha a szűrő alkalmazása után,
 - o bármely két cellára, melyek kielégítik, hogy $\hat{f}\langle i_1, j_1, k_1 \rangle = \hat{f}\langle i_2, j_2, k_2 \rangle = 1$, és $(i_1 - i_2)^2 + (j_1 - j_2)^2 + (k_1 - k_2)^2 = 2$, (éleken csatlakoznak), léteznie kell olyan i_3, j_3, k_3 indexeknek, melyekre igaz, hogy $\hat{f}\langle i_3, j_3, k_3 \rangle = 1$ és $\hat{N}\langle i_1, j_1, k_1 \rangle \langle i_3, j_3, k_3 \rangle = \hat{N}\langle i_2, j_2, k_2 \rangle \langle i_3, j_3, k_3 \rangle = 1$ (azaz létezik egy olyan *voxel cella*, mely közvetlenül kapcsolódik mindkét cellához, vagyis az élen három *voxel cella* osztozik).
 - o nem létezik két olyan cella $\langle i_1, j_1, k_1 \rangle, \langle i_2, j_2, k_2 \rangle$, melyekre igaz lenne, hogy, $\hat{f}\langle i_1, j_1, k_1 \rangle = \hat{f}\langle i_2, j_2, k_2 \rangle = 1$, és $|i_1 - i_2| * |j_1 - j_2| * |k_1 - k_2| = 1$, $|i_1 - i_2| + |j_1 - j_2| + |k_1 - k_2| = 3$ (csúcsokon csatlakoznak), de nem létezik olyan i_3, j_3, k_3 index, melyre $\hat{f}\langle i_3, j_3, k_3 \rangle = 1$ és $\hat{N}\langle i_1, j_1, k_1 \rangle \langle i_3, j_3, k_3 \rangle = 1$ vagy $\hat{N}\langle i_2, j_2, k_2 \rangle \langle i_3, j_3, k_3 \rangle = 1$.

Amennyiben kiegészítjük a programot a fenti definícióval, kiderül, hogy bár a szerkezet immáron *konzisztens* marad végig, de a *csomópontok* száma még mindig nőhet. Ennek elkerülése végett bevezetésre kerül az *inverz szerkezet* fogalma. Amikor a szerkezeten a szűrést hajtjuk végre, valójában egy másik – rejtett – szerkezet is létezik, amely a mi szerkezetünk inverze (azaz az *üres cellákból* áll: $\hat{f}(i, j, k) = 0$). Ha megvizsgáljuk ezt a szerkezetet, nyilvánvalóvá válik, hogy az inverz szerkezet topológiáját is sértetlenül kell hagynunk, akárcsak a *konzisztenciáját* (de ennek a szerkezetnek a mérete nem csökken, hanem pont nő a szűrés folyamán). Azokat a *szűrőket*, amelyek kielégítik azt a feltételt, hogy az inverzük is *konzisztens* marad, **inverzen konzisztens szűrőknek** nevezzük, és ezt az állapotot az **inverzen konzisztens érvényességi operátor**: $\widetilde{V}^C[\text{filtername}] = 1$ jelöli.

A matematikai megfogalmazás hasonló a *konzisztens érvényességi operátoréhoz*, csak most az inverz szerkezetre kell alkalmazni.

Egy utolsó feltétel még továbbra is hiányzik ahhoz, hogy teljesen pontos legyen az eljárás. Az előző operátorok már garantálták a *konzisztenciát*, és azt, hogy a *csomópontok* száma nem csökkenhet. De a növekedés még nem kizárt továbbra sem, bár igaz, hogy kevesebb esetben fordulhat elő. Belátható, hogy ha törölünk egy elemet a szerkezetből, akkor egy elemet hozzáadunk annak inverzéhez. Amennyiben ez az új elem két, korábban egymással nem kapcsolódó részét köti össze az inverz szerkezetnek, akkor abban egy új rúd jött létre, azaz a topológia sérült. Ez utóbbi feltételt kielégítő szűrőket fogjuk **inverzen érvényes szűrőknek** nevezni, melyeknek ki kell elégíteniük az **inverz érvényességi operátort**: $\widetilde{V}^I[\text{filtername}] = 1$, melyet megfogalmazva:

$\widetilde{V}^I[\text{filtername}] = 0$ bármely olyan *szűrőre*, amely két egymáshoz nem kapcsolódó üres régiót összekötne az aktiválás után.

Az előző definíciókat felhasználva állíthatjuk tehát, hogy a szűrési folyamat során kizárólag *érvényes*, *inverzen érvényes*, *konzisztens*, *inverzen konzisztens szűrőket* alkalmazhatunk. Ezzel párhuzamosan pedig csökken a megfelelő *szűrő-szekvenciák* kiválasztásának jelentősége is. A tesztek folyamán a $\widetilde{B} \widetilde{S} \widetilde{M}\{11 - 26\} \widetilde{D} \widetilde{M}\{14 - 13\} \widetilde{S} \widetilde{M}\{11 - 26\}$ és $\widetilde{B} \widetilde{S} \widetilde{M}\{11 - 14\}$ egyszerű, de nagyon gyors *szűrő-szekvenciák* kerültek alkalmazásra. A pontos matematikai megfogalmazásnak hála ugyanis a *szűrő-szekvenciák* már nem a végeredmény korrektségét határozzák meg, hanem az algoritmus sebességét, és azt, hogy a végeredmény milyen közel fog elhelyezkedni a tényleges középvonalhoz.

Bár a 3D-s megközelítés ismertetése után megkérdőjelezhető válik, hogy a 2D-s megoldás miért nem ugyanezeket az elveket követi, de a válasz meglepően egyszerű. Amennyiben megvizsgáljuk a definiált szűrőket, akkor észrevehető, hogy azok mind kielégítik ezeket a feltételeket is, azaz a módszer implicit módon tartalmazza. Azáltal, hogy explicit módon beleépítenénk ezeket a feltételeket a 2D-s módszerbe is, az eljárás sebessége növekedne, de csökkenne az afölötti irányításunk (pl. a 2D-s eljárás képes bordák detektálására is megfelelő szűrő szekvencia esetén), míg a 3D-s megközelítés inkább az *evidens csomópontok*, ill. peremfeltételek és *evidens csomópontok* közötti rudakra fókuszál – ami a legtöbb mérnöki szerkezet esetében elegendő.

Poszt-processzálas 3D-ben:

Gömbhal operátor 3D-ben: A gömbhal operátor ebben az esetben a gömbhal térfogatát fogja megadni, azaz matematikailag megfogalmazva,

$$\hat{P}[l, m, n] = \max(4r^3\pi/3), \text{ az összes } r \text{ értékre, amelyek kielégítik, hogy}$$
$$\hat{f}[l + r' \sin \theta \cos \varphi, m + r' \sin \theta \sin \varphi, n + r' \cos \theta] = 1$$
$$\text{minden } 0 \leq r' \leq r, 0 \leq \varphi \leq 2\pi, 0 \leq \theta \leq \pi \text{ értékekre}$$

Lokális csomóponti pozíció optimalálás (LLPO - Local point position optimization): Az optimalálás alapötlete megegyezik a 2D-s esetével, azonban a matematikai megfogalmazást kis mértékben korrigálni kell:

$$\text{Maximáljuk: } \hat{P} \left[p_{i,x} + d_x, p_{i,y} + d_y, p_{i,z} + d_z \right] \rightarrow \max$$

$$\text{Úgy, hogy } \theta_j \leq \alpha_{\max}, \text{ where } \theta_j = \arccos \frac{(d_x e_x + d_y e_y + d_z e_z) \cdot (p_i - p_j)}{|d_x e_x + d_y e_y + d_z e_z| \cdot |p_i - p_j|} \text{ ha } t_{ij} = 1.$$

(A lehetséges mozgás iránya és a csomópontot egy másik csomóponttal összekötő irány által bezárt szög).

$$\text{ahol, } 0 \leq d_x, d_y, d_z \leq D_{\max}$$

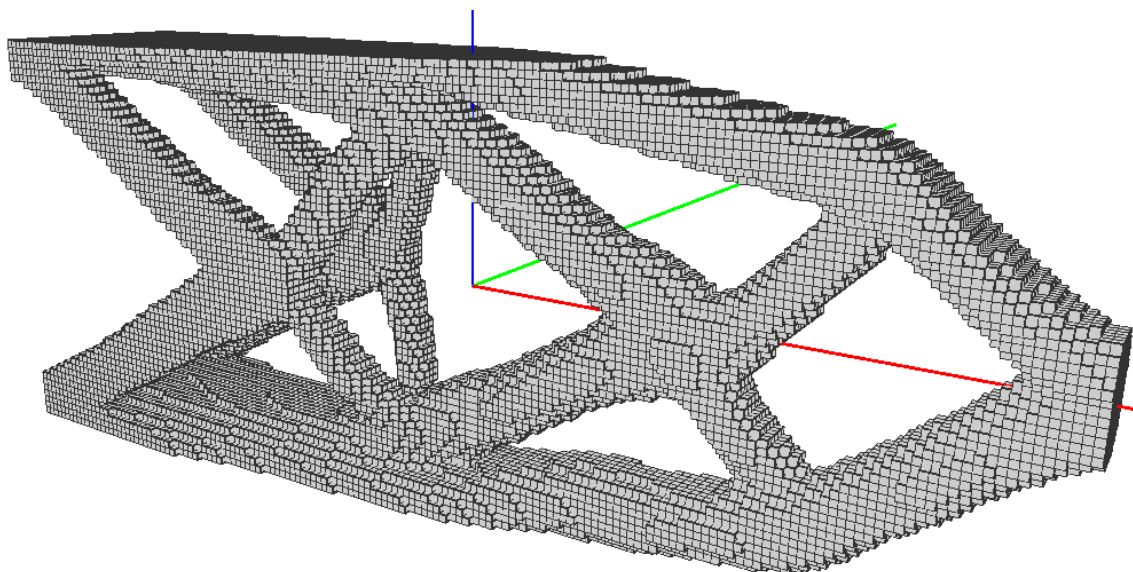
Az új pozíció pedig $p_i^{\text{new}} = p_i^{\text{old}} + d_x e_x + d_y e_y + d_z e_z$.

Az LLPO eljárásnál továbbra is meg kell említeni – akárcsak a 2D-s esetről – hogy ahogy egyre több tapasztalat gyűlik fel az algoritmus alkalmazásával, előfordulhat, hogy további feltételek beiktatása válik szükségessé. 3D esetében az LLPO eljárás egy javított alakjának létrehozása még jobban sürgető kérdés, mivel több olyan eset képzelhető el, ahol az eljárás működésképtelenné válna (pl. két vékony lemezt összekapcsoló csomópont esetében).

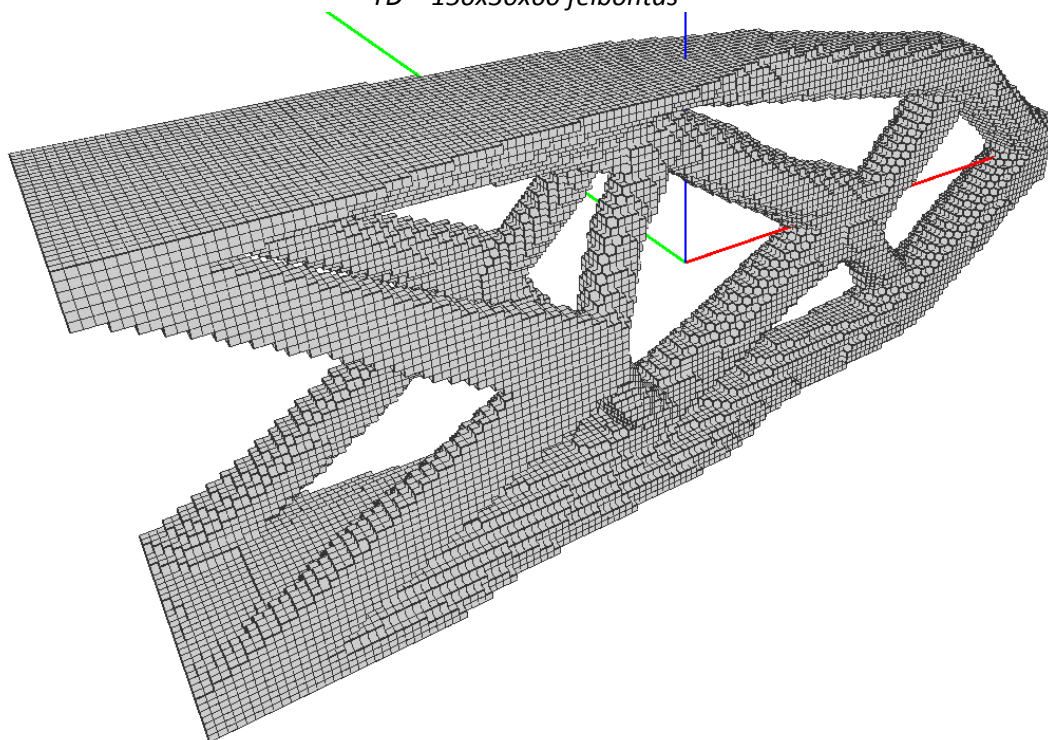
Az algoritmus további részei változatlanok maradnak 3D esetében is. A *pre-processzálasra* egy példa található az alábbiakban, az eredeti TD-vel a **22.**, illetve az RTD-vel a **23. ábrán**, illetve egy második példa a **24. ábrán**.

A BPPF szűrés eredménye (vörös/színes), az eredeti RTD-vel együtt (világoskék), valamint a detektált TCM-mel együtt (zöld golyók) a **25.**, **26.**, **27.** és **28. ábrán** találhatóak. Az eredeti peremfeltétel (teljes fal) itt is manuálisan a négy sarokra lett megváltoztatva.

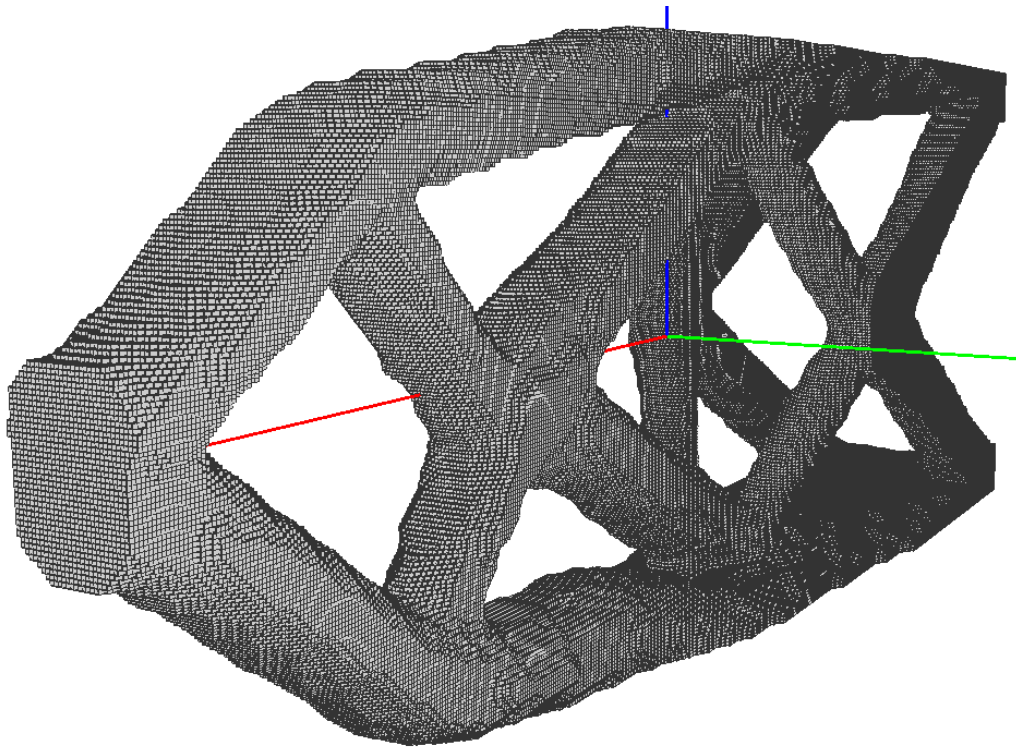
Egy érdekes tapasztalat az algoritmussal kapcsolatban, hogy bár a háló finomítása, simítása, illetve a gömbhal operátor függvénymező kiszámítása több időt vesz igénybe a 2D-hez képest, de a BPPF részéről az algoritmus nem jelentősen lassabb a 2D-s megközelítéshez képest, elsősorban az eltérő megközelítésnek hála.



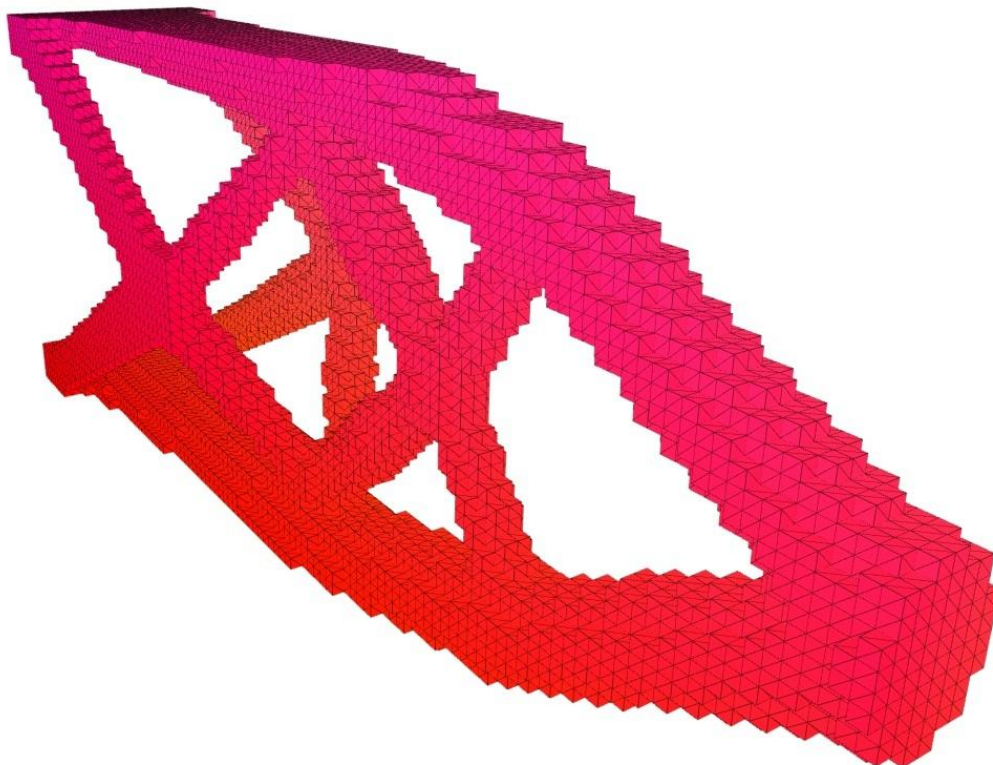
22./a ábra
TD – 150x30x60 felbontás



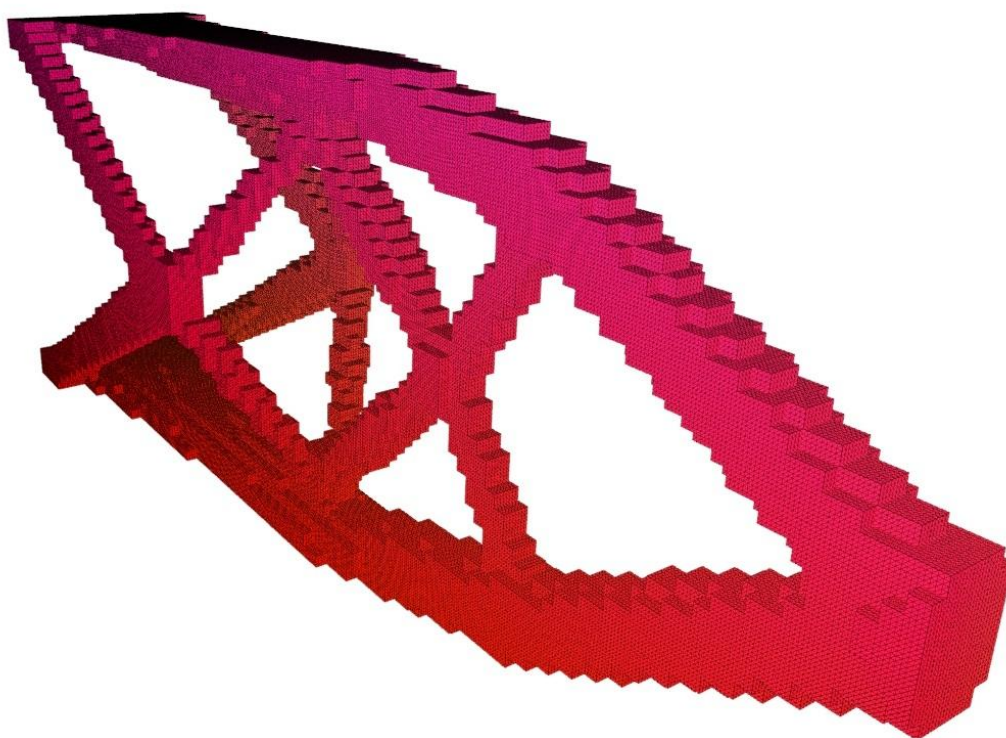
22./b ábra
TD – 150x30x60 felbontás



23. ábra
RTD (Refined Topology Design) – 450x90x180 felbontás

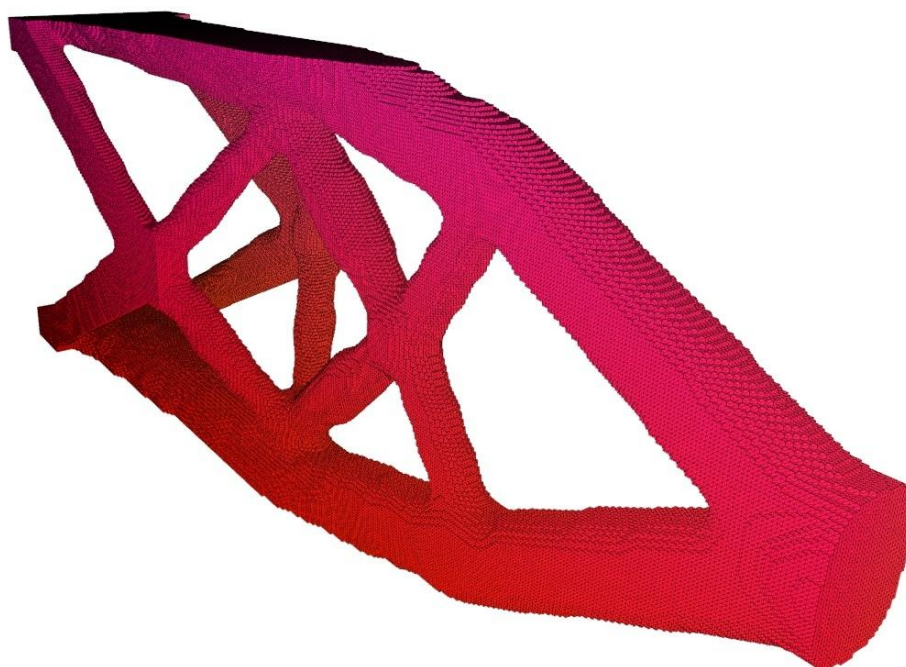


24./a ábra
TD – 100x20x40 felbontás



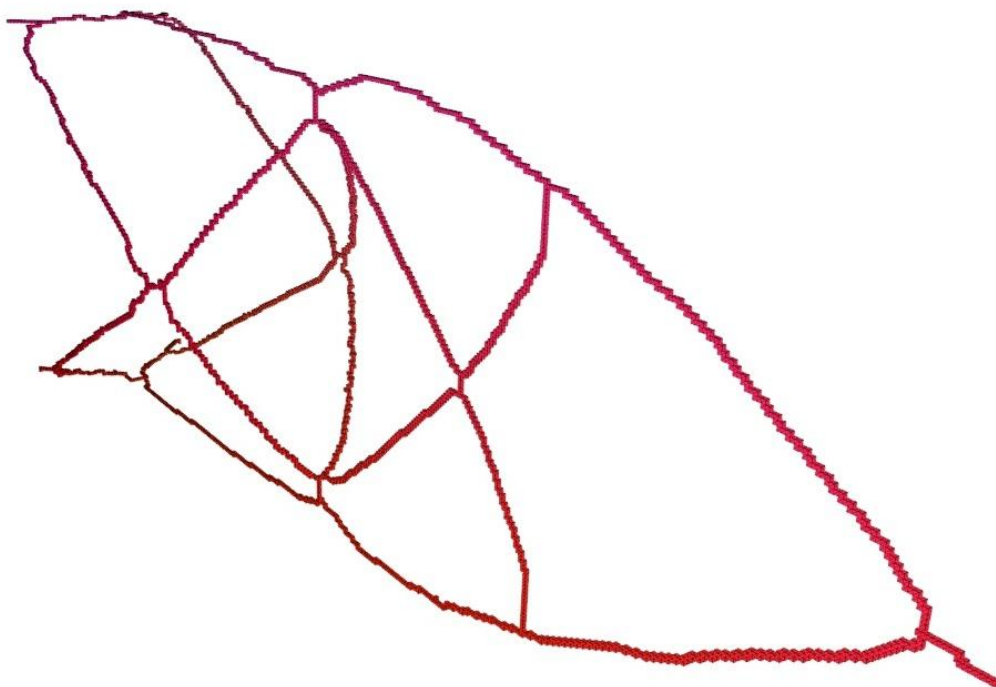
24./b ábra

RTD (simítás és előszűrés nélkül) – 400x80x160 felbontás



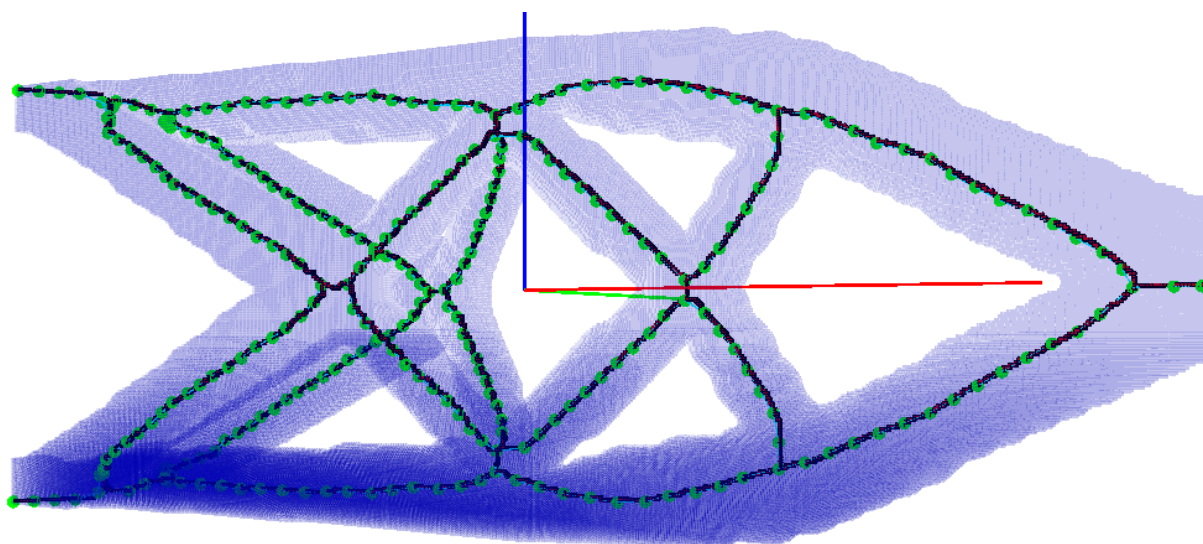
24./c ábra

RTD – 400x80x160 felbontás



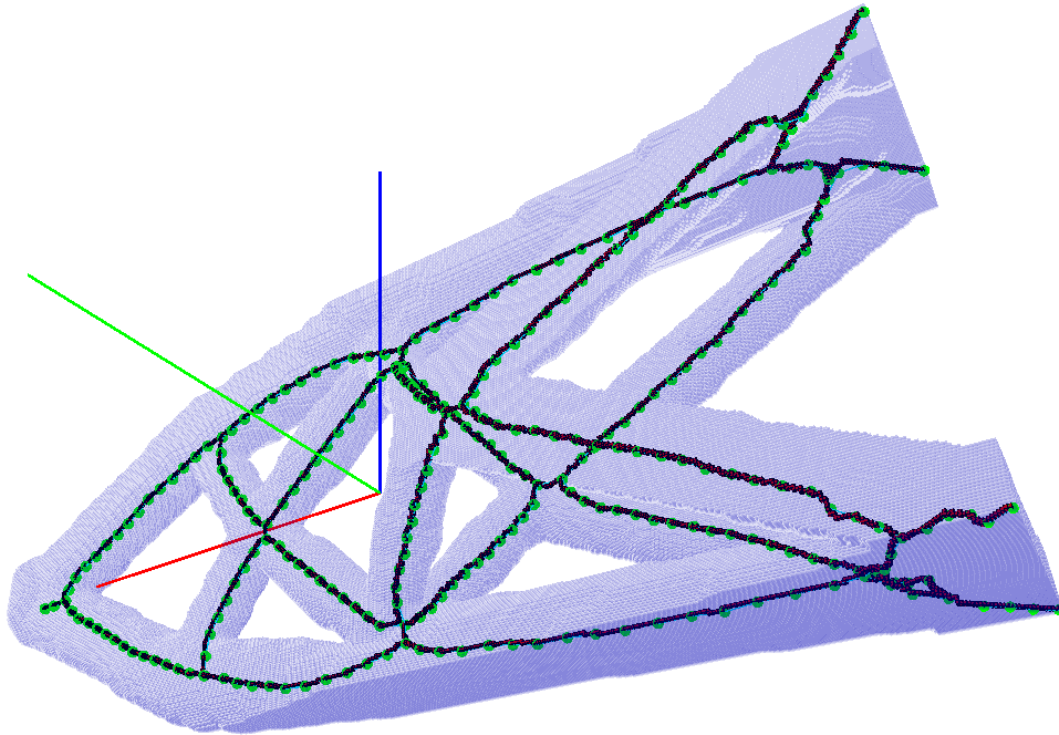
24./d ábra

TCM

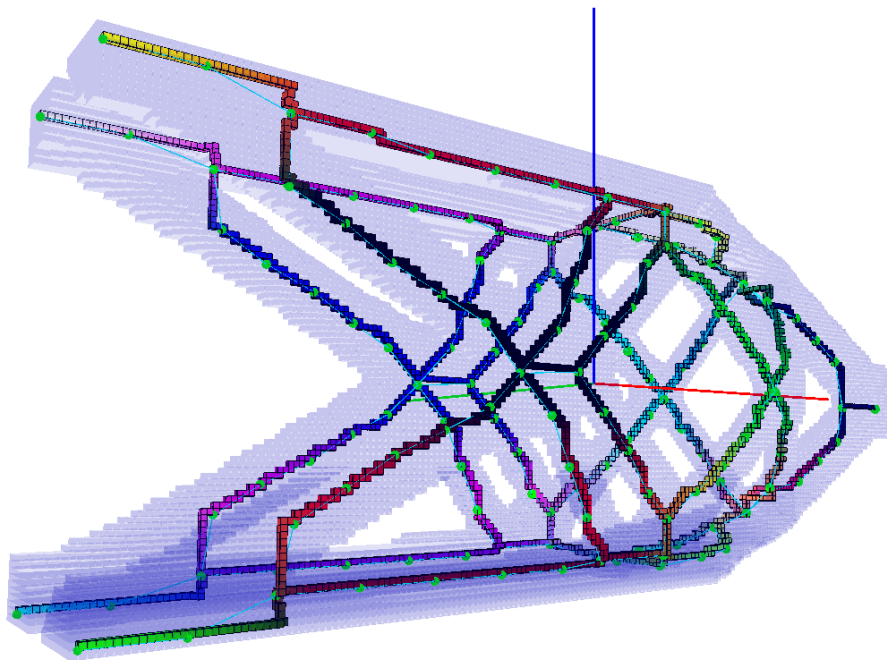


25./a ábra

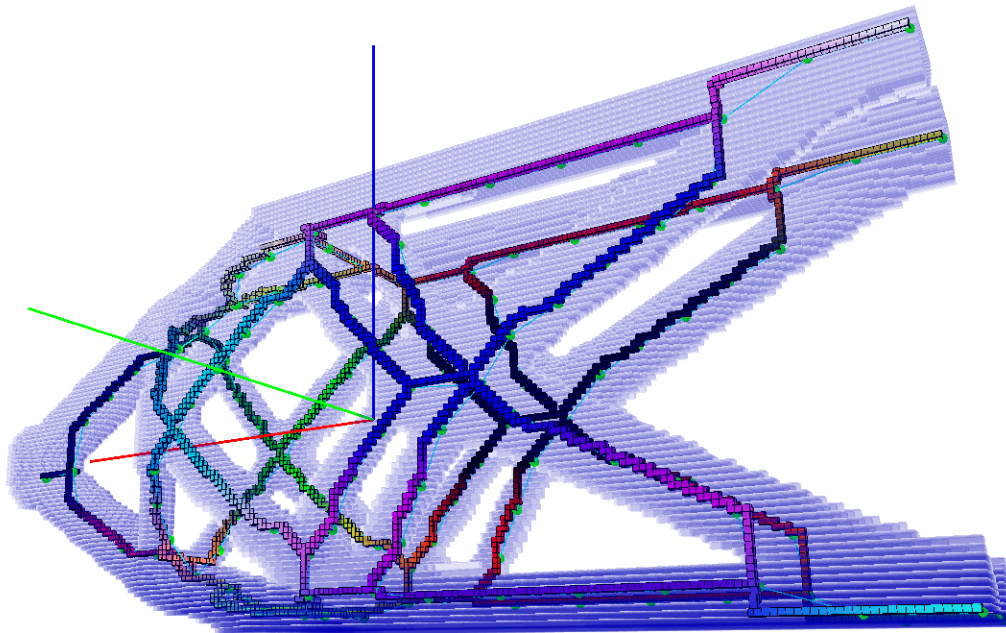
RTD (Refined Topology Design) – 500x100x200 felbontás
Szűrő-szekvencia: $\tilde{B} \tilde{S} \tilde{M} \{N1 - 14\}$



25./b ábra
RTD (Refined Topology Design) – 500x100x200 felbontás
Szűrő szekvencia: $\tilde{J} \tilde{B} \tilde{S} \tilde{M} \{N1 - 14\}$



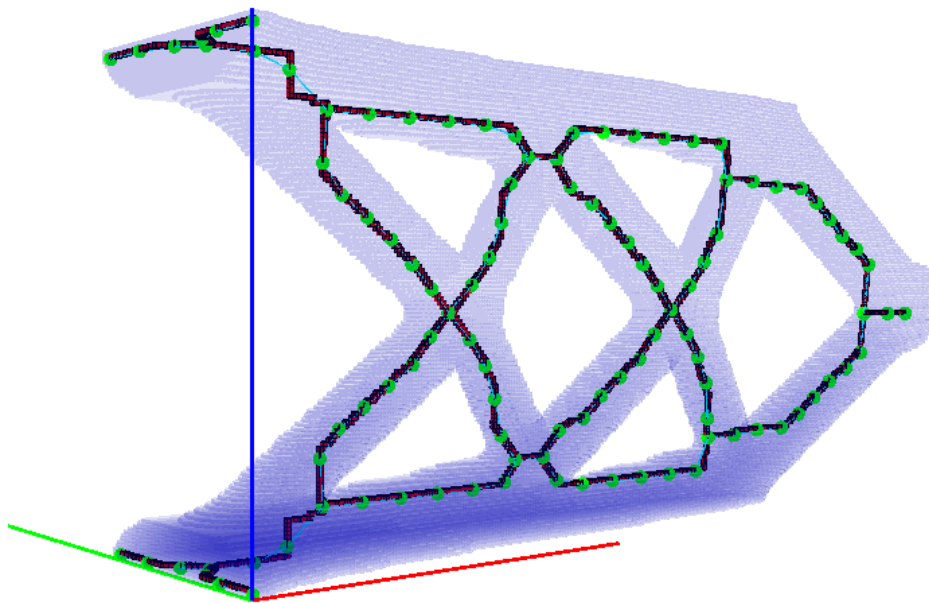
26. ábra/a
RTD (Refined Topology Design) – (eltérő optimalizációs eljárás – közbelső eredmény) – 200x40x80 felbontás
Szűrő szekvencia: $\tilde{J} \tilde{B} \tilde{S} \tilde{M} \{I1 - 26\} \tilde{D} \tilde{M} \{L4 - 13\} \tilde{S} \tilde{M} \{I1 - 26\}$



26. ábra/b

RTD (Refined Topology Design) –(eltérő optimalási eljárás – közbelső eredmény) – 200x40x80 felbontás

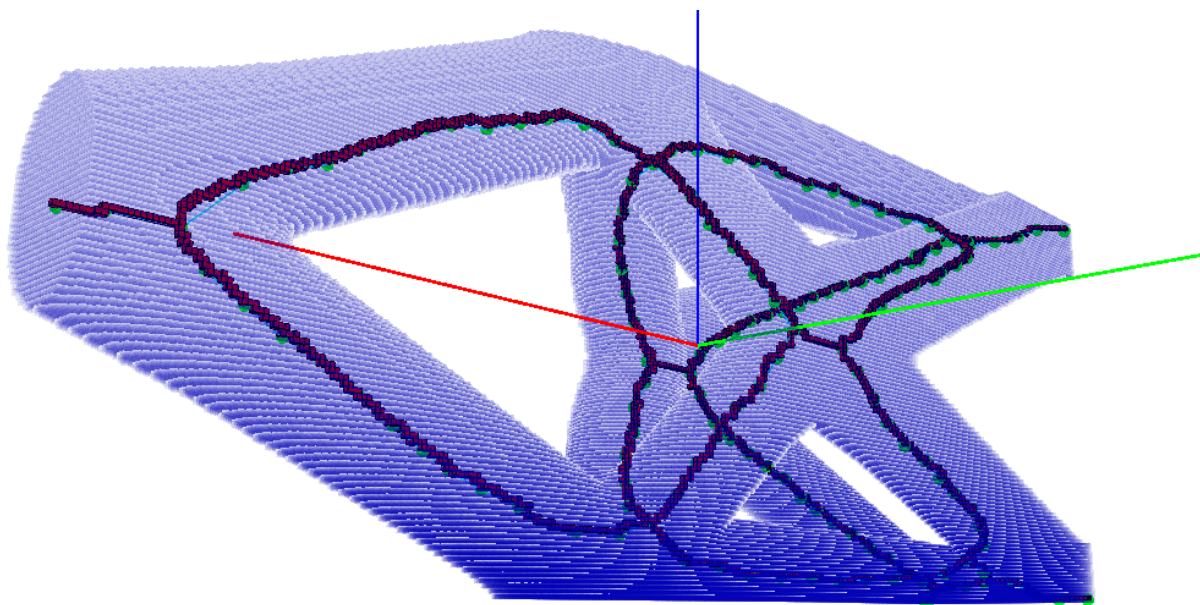
Szűrő szekvencia: $\tilde{J} \tilde{B} \tilde{S}\tilde{M}\{I1 - 26\} \tilde{D}\tilde{M}\{L4 - 13\}\tilde{S}\tilde{M}\{I1 - 26\}$



27. ábra

RTD (Refined Topology Design) –(eltérő optimalási eljárás) – 200x40x80 felbontás

Szűrő szekvencia: $\tilde{J} \tilde{B} \tilde{S}\tilde{M}\{I1 - 26\} \tilde{D}\tilde{M}\{L4 - 13\}\tilde{S}\tilde{M}\{I1 - 26\}$



28. ábra
 RTD (Refined Topology Design) – 300x60x120 felbontás
 Szűrő szekvencia: $\tilde{J} \tilde{B} \tilde{M}\{N1 - 14\}$

6. Összefoglalás

A cikkben javasolt eljárás nemcsak gyors és megbízható, de viszonylag egyszerűen implementálható is.

Az algoritmus a tesztek során az eredeti *TD* méretétől függően 1-60 másodperc alatt elkészült a számítással (2D) egy Intel® Core i5™ 760 processzor egyetlen magján (3.2 GHz).

Megfigyelhető az is, hogy a *BPPF* hasonlít a rajintelligenciára sok szempontból tekintve. Nagyszámú primitívet „eresztünk” rá a szerkezetre, amelyek mintegy „lerágják” a szerkezet felesleges részeit, csak a „csontvázat” hagyva hátra. Bár az egyes elemek intelligenciája és összetettsége erősen kötött ($\sim 3 \times 3$, ill. $3 \times 3 \times 3$ szenzorral rendelkeznek), de a hatalmas számú egyszerű primitív együtt olyan hatást tud elérni, amelyet egy komplex matematikai algoritmusnak is nehezebb esik. Az $\tilde{F}$, $\tilde{S}$, $\tilde{D}$ szekvencia operátorok alkalmazása esetén a raj egyes tagjai teljesen önállóan dolgoznak, azonban az $\tilde{M}$ operátor esetén egy viszonylag egyszerű kommunikáció már megjelenik (mivel az egyes elemeknek várakozniuk kell minden egyes lépés esetén a raj többi elemére, ami egyfajta szinkronizációként fogható fel). Ez részlegesen magyarázatot ad arra, miért ilyen hatékony az eljárás.

Bár az algoritmus sebessége csökkenni fog a *TD* felbontásának növekedésével, azonban az algoritmus nagyon könnyen párhuzamosítható mind a CPU-n, mint a GPU-n a nagyszámú primitívnek hála. Így igény esetén egy további hatalmas sebességnövekedést lehet elérni az algoritmusnál.

Az algoritmus már a jelenlegi formájában is nagyon pontos, azonban a poszt-processzálas területén – főleg az *LLPO* eljárás, valamint a rúd detektálási eljárás területén – a detektálási pontosság növelése érdekében további fejlesztések szükségesek [12,13]. Ez a kettő azonban különösen érdekes terület, mivel pontos definíciókat nem határozhatunk meg előre a programozás szakaszában, lévén hogy a szükséges feltételek a felhasználó által megszabott kritériumoktól fog függeni. Így a fejlesztés következő szakasza a méretoptimalizációs modul, az *LLPO* egy továbbfejlesztett változatának létrehozása, valamint végül, de nem utolsósorban egy olyan adatbázis létrehozására fog irányulni, amely sok, mérnöki szempontból is jelentőséggel bíró kötés- és rúdtípus kritériumait fogja tartalmazni. Bár az alábbi munkában a *BPPF* kifejezetten a középvonalak detektálására került alkalmazásra, de a képességeinek és a lehetőségek széles tárházának hála (pl. *szűrők*, *szűrő* méret, *LLPO* paraméterek, stb.) az eljárás nagy valószínűséggel kiterjeszthető nem csak középvonal, de például lemezek, felületek, és egyéb mérnöki szempontból fontos „feature” csoportok detektálásra is, vagy akár a képfelismerés területeire is.

A **függelékben** az olvasó megtekintheti az algoritmus alkalmazásának eredményét néhány topológiai optimalizációs eredményen, melyek közül több a klasszikus topológiai feladatok közé tartozik. A tesztek folyamán a *BPPF* gyakorlatilag 100%-os megbízhatósággal üzemelt, de a rúddetektálási eljárás még számos ponton fejlesztésre szorul. Ennek az az oka, hogy egy rendkívül egyszerű – de ennek ellenére hatékony – rúd-detektálási eljárás került alkalmazásra, amely azonban a későbbiekben frissítésre fog kerülni, hogy a mérnöki gyakorlati szempontjából fontos esetekre is felkészüljön.

7. Köszönetnyilvánítás

A bemutatott kutató munka a TÁMOP-4.2.1.B-10/2/KONV-2010-0001 jelű projekt részeként az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg, valamint az Országos Tudományos Kutatási Alap OTKA T 75678 támogatásával.

Emellett külön köszönet jár Dr. Jármay Károlynak a tanulmány elkészítése során nyújtott támogatásáért.

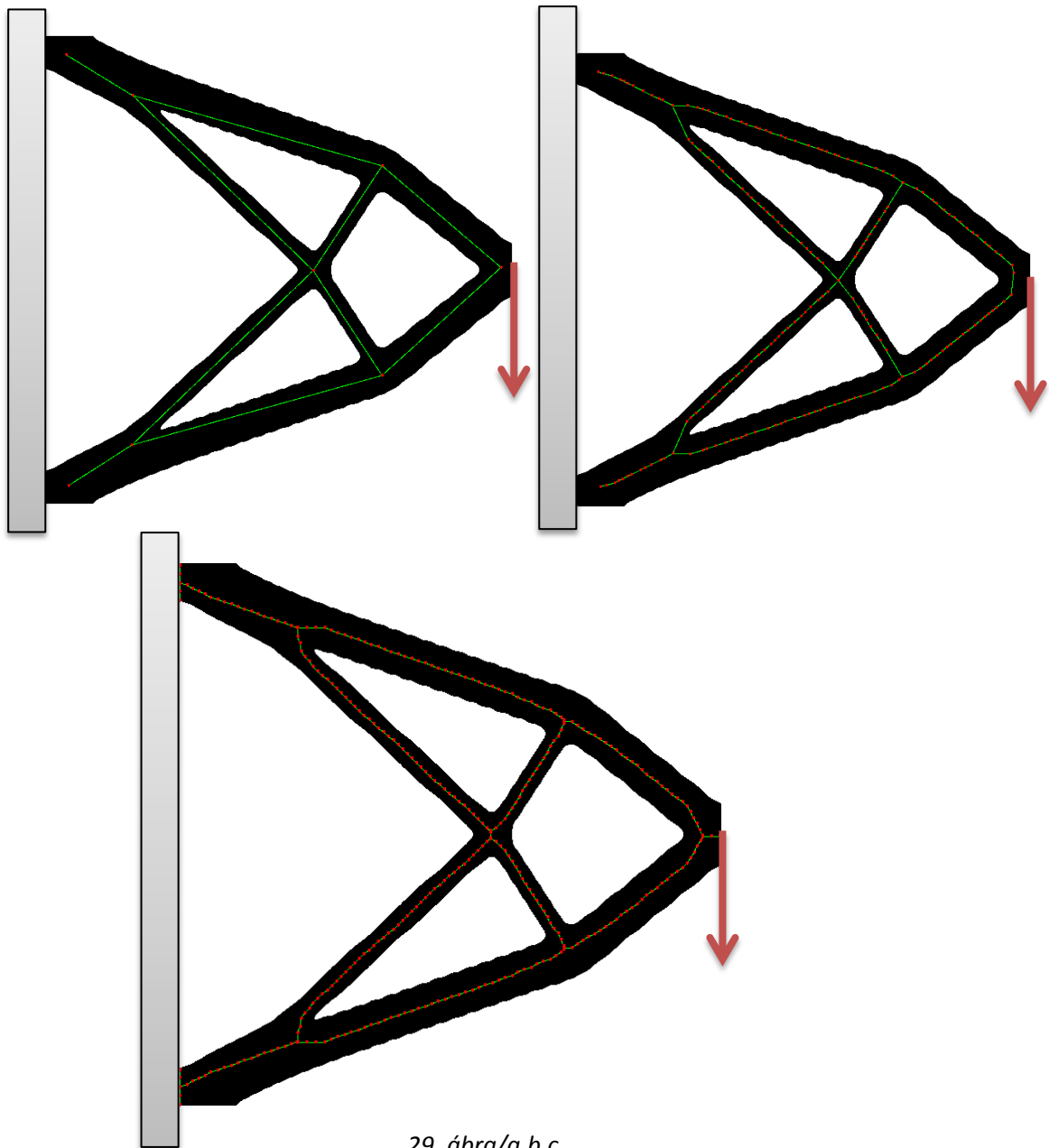
8. Irodalomjegyzék:

- [1] Bendsoe M.P. & Sigmund O.: Topology Optimization – Theory, Methods and Applications – Chapter 4. Springer, page 16-68
- [2] Bendsoe, M.P., Ben-Tal, A., and Zowe, J.: “Optimization methods for truss geometry and topology design”, Structural Optim. 7(3) – 141-158 (1994)
- [3] Jármai, K., Farkas, J. (Eds.): Mechanics and Design of Tubular Structures, Springer Verlag, 1998. 337 p. ISBN 3-211-83145-2
- [4] Farkas J. & Jármai K.: Economic design of metal structures, Millpress Science Publisher, Rotterdam, 2003, 340 p. ISBN 90 77017 99 2
- [5] Farkas, J., Jármai, K.: Design and optimization of metal structures, Horwood Publishers, Chichester, UK, 2008. 328 p. ISBN: 978-1-904275-29-9
- [6] Sigmund O. : “A 99 line topology optimization code written in Matlab”, Struct Multidisc. Optim. 21, 120-127
- [7] Zhou, M. & Rozvany, G. I. N.: The COC algorithm, part II: Topological, geometry and generalized shape optimization, Computer Methods in Applied Mechanics and Engineering 89. (1991).
- [8] Zhou, M. & Rozvany, G.I.N.: On the validity of ESO type methods in topology optimization. Struct. Multidisc. Optim. 21: 80–3. (2001).
- [9] Querin, O.M., Steven, G.P, Xie, Y.M.: "Evolutionary structural optimisation (ESO) using a bidirectional algorithm", Engineering Computations, Vol. 15, No. 8, pp.1031 – 1048. (1998)
- [10] X. Huang, Y. M. Xie: Evolutionary Topology Optimization of Continuum Structures – Methods and Applications, Wiley, 2010, page 19-23.
- [11] Sigmund, O. & Petersson, J.: “Numerical instabilities in topology optimization: A survey on procedures dealing with checkerboards, mesh-dependencies and local minima”; Structural Optimization 16(1): 68-75. (1998).
- [12] Liu, Z., Gao, Q., Zhang, P., Xuan, M. and Wu, Y.: Topology optimization of fluid channels with flow rate equality constraints, Structural and Multidisciplinary Optimization, Vol. 44. No. 1. pp. 31-37. 2011.
- [13] Rozvany, G.I.N.: Domain augmentation and reduction in structural topology optimization, Structural and Multidisciplinary Optimization, Vol. 44. No. 4. pp. 589-591, 2011.

Függelék

Teszt eredmények:

1. eset:



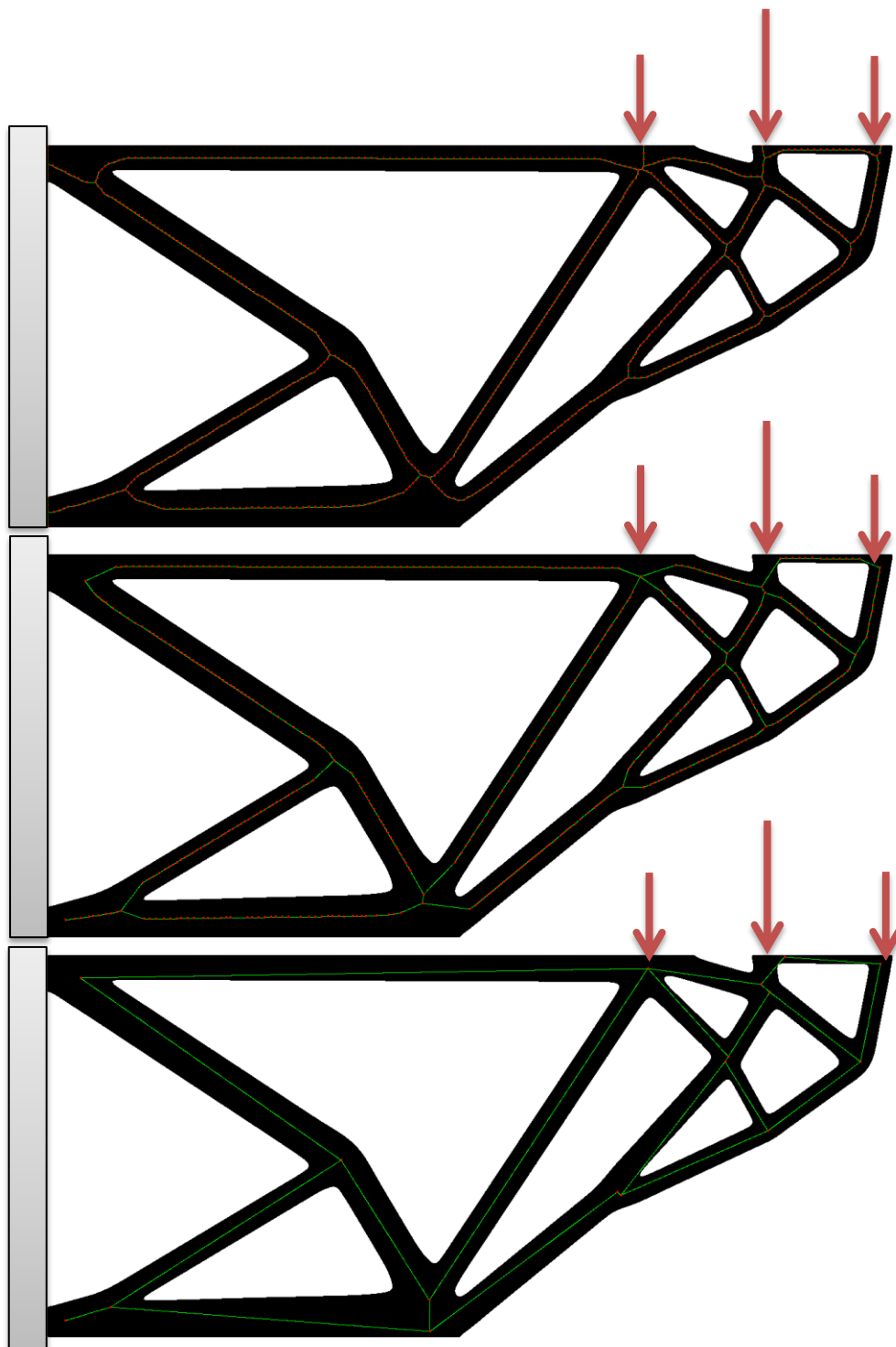
29. ábra/a,b,c

Rúd-detektálás eredménye (bal felső)

LLPO eredménye (jobb felső)

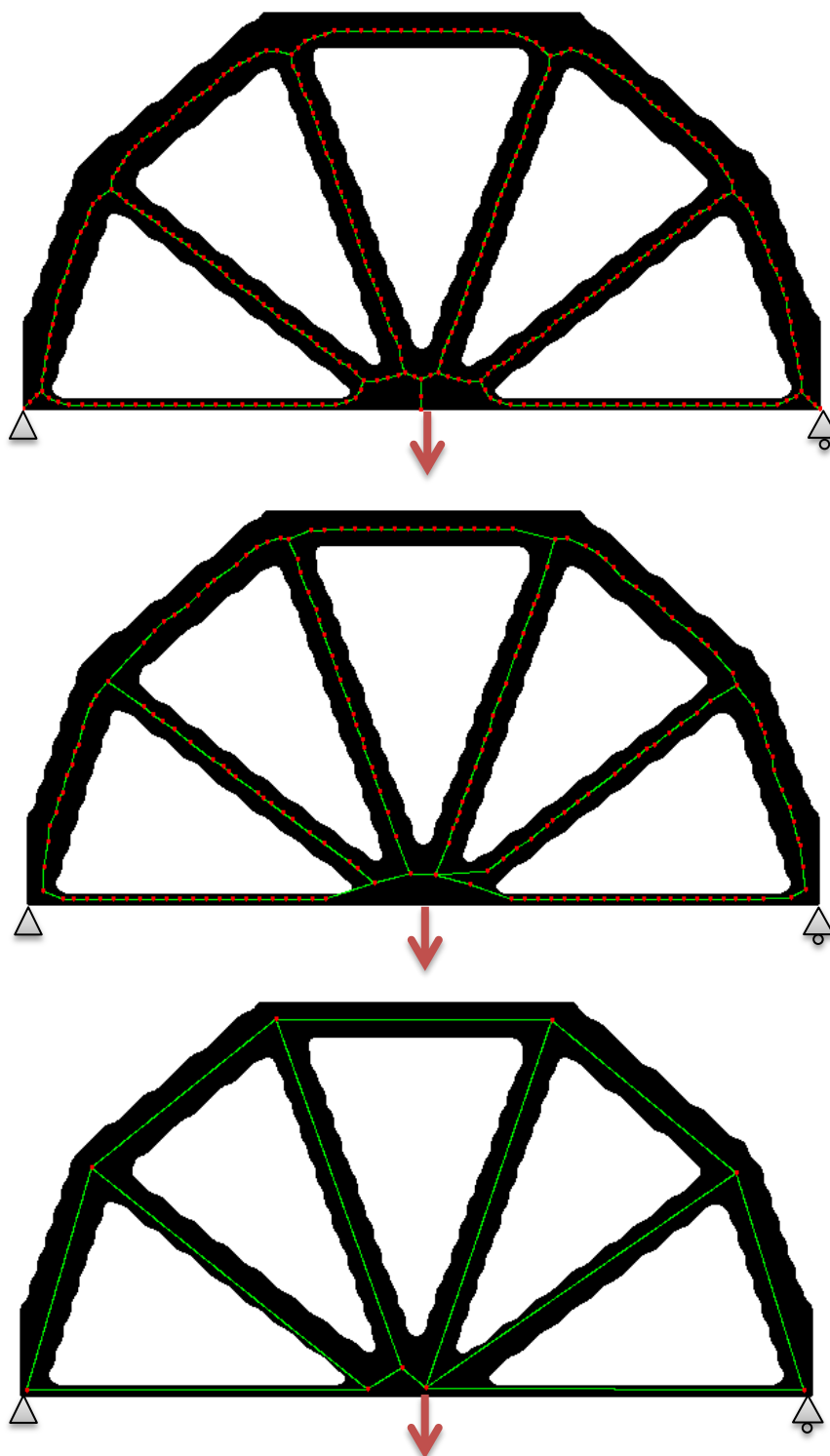
BPPF eredménye (alsó középső)

2. eset:



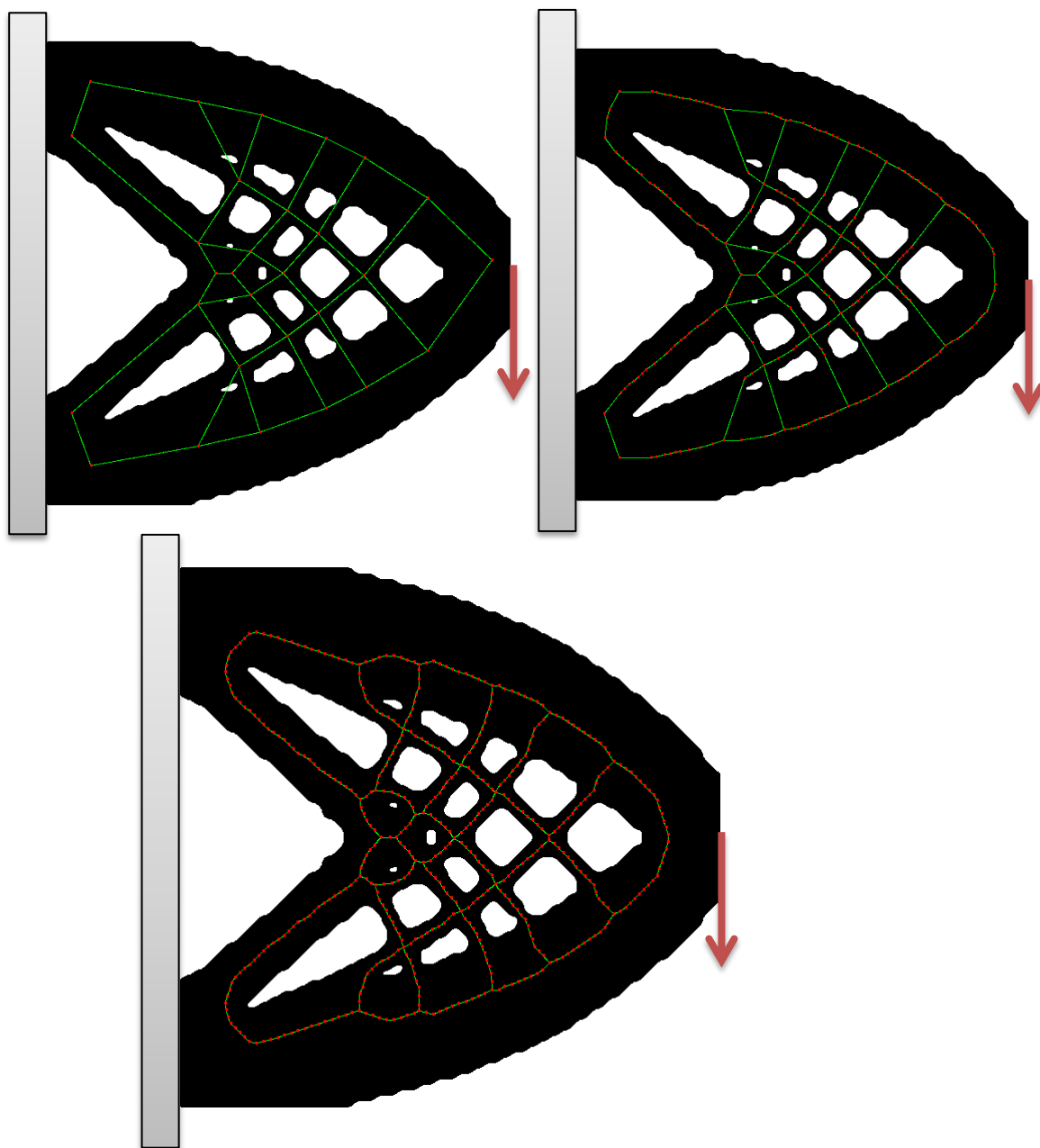
30. ábra/a,b,c
a, BPPF eredménye
b, LLPO eredménye
c, Rúd-detektálás eredménye

3. eset:



31. ábra/a,b,c
a, BPPF eredménye
b, LLPO eredménye
c, Rúd-detektálás eredménye

4. eset:



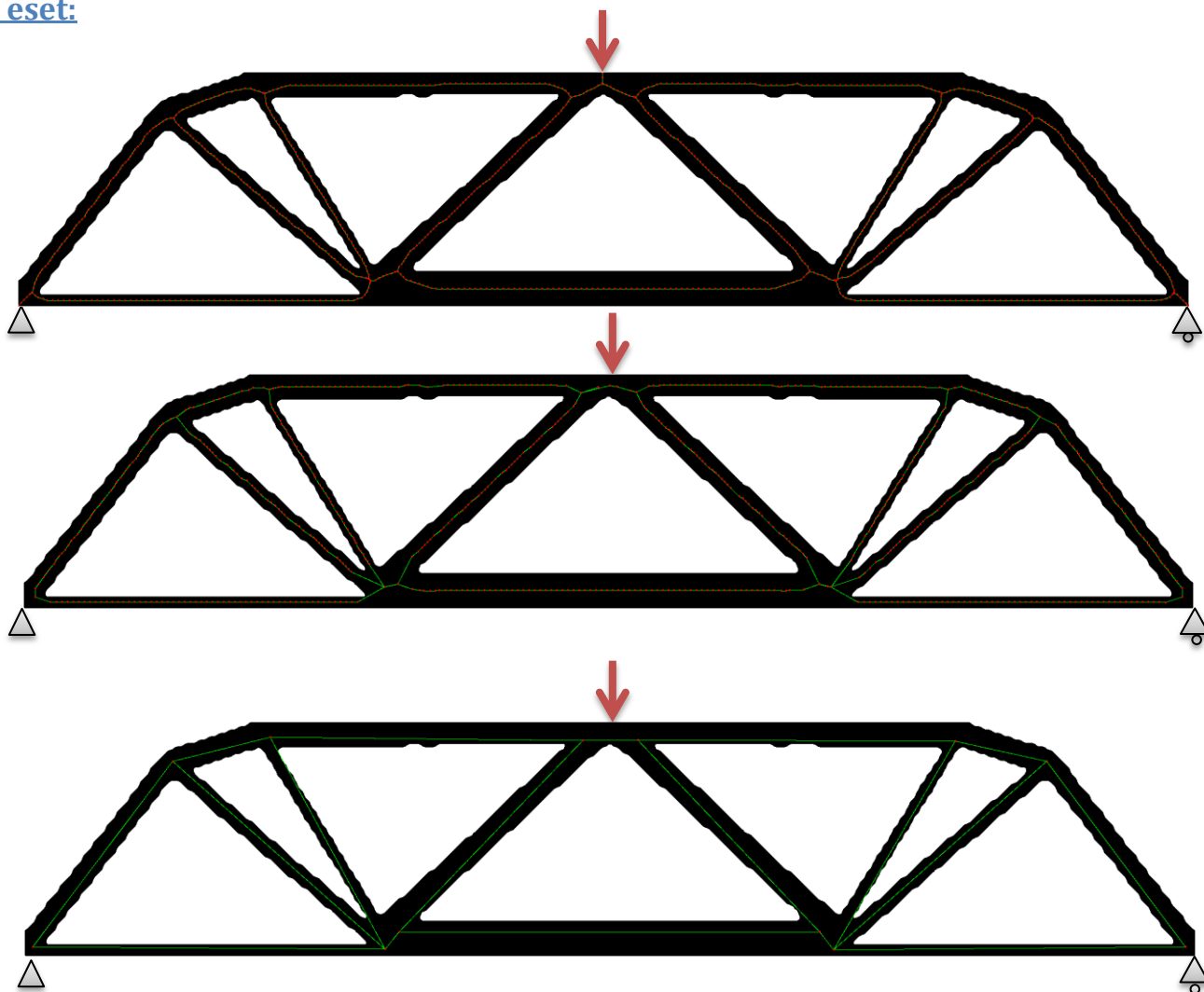
32. ábra/a,b,c

Rúd-detektálás eredménye (bal felső)

LLPO eredménye (jobb felső)

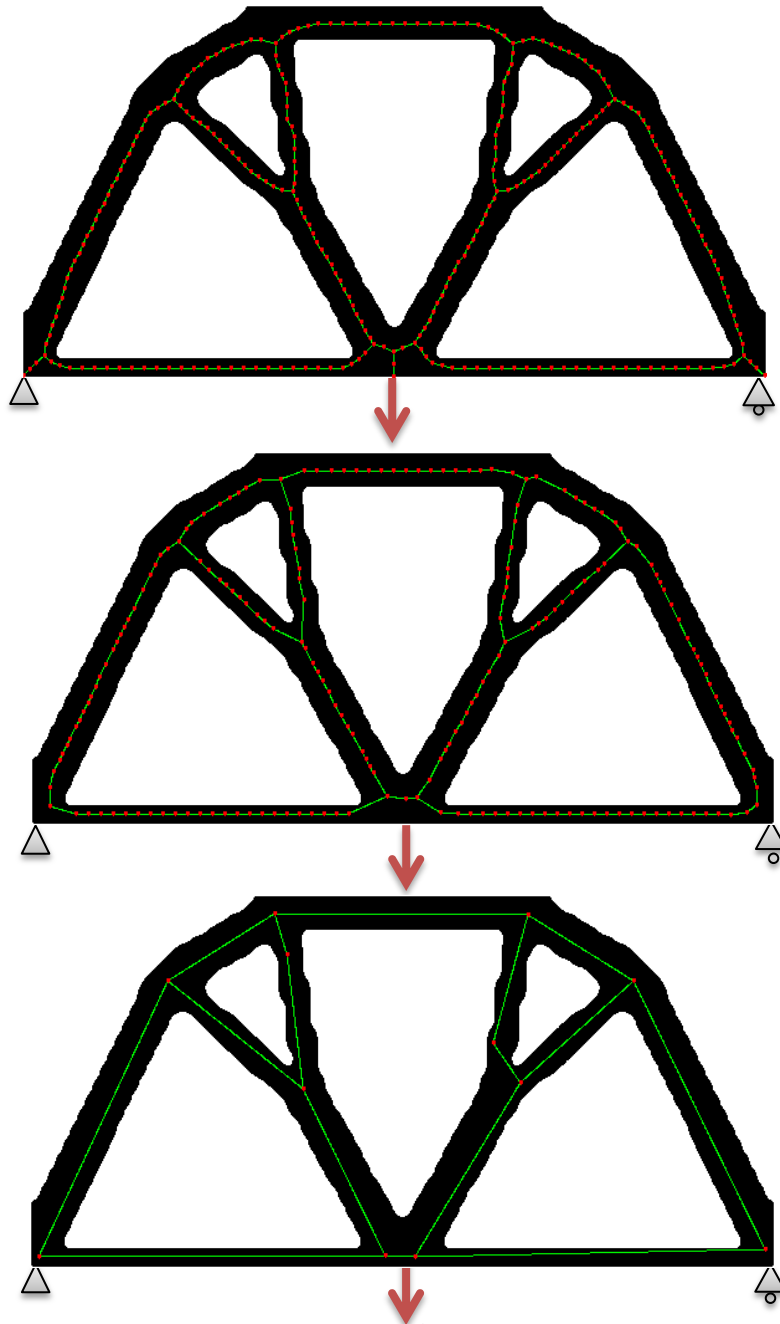
BPPF eredménye (alsó középső)

5. eset:



33. ábra/a,b,c
Nagyfelbontású mintapélda
a, BPPF eredménye
b, LLPO eredménye
c, Rúd-detektálás eredménye

6. eset:



34. ábra/a,b,c
a, BPPF eredménye
b, LLPO eredménye
c, Rúd-detektálás eredménye⁴

⁴ **Megjegyzés:** A rúd-detektálás eredményében még találhatóak hibák, azonban ez nem az algoritmus hibája, hanem még nem kerültek implementálásra a *csomópontokra* vonatkozó feltételek, aminek eredményeként az algoritmus nem tudta pontosan meghatározni a csomópont típusát (ha a középvonalak nem egy pontban találkoznak, akkor a kapcsolódás típusa kérdéses)