

Strukturális alaksajátosságok érvényesítése CAD modellek rekonstrukciója során

Kovács István, Salvi Péter, Várady Tamás

Budapesti Műszaki és Gazdaságtudományi Egyetem

Kivonat A mérnöki visszafejtés (digitális alakzatrekonstrukció) különböző tárgyak számítógépes modelljének létrehozásával foglalkozik mért adatpontok alapján. A folyamat során pontatlanság származik a zajos adatokból, illetve az algoritmusok numerikus bizonytalanságából. A rekonstruált síkok nem lesznek pontosan párhuzamosak/merőlegesek, a forgásfelületek tengelyei nem lesznek teljesen azonosak, a sima felületek nem tökéletesen kapcsolódnak stb. Cikkünkben algoritmusokat ismertetünk ezen problémák kezelésére, és ezáltal olyan modellek létrehozására, amelyek megfelelnek a CAD alkalmazások pontossági követelményeinek. A kifejlesztett eljárások lehetőséget nyújtanak szegmentált felületek közötti kényszerek automatikus felismerésére és érvényesítésére. Kutatásunk egy korábbi publikáció [1] eredményeire épít. Ez előre megadott felületekkel és kényszerekkel foglalkozott, a bemutatásra kerülő eljárás lényege viszont, hogy a kényszeres illesztés elemei automatikusan választódnak ki. Algoritmusokat adunk egy adott modellhez legjobban illeszkedő koordinátarendszerek, globális szerkesztőrácsok, valamint teljes vagy részleges tükröszimmetriák felismerésére. Ezek működését számos példával illusztráljuk.

1. Bevezetés

A mérnöki visszafejtés (digitális alakzatrekonstrukció, vagy reverse engineering) célja, hogy mért adatok alapján elkészítsük egy tárgy számítógépes modelljét. Ezt a technológiát számos területen alkalmazzák, többek között régi alkatrészek újragyártására, a gyártási minőség ellenőrzésére, személyre szabott gyógyászati segédeszközök (pl. protézisek, fogsorok, hallókészülékek) előállítására, illetve kulturális örökségünk (pl. épületek, szobrok) digitális megőrzésére.

A digitális alakzatrekonstrukció folyamata a következő technikai fázisokra tagolható: (1) 3D-s mérés (szkennelés), (2) pontfelhalmazok szűrése és egyesítése, (3) háromszöghálók létrehozása, (4) háromszöghálók egyszerűsítése és javítása, (5) szegmentálás (tartományokra bontás), (6) tartományok osztályozása, (7) felületek illesztése, (8) összekötő felületek (pl. lekerekítések) illesztése, (9) *felületek tökéletesítése (kényszerek alkalmazása, simítás)*, (10) határolóelem reprezentáció (B-rep) modell elkészítése, (11) exportálás CAD/CAM rendszerekbe.

Mérésből származó zajos adatpontokat matematikai felületekkel közelítve a különböző kényszerek nem fognak pontosan teljesülni; ugyanakkor a mérnöki alkalmazások során a modelleknek pontosaknak kell lenniük, és elengedhetetlen, hogy a felületek között „tökéletes” geometriai kényszerek érvényesüljenek. Ezek lehetnek lokális, két felület között ható kényszerek, ilyen pl. a merőlegesség, párhuzamosság, közös tengelyek stb., de lehetnek globálisak is, mint pl. egy közös koordináta-rendszerhez való illeszkedés vagy a tükrörszimmetria. Hagyományosan a kényszereket általában a felhasználónak kellett megadnia, ezzel szemben a jelen kutatás célja egy olyan „intelligens” eljárás kifejlesztése, amely a kényszereket automatikusan felismeri és ezáltal lehetővé teszi rekonstruált mérnöki modellek tökéletesítését minimális felhasználói segítség igénybevételével.

A továbbiakban feltesszük, hogy adott egy szegmentált háromszögháló, melynek tartományait felülettípus szerint osztályoztuk, és ezeket kényszerek nélkül külön-külön megillesztettük. Algoritmusaink az alábbi gyakran előforduló felülettípusokat használják: síkok; másodfokú felületek (henger, kúp, gömb); (egy profilgörbe és egy kihúzási irány által meghatározott) kihúzott és ferdén kihúzott felületek; (egy profilgörbe és egy forgástengely által meghatározott) forgásfelületek. Minden más felületet szabadformájúnak tekintünk – ezekkel a jelen cikkben nem foglalkozunk, az ilyen típusú kényszeres illesztések jövőbeni kutatásunk tárgyát képezik.

A bemutatásra kerülő algoritmusok mindegyike a következő négy lépésre bontható: (1) felületek független illesztése, (2) optimális, a globális struktúrát jellemző segédelemek meghatározása és rangsorolása, (3) a legjobb(ak) kiválasztása, az „elfogadás” ellenőrzése egy küszöb szerint, és (4) az illeszkedő elemek kényszeres újraillesztése. A fenti koncepciót részletesen, több speciális esetben mutatjuk be. Mért adatok esetén általában kevés információnk van a modell tervezésekor használt, „természetes” koordináta-rendszeréről, ennek megtalálására a 4. fejezetben adunk algoritmust. Az 5. fejezetben bemutatunk egy eljárást, amely egy optimális *szerkesztőrácsot* állít elő, melyhez a modell teljesen vagy részlegesen illeszkedik. Végül a 6. fejezetben bemutatjuk, hogyan ismerhetők fel egy modell teljes vagy részleges tükrörszimmetria-síkjai.

2. Korábbi eredmények

A modellek tökéletesítése és szabályos tulajdonságainak vizsgálata intenzíven kutatott téma. A kapcsolódó publikációk különböző területeket fognak át, a képfeldolgozástól kezdve az orvosi alkalmazásokon át a mérnöki visszafejtésig. A feldolgozandó adatok jellege teljesen eltérő lehet, pl. bittérképek, ponthalmazok, háromszöghálók, voxelek, vagy határolófelület reprezentációs (B-rep) modellek. Természetesen az input alapvetően meghatározza a különböző algoritmusok működését, és gyakran az egyik területen hatékony algoritmus a másikon használhatatlan.

A gépészeti alkatrészek rekonstrukciója nagyon komplex feladat; a háromszöghálók elkészítésétől a CAD rendszerekbe való exportálásig számos lépést kell végrehajtanunk. A mérnöki visszafejtésről a szakirodalomban részletes és átfogó

képet kaphatunk [10], jelenleg a számunkra érdekes lépés a modellek tökéletesítése. Az ehhez használt kényszerek nagyméretű nemlineáris egyenletrendszerek megoldásához vezetnek, melyek megoldására több algoritmus is ismert [1, 8, 11]. Egy közelmúltban publikált eljárás RANSAC alapú algoritmus segítségével szegmentálja és azonosítja a felületeket, és illesztés közben tökéletesíti a teljes modellt [7, 9]. Ezek a algoritmusok csak bizonyos felületek és bizonyos kényszerek lekezelésére alkalmasak, a túlhatározott kényszeregyenletek kezelésére egy komplex gráfelméleti algoritmust javasolnak. Fontos megemlítenünk Ke et al. [2] munkáját, amelyben kihúzásos és forgásfelületeket is kezelnek – ezek profilját felismerik, tökéletesítik, majd kényszeresen illesztik. A Cardiff Egyetemen számos fontos publikáció született rekonstruált modellek tökéletesítéséről [4–6]. Ezek a publikációk nagyrészt teljes vagy részleges szimmetriák felismeréséről szólnak, algoritmusaikban algebrai módszereket alkalmaznak jellegzetes pontok (feature points) alapján. Ezen eljárás egyik fő nehézsége, hogy ilyen pontok nem mindig állnak rendelkezésünkre, vagy azokat nehéz megtalálni. A szerzők foglalkoztak még a modellben előforduló számadatok arányainak tökéletesítésével is, ezt a kérdést cikkünkben általánosabb módon vizsgáljuk.

Jelen kutatás Benkő et al. [1] módszerén alapul, mely segítségével a kényszeres illesztés során előforduló nagy méretű nemlineáris egyenletrendszereket hatékonyan meg tudjuk oldani, és az esetleges redundáns, illetve ellentmondó kényszereket le tudjuk kezelni. Egy, a közelmúltban megjelent munkánkban bemutattuk, hogy hogyan lehet lokális kényszereket felismerni és érvényesíteni 2D-ben [3], jelen publikáció ezen algoritmusok 3D-s kiterjesztésével foglalkozik.

3. A kényszeres illesztés alapjai

A mért adatokból előállított szegmentált modell önmagában nem tartalmaz információt a szükséges kényszerekről, azaz a felületek lokális és globális kapcsolatairól. Ezeket vagy a felhasználónak kell megadnia, vagy valamilyen intelligens algoritmusnak kell felismernie.

A kényszeres illesztés során kétféle objektumtípust különböztetünk meg. Az elsőbe tartoznak az ún. *elsődleges objektumok* (primary objects) – olyan görbék és felületek, amelyek ténylegesen illeszkednek mért adatpontokra; a másodikba pedig a *segédobjektumok* (auxiliary objects), ezek nem illeszkednek mért adatpontokra, de kényszerek állíthatók fel köztük és az elsődleges objektumok között. A segédobjektumok egyszerűsítik és stabilizálják az egyenletrendszert, és lehetővé teszik bonyolultabb kényszerek definiálását is. (Egyszerű példák találhatók [3]-ben.) Az objektumok n paraméterét egy n -dimenziós vektor határozza meg, a kényszerek ezen vektor koordinátáira felírt algebrai egyenletek illetve egyenletrendszerek segítségével fogalmazhatóak meg.

A felületillesztést a következőképpen formalizálhatjuk. Jelöljük az illesztendő felületeket $\{s_i\}$ -vel, és a hozzájuk tartozó adatpontokat $\{p_{ij}\}$ -vel. A felületapproximációs algoritmus célja, hogy minimalizálja a négyzetes távolságot a felületek és az adatpontok között. Az $\mathbf{x}_i$ vektor tartalmazza az i -edik felület paramétereit,

ekkor minden felületre *külön-külön* minimalizáljuk az alábbi kifejezéseket:

$$f_i(\mathbf{x}_i) = \sum_j d(p_{ij}, s_i(\mathbf{x}_i))^2. \quad (1)$$

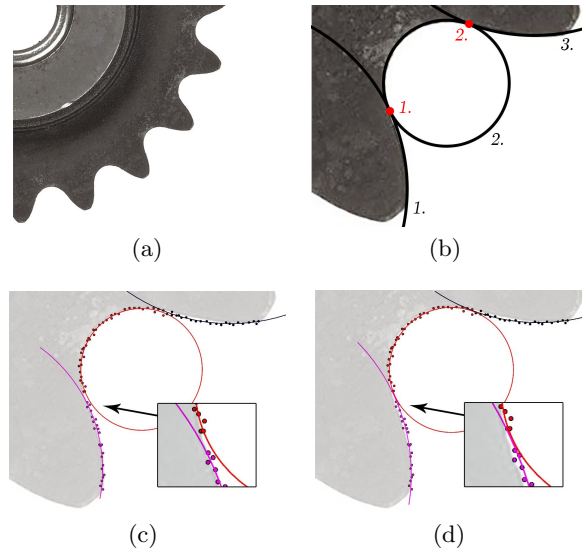
Annak érdekében, hogy a modelljeinket tökéletesítsük, kényszeres illesztést alkalmazunk. Ez azt jelenti, hogy több felületet *együttesen* illesztünk, ahol a felületek paramétereit a kényszeregyenletek kapcsolják össze. Így a minimalizálandó kifejezés

$$f(\mathbf{x}) = \sum_i \alpha_i \sum_j d(p_{ij}, s_i)^2 \quad (2)$$

alakban írható, ahol α_i az i -edik felülethez tartozó súly (pl. a felület mérete, vagy a pontok száma), és $\mathbf{x}$ tartalmazza az összes felület paramétereit. A kényszereket általában egy vagy több nemlineáris egyenlettel adjuk meg, $c_k(\mathbf{x}) = 0$ alakban, ahol c_k egy felülepáron vagy felületcsoporton ható kényszer. A globális kényszereket leíró egyenletrendszerünk pedig

$$\mathbf{c}(\mathbf{x}) = 0. \quad (3)$$

A kényszeres illesztési feladat algebrai megfogalmazását egy egyszerű példa segítségével demonstráljuk.



1. ábra. Kényszeres illesztés: (a) egy fogaskerék profilja; (b) három kör előírt érintési kényszerekkel; (c) független illesztés – rossz kapcsolódás; (d) kényszeres illesztés – sima kapcsolódás.

3.1. Kényszeres illesztés – példa

Tekintsük az 1(b) ábrán látható fogaskerék köríveire illeszkedő, szegmentált profilgörbét. Ha a köröket függetlenül illesztjük, az érintőleges kapcsolódások nem teljesülnek, míg kényszeres illesztéssel egy pontos megoldást kapunk, ahol a kapcsolódás sima. Jelöljük az i . kör paramétereit (A_i, B_i, C_i, D_i) -vel, ahol a kör egyenlete $A_i(x^2 + y^2) + B_ix + C_iy + D_i = 0$. A minimalizálandó négyzetes távolság

$$f(\mathbf{x}) = \sum_{i,j} d_{i,j}^2 = \sum_i \frac{1}{n_i} \sum_j (A_i(x_{ij}^2 + y_{ij}^2) + B_ix_{ij} + C_iy_{ij} + D_i)^2.$$

A kényszeregyenletek:

- három normalizálási kényszer $(B_i^2 + C_i^2 - 4A_iD_i = 1)$,
- két érintőlegességi kényszer $(2A_jD_i + 2A_iD_j - B_iB_j - C_iC_j \pm 1 = 0)$.

Így végül hét szabad paraméterünk marad; a minimalizálandó egyenletrendszer megoldva megkapjuk a körívek optimális, simán összekapcsolódó illesztését.

3.2. A numerikus eljárás

Most tömören bemutatjuk Benkő et al. [1] eljárását a kényszeres illesztésre, mivel ez nagyban segít a későbbi algoritmusok viselkedésének megértésében. Legyenek a kényszeregyenleteink fontossági sorrendben $\mathbf{c}(\mathbf{x}) = (c_1(\mathbf{x}), \dots, c_k(\mathbf{x}))$ és tegyük fel, hogy $f(\mathbf{x})$ és $\mathbf{c}(\mathbf{x})$ elegendően simák (legalább C^2 folytonosak). Ez egy igen nagy méretű nemlineáris egyenletrendszer, amit egy speciális Newton-iterációval oldunk meg. Ehhez először vizsgáljuk meg f és $\mathbf{c}$ Taylor sorát $\mathbf{x}_0$ körül.

$$\mathbf{c}(\mathbf{x}_0 + \mathbf{d}) \approx \mathbf{c}(\mathbf{x}_0) + \mathbf{c}'(\mathbf{x}_0)\mathbf{d}, \quad (4)$$

$$f(\mathbf{x}_0 + \mathbf{d}) \approx f(\mathbf{x}_0) + f'(\mathbf{x}_0)\mathbf{d} + \frac{1}{2}\mathbf{d}^T f''(\mathbf{x}_0)\mathbf{d}. \quad (5)$$

Ezek segítségével a feladat közelítőleg a következő formában írható:

$$C\tilde{\mathbf{d}} = 0 \quad (6)$$

$$\tilde{\mathbf{d}}^T A \tilde{\mathbf{d}} \rightarrow \min, \quad (7)$$

ahol $\tilde{\mathbf{d}} = (d_1, \dots, d_n, 1)$, $C = [\mathbf{c}'(\mathbf{x}_0) | \mathbf{c}(\mathbf{x}_0)]$ és A az alábbi $(n+1) \times (n+1)$ méretű mátrix:

$$A = \left[\begin{array}{c|c} f''(\mathbf{x}_0) & f'(\mathbf{x}_0) \\ \hline f'(\mathbf{x}_0)^T & 0 \end{array} \right].$$

Annak érdekében, hogy $\tilde{\mathbf{d}}$ -ot kiszámítsuk, először redukáljuk egy alacsonyabb dimenziós $\mathbf{d}^*$ vektorra (6) segítségével úgy, hogy $\mathbf{d}^*$ koordinátái már függetlenek legyenek. Ehhez (a Gauss eliminációhoz nagyon hasonló módon) kiszámítunk egy M mátrixot, amivel $\tilde{\mathbf{d}} = M\mathbf{d}^*$, és $CM = 0$. Ekkor $\mathbf{d}^*$ koordinátáinak száma megadja, hogy hány független változóból áll a rendszer, és már csak a $\mathbf{d}^{*T} A^* \mathbf{d}^*$

kifejezést kell minimalizálnunk (kényszerek nélkül), ahol $A^* = M^T A M$. A kényszer nélküli minimalizáció egy egyszerű lineáris egyenletrendszer megoldására vezet.

Az M kiszámításakor az eliminációs lépések során előfordulhat, hogy egy lépés nem végezhető el. Ekkor az aktuálisan vizsgált egyenlet vagy következik az előzőekből, vagy ellentmond azoknak, a C mátrixból ez könnyen leolvasható. Megjegyezzük, hogy vannak kényszerek, amelyek több egyenletből állnak, így ellentmondás esetén az összes egybetartozó egyenletet figyelmen kívül kell hagynunk.

3.3. Lokális kényszerek automatikus felismerése

A kényszerek általában csak valamilyen tolerancián belül teljesülnek a mért adatok zajosságából eredően, például merőleges síkok hajlásszöge nem pontosan 90 fok, azonos sugarú körök sugara egy picit eltér, vagy egyenesek nem haladnak át egy közös ponton, holott az eredeti CAD objektum tervezésekor ezek a megszorítások feltehetően fennálltak. Ahhoz tehát, hogy hipotéziseket állíthassunk fel a lehetséges kényszerekre vonatkozóan, toleranciaszinteket kell beállítanunk. Megjegyezzük, hogy a távolságtoleranciák általában függnak az éppen vizsgált adathalmaz méretétől, ezért ezeket gyakran az objektum befoglaló dobozának átlójához viszonyított százalékos arányban specifikálják. A különböző szögtoleranciák pedig az adathalmaz zajosságával vannak összefüggésben.

Az algoritmus alapötlete, hogy az automatikus felismeréshez *az eredeti kényszeregyenleteknek egy módosított változatát használjuk*, amely csak akkor lép érvénybe, ha az adott kényszer hibája a felhasználó által megadott tolerancián belül van. Az algoritmus részleteit korábbi 2D-s publikációnk tartalmazza [3].

4. Optimális koordinátarendszerek keresése

A legtöbb gépészeti alkatrészt egy jól definiált koordinátarendszerben tervezik meg. Ez azt jelenti, hogy a különböző geometriájú felületek (síkok, hengerek, forgásfelületek stb.) orientációja általában jól illeszkedik egy megadott koordinátarendszerhez. Amennyiben ezeket a modelleket mért adatokból állítjuk elő, a modell a szkennelés során előálló koordinátarendszerbe kerül, mely általában nem azonos a tervezés során adott természetes koordinátarendszerrel. Feladatunk tehát, hogy felismerjük a modellhez legjobban illeszkedő koordinátarendszert vagy rendszereket, az adatokat áttaszformáljuk ebbe az optimális rendszerbe, és kényszeresen megillesszük a hozzájuk tartozó felületeket. A kereséshez a leggyakrabban előforduló, orientációval egyértelműen rendelkező felülettípusokat használjuk fel. Az általános söpört és szabadformájú felületeket nem vesszük figyelembe.

A szegmentált tartományokhoz – a felületek típusának megállapítása mellett – hozzárendelünk d_i irányvektorokat is, és meghatározzuk a tartományok felületét, amit A_i jelöl. Így például a síkokhoz a legjobban illeszkedő normálvektort, a kihúzásos felületekhez a legjobb kihúzásos irányvektort, a forgásfelületek-

hez a tengelyez tartozó irányvektort stb. rendeljük. A modell összfelülete legyen $A = \sum_i A_i$.

Ahhoz, hogy eldönthessük, melyik a legjobb koordináta-rendszer, minősítünk kell azokat. Tegyük fel, hogy adott már egy (n_x, n_y, n_z) páronként merőleges egységvektorokkal rendelkező koordináta-rendszer. Az elsődleges megfelelési mérték azt vizsgálja, hogy a modellnek mekkora része illeszkedik az adott koordináta-rendszerhez. Először összegyűjtjük a hozzávetőlegesen illeszkedő felületek indexeit

$$I(n_x, n_y, n_z) = \{i : |d_i \times n_x| < \varepsilon \vee |d_i \times n_y| < \varepsilon \vee |d_i \times n_z| < \varepsilon\},$$

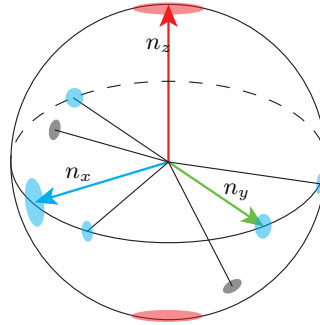
és kiszámoljuk a *fedési mértéket*

$$\text{Coverage}(n_x, n_y, n_z, \varepsilon) = \frac{\sum_{i \in I(n_x, n_y, n_z)} A_i}{A}.$$

Szintén lényeges, hogy az orientált elemek mennyire jól illeszkednek az adatokra, ennek mérésére használjuk az *approximációs hibát*:

$$\text{ApprErr}(n_x, n_y, n_z) = \frac{\sum_{i \in I(n_x, n_y, n_z)} A_i \min(|d_i \times n_x|, |d_i \times n_y|, |d_i \times n_z|)}{A}.$$

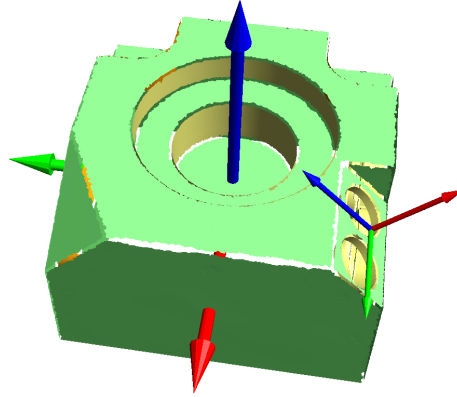
Célunk tehát megkeresni azt a koordináta-rendszert, amire a fedési mérték a lehető legjobb, és az approximációs hiba ezen belül a legkisebb.



2. ábra. Domináns klaszterek a Gauss-gömbön.

4.1. Az algoritmus

Vegyük a Gauss gömböt, és képezzük rá a d_i egységvektorokat úgy, hogy mindegyikhez hozzárendeljük az A_i súlyt is. Ezen vektorok között fogunk klasztereket kialakítani, amelyek majd a legjobb koordináta-hármasokat fogják definiálni. Az



3. ábra. Egy globális (96.34% fedés) és egy lokális koordinátarendszer felismerése.

algorithmus a *klaszternövesztés* elvén működik (lásd 2. ábra). Minden C klaszterhez hozzárendelünk egy p vektort a következőképpen:

$$p' = \sum_{d_i \in C} A_i d_i, \quad p = \frac{p'}{\|p'\|}.$$

1. *lépés.* Vegyünk egy d_i vektort. Nézzük meg, van-e olyan klaszter, amelyben a p és d_i (vagy $-d_i$) vektorok által bezárt szög kisebb, mint ε . Ha igen, vegyük hozzá d_i -t (vagy $-d_i$ -t) a klaszterhez. Ha nem, hozzuk létre a $C = \{d_i\}$ egyelemű klasztert. Ezt a lépést végezzük el az összes vektorra.

2. *lépés.* Számítsuk ki a klaszterek elsődleges súlyát:

$$G(p) = \{i : |d_i \times p| < \varepsilon\}, \quad W_1(p) = \sum_{i \in G(p)} A_i.$$

3. *lépés.* Számítsuk ki a klaszterek másodlagos súlyát, ami a p -re merőleges klaszterek összsúlya, azaz

$$H(p) = \{k : |\langle p_k, p \rangle| < \varepsilon\}, \quad W_2(p) = \sum_{k \in H(p)} A_k.$$

4. *lépés.* Rangsoroljuk a klasztereket $W_1(p) + W_2(p)$ szerint, majd válasszuk ki a legerősebbet. A hozzátartozó p legyen n_z . Ezek után a rá merőleges klaszterek közül fogunk választani megfelelő n_x és n_y vektorokat.

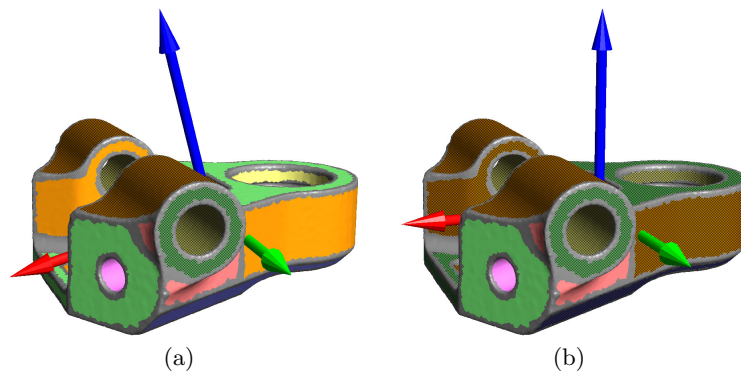
5. *lépés.* A $H(n_z)$ klaszterhalmaz egy eleme legyen p^* , az ehhez tartozó harmadlagos súly legyen

$$L(p^*) = \{l : |\langle p_l^*, p^* \rangle| < \varepsilon\}, \quad W_3(p^*) = \sum_{l \in L(p^*)} A_l,$$

ezek után legyen n_x az, amire $W_1(p^*) + W_3(p^*)$ a legnagyobb.

6. lépés. Legyen $n_y = -n_x \times n_z$.

Egy optimális globális koordinátarendszert mutat a 3. ábra. A nem illeszkedő, megmaradt felületekre az algoritmust újra alkalmaztuk, így egy kisebb lokális koordinátarendszert határoztunk meg. A fedési mérték megváltozását szemlélteti a 4. ábra az (a) eredeti koordináta rendszerben, és a (b) optimális transzformáció végrehajtása után.



4. ábra. Koordinátarendszer meghatározása: (a) eredeti (29.82% fedés, 1.864 approximációs hiba); (b) optimalizált (86.11% fedés, 0.078 approximációs hiba).

5. Optimális szerkesztőrács keresése

A jó CAD modellek ismérve, hogy méreteinek értékei egészek (vagy egy adott szám többszörösei) valamilyen mértékegység szerint. Ennek a feladatnak többféle megfogalmazása lehetséges.

(i) A legegyszerűbb eset, amikor az értékeket egy egész számra szeretnénk beállítani, például az illesztett kör sugara 10.2, de mi előírjuk azt a kényszert, hogy a sugár legyen 10.

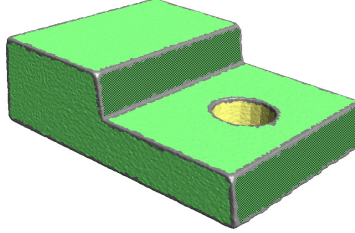
(ii) Ennél némileg általánosabb, ha előírjuk a kvantálás egységét (h), vagyis legyenek az értékek h többszörösei, azaz

$$r_i/h - \text{round}(r_i/h) = 0.$$

(iii) Még általánosabb eset, amikor h sem ismert. Vegyük például az 5. ábrán látható vízszintes síkokat. Ezeknek az origótól való távolsága ($z_1 = 2.12, z_2 = 11.97, z_3 = 22.06$). Keressük a legjobb h -t, amivel a számok különbségének h -tól vett átlagos (h -hoz viszonyított) maradéka a legkisebb, azaz

$$\delta(h) = \sum_{i,j} \min \left(\left\{ \frac{z_i - z_j}{h} \right\}, 1 - \left\{ \frac{z_i - z_j}{h} \right\} \right), \quad (8)$$

ahol $\{x\}$ az x törtrészét jelöli. Végül azt kapjuk, hogy $h = 10$ és $(z_1 = 2, z_2 = 12, z_3 = 22)$.



5. ábra. Tesztobjektum közel párhuzamos síkokkal.

(iv) A legáltalánosabb probléma, amikor egy optimális szerkesztőrácsot keressünk, azaz egy (x_0, y_0, z_0) origót, és egy h cellaméretet. Ekkor a minimalizálandó mennyiség

$$\delta(h, z_0) = \sum_i \min \left(\left\{ \frac{z_i - z_0}{h} \right\}, 1 - \left\{ \frac{z_i - z_0}{h} \right\} \right). \quad (9)$$

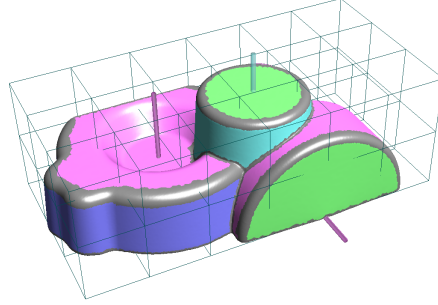
Az előző példában az eredmény $z_0 = 2$, és a síkok pozíciója az új origóhoz viszonyítva $(z_1 = 0, z_2 = 10, z_3 = 20)$.

Tipikus tervezési gyakorlat, hogy a modell elemei valamilyen koordináta-rendszerben egy adott képzeletbeli rácshoz vannak illesztve. Jelen algoritmus működéséhez feltesszük, hogy a koordináta-rendszert már meghatároztuk, a rács origója és a cellaméret viszont ismeretlen. A rácskeresés legnagyobb nehézsége, hogy a minimalizálandó függvény nem sima, ezért a Newton–Raphson féle algoritmusok nem működnek, és sajnos magas futásidejű módszereket kell használnunk. Bemutatunk egy kétlépéses algoritmust, ahol először meghatározzuk a cellaméretet, majd az origót. Megjegyezzük, hogy a keresés egy lépésben is elvégezhető, ha a két paramétert egyszerre optimalizáljuk, de ez nagyon költséges, így a gyakorlatban alkalmazását nem javasoljuk.

5.1. Az algoritmus

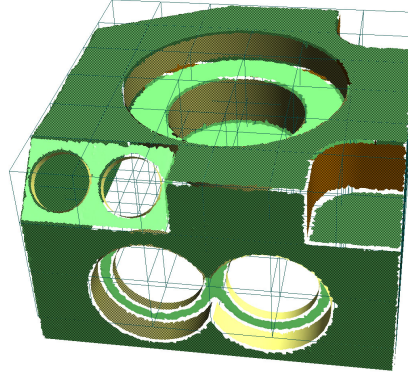
Az algoritmus először kiszámítja az optimális cellaméretet, majd beállítja az optimális origót. Ehhez először ki kell számítani az egymással párhuzamos, orientált entitások (síkok és tengelyek) közötti távolságokat, legyenek ezek $\{d_i\}$, $i = 1, \dots, N$, valamint legyenek a súlyok $w_i = \min(A_k, A_l)$, ahol A_k és A_l a megfelelő objektumok felületei. A feladatunk ezen számok között egy legjobb approximatív közös osztó keresése, azaz minimalizálandó a

$$\delta(h) = \sum_{i=1}^N \frac{w_i}{h} \min \left(\left\{ \frac{d_i}{h} \right\}, 1 - \left\{ \frac{d_i}{h} \right\} \right).$$



6. ábra. Optimális szerkesztőrács egy tesztobjektumon (66.82% fedés).

Célunk megkeresni $\delta(h)$ minimumát egy előre adott $[h_{\min}, h_{\max}]$ intervallumban. Egyszerűen látható, hogy a függvény szakaszonként monoton, és a szakaszok végpontjai d_i/k alakú számok, ahol k pozitív egész. Így tehát a minimumot csak ezekben az előírt számokban kell megkeresni, ami $O(N^2 d_{\max}/h_{\min})$ lépés, ahol $d_{\max} = \max_i n_i$.



7. ábra. Optimális szerkesztőrács egy tesztobjektumon (77.45% fedés)

Miután meghatároztuk az optimális cellaméretet, rátérhetünk az origó meghatározására. Előfordulhat az is, hogy az origót valamilyen előre adott síkra akarjuk kényszeríteni. Ekkor a minimalizálást kevesebb koordinátára kell elvégeznünk. Általános esetben azonban mindhárom koordinátát ki kell számolnunk. Jelölje u az x , y és z valamelyikét. Ekkor az alábbi kifejezést kell tengelyenként minimalizálni:

$$\delta(u_0) = \sum_i \frac{w_i}{h} \min \left(\left\{ \frac{u_i - u_0}{h} \right\}, 1 - \left\{ \frac{u_i - u_0}{h} \right\} \right). \quad (10)$$

A minimalizálás a cellamérethez hasonlóan működik, a számokat csak előírt helyeken kell keresni, amit elvégezhetünk $O(n^2)$ lépésben. Tesztpéldákat láthatunk a 7. és 6. ábrákon; mindkét esetben a főtengekkel párhuzamos forgásfelületek tengelyei is részt vettek az optimális rács meghatározásában.

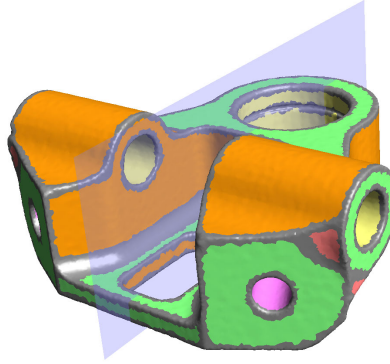
6. Tükörszimmetriák detektálása

A CAD modellek tökéletesítésének egyik legfontosabb eleme a szimmetriák felismerése és érvényesítése. Jelen kutatás során a tükörszimmetria-síkok megkeresésével foglalkoztunk. A legtöbb esetben a modellek csak részben szimmetrikusak, azaz a legjobb szimmetriasíkkal a modellnek csak egy része szimmetrikus, egyes lokális elemek nem. Ebben a fejezetben egy olyan algoritmust ismertetünk, amely meg tudja találni egy modell globális (teljes vagy részleges) szimmetriasíkjait.

Legyen az s_i felület P síkra tükrözött képe $P(s_i)$. A szimmetria mértéke ekkor az átfedő részek felülete, azaz

$$\frac{2\text{Area}(s_1 \cap P(s_2))}{\text{Area}(s_1) + \text{Area}(s_2)}.$$

Teljes szimmetria esetén ez az érték 1 (lásd 8. ábra).



8. ábra. Teljes szimmetria.

Jelen helyzetben, mért adatok esetén, az átfedések nem tökéletesek, így s_i és $P(s_i)$ nem tökéletesen illeszkednek, és az átfedő területet csak valamely ε tolerancián belül tudjuk értelmezni.

Az algoritmus főbb lépései a következők:

1. lépés. Szimmetriasík-jelöltek meghatározása az objektumpárok között.
2. lépés. Klaszterezés a jelöltek között, majd ezek rangsorolása egy speciális súlyozás szerint.
3. lépés. Legjobb sík meghatározása, kényszeres illesztés.
4. lépés. Az előző lépés ismétlése minden olyan klaszterre, ahol még elfogadható a szimmetria mértéke.

6.1. Szimmetriasík-jelöltek

Ahhoz, hogy szimmetriasík-jelölteket definiáljunk, válasszuk ki az azonos típusú objektumpárokat, azaz síkpárokat, hengerpárokat, forgásfelület-párokat stb. Egy p_i, p_j síkpárra legyen a jelölt $P_{i,j}$ egy szögfelező sík. Az r_i, r_j forgásfelület párokra l_i, l_j tengelyekkel akkor értelmezzük jelöltet, ha l_i és l_j egy síkban vannak (tolerancián belül). Ekkor $P_{i,j}$ legyen a két egyenesnek egy olyan szögfelező síkja, amely merőleges a közös síkra. Ebben a lépésben az általános profilú kihúzásos és ferdén kihúzásos felületek nem vesznek részt.

6.2. Klaszterezés

Azt mondjuk, hogy két sík (ε, δ) tolerancián belül megegyezik, ha a normálvektoraik által bezárt szög kisebb, mint ε , és az origótól való távolságaik különbsége kisebb, mint δ . Olyan klasztereket képzünk, ahol a síkok páronként (ε, δ) tolerancián belül megegyeznek. A klaszterek elsődleges súlya a klaszterekben levő síkokhoz tartozó objektumpárok átfedési felülete. A koordinátarendszerekhez hasonlóan itt is klaszternövesztést használunk, ami azt jelenti, hogy sorban próbáljuk a síkokat a klaszterekbe beszúrni, és ha nem sikerül, új egyelemű klasztert hozunk létre.

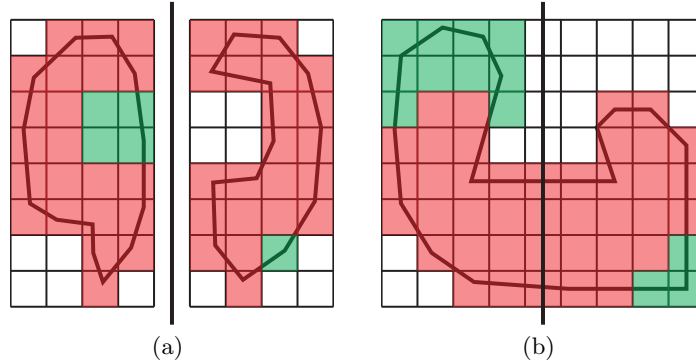
A teljes modell szimmetriájáról az elsődleges súlyozás nem ad teljes képet, ezért bevezettünk még két súlyt. Legyen P egy szimmetriasík-jelölt. A másodlagos súlyozás az elemek „önszimmetriáját” méri, azaz az önmagukkal szimmetrikus felületek szimmetrikus részének felületeit. Ilyenek lehetnek (i) síkok, melyek merőlegesek P -re, forgásfelületek, melyek tengelyei (ii) illeszkednek P -re, vagy (iii) merőlegesek rá, azok a kihúzott vagy ferdén kihúzott felületek, melyek tengelyei (iv) párhuzamosak P -vel, vagy (v) merőlegesek rá.

Az elsődleges és másodlagos súlyozással még bizonyos speciális eseteket nem kezeltünk. Ez akkor fordulhat elő, ha két szimmetrikus objektumpárnak közös a P -re merőleges tengelye, vagy például két P -re merőleges sík egybeesik. Az ilyen jellegű objektumpárok szimmetrikus részének területei adják a harmadlagos súlyozást.

Mindhárom súlyozás olyan objektumok szimmetrikus részeinek felületéből tevődik össze, amelyek egymással vagy önmagukkal szimmetrikusak. Mindhárom különböző típusú felületeket vagy felületepárokat vizsgál. A három súly minden lehetséges esetet lefed, így a végső rangsorolás ezek összege alapján adódik. A következő alfejezetben megmutatjuk, hogyan számítható ki hatékonyan az átfedő felület nagysága.

6.3. A szimmetria mértékének kiszámítása

Legyen adott egy P szimmetriasík-jelölt, és az s_1, s_2 felületek. A feladatunk, hogy hatékonyan kiszámoljuk s_1 és $P(s_2)$ tolerancián belül átfedő felületét. Kezdetben minden egyes felületet egy egyenletes ráccsal paraméterezzük (ez az általunk használt felülettípusoknál egy könnyű feladat, pl. síknál egyenletes rácsozás, hengernél alkotók és körök egyenletes rendszere stb.), majd minden felülethez



9. ábra. Bitmátrix: (a) két objektum szimmetriája; (b) egy objektum saját szimmetriája.

elkészítünk egy bitekből álló diszkretizált mátrixreprezentációt, amely azokat a rácscellákat reprezentálja, amelyek középpontjában a felületnek van pontja. Ezek után csak a felülethez tartozó rácscellák középpontjait tükrözzük, és ezekhez vizsgáljuk meg, hogy a másik ráccsal mekkora az átfedő felület. Ezt elvégezzük a fordított felállásban is, amikor az első felületet tükrözzük. A két érték átlagát tekintjük $s_1 \cap P(s_2)$ felületének. Láthatjuk, hogy az algoritmus futása csak a rács felbontásától függ. Ahhoz, hogy a számítás sok szimmetriasíkkal is hatékony legyen, durva felbontást kell választani, ami azt eredményezi, hogy a becslésünk is elég durva lesz, de ez a jelöltek rangsorolásához kielégítő. Amennyiben pontosítani szeretnénk az eredményen, a legjobb sikokhoz újra lefuttatjuk a számításokat nagyobb felbontással.

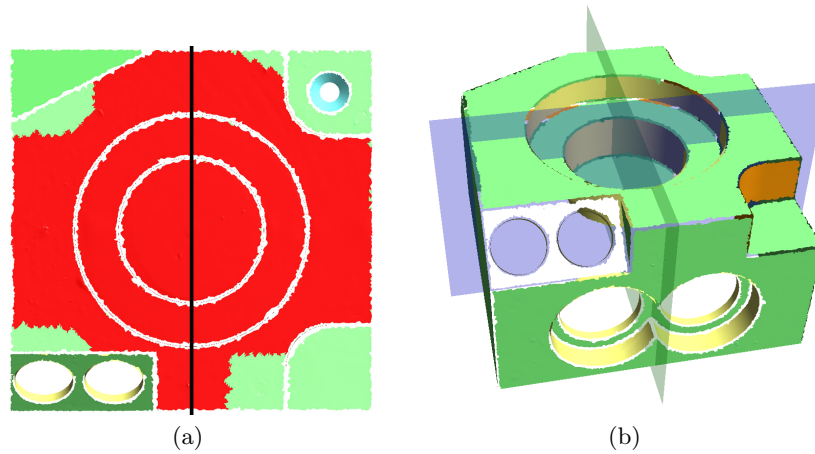
Egy példát láthatunk a 9. ábrán, ahol (a) objektumpárok szimmetriáját, és (b) egy objektum approximatív önszimmetriáját mutatja. A 10. (a) ábrán pedig egy modell felülnézeti képén láthatjuk egy síklap önszimmetriáját. Az algoritmus működésének eredményét a 10. (b) ábra szemlélteti.

7. Összefoglalás

Mért adatokból származó modellek tökéletesítése a mérnöki visszafejtés egy kritikus fázisa. Cikkünkben számos kényszeres illesztésen alapuló algoritmust mutattunk be, többek között (i) globális koordináta-rendszer(ek) és (ii) szerkesztő-rácsok meghatározására, valamint (iii) teljes illetve részleges tükörszimmetriák megtalálására.

Algoritmusainkat egy prototípus tesztkörnyezetben implementáltuk, a cikkben szereplő képek és numerikus eredmények is innen származnak. Az általunk használt modellek egy része szimulált, más része valós mért adatokból származik.

Igen nehéz eredményeinket összehasonlítani mások eredményével, mert ennek a feladatnak ilyen módon történő megoldását nem publikálták. Legjobb tudomásunk szerint nincsenek olyan benchmark objektumok, amelyek a kény-



10. ábra. (a) Szimmetrikus felületrészek ráccsal közelítve; (b) két felismert szimmetriasík, a zöld sík 86.74%, a kék 79.52%-os fedéssel.

szeres illesztés minőségének mérését segítenék. Az igazi teszt az lenne, ha egy adott zajos ponthalmazból a különböző konkurens megoldások elő tudnák állítani a tervezői szándékhoz legjobban illeszkedő „tökéletesített” megoldást, de ilyen nincs. Ezért be kell érünk azzal, hogy egyrészt százalékos értékek segítségével, másrészt grafikus ellenőrzéssel értékeljük, hogy a „legjobb” globális kényszerek milyen felülethalmazt érintettek.

A jövőben szeretnénk algoritmusainkat kiterjeszteni általános szabadformájú felületekre is. Fontos lehet még az algoritmusok további optimalizálása, hogy nagy számú felületet tartalmazó modellekkel is hatékonyan működjenek.

Köszönetnyilvánítás

Jelen kutatást az OTKA 101845 számú pályázata támogatta. Köszönjük Karikó Györgynek az algoritmusok fejlesztésében és a tesztrendszer fejlesztésében való részvételét.

Hivatkozások

1. P. Benkő, G. Kós, T. Várady, L. Andor, and R. R. Martin. Constrained fitting in reverse engineering. *Computer Aided Geometric Design*, 19(3):173–205, 2002.
2. Y. Ke, S. Fan, W. Zhu, A. Li, F. Liu, and X. Shi. Feature-based reverse modeling strategies. *Computer-Aided Design*, 38(5):485–506, 2006.
3. I. Kovács and T. Várady. Perfecting 3D computer models reconstructed from measured data. *VII. Magyar Számítógépes Grafika és Geometria Konferencia*, 7(16):109–115, 2014.

4. F. C. Langbein, A. D. Marshall, and R. R. Martin. Choosing consistent constraints for beautification of reverse engineered geometric models. *Computer-Aided Design*, 36(3):261–278, 2004.
5. M. Li, F. C. Langbein, and R. R. Martin. Detecting approximate symmetries of discrete point subsets. *Computer-Aided Design*, 40(1):76–93, 2008.
6. M. Li, F. C. Langbein, and R. R. Martin. Detecting design intent in approximate CAD models using symmetry. *Computer-Aided Design*, 42(3):183–201, 2010.
7. Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra. Globfit: Consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics (TOG)*, 30(4):52, 2011.
8. J. Porrill. Optimal combination and constraints for geometrical sensor data. *The International Journal of Robotics Research*, 7(6):66–77, 1988.
9. R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for point-cloud shape detection. *Computer graphics forum*, 26(2):214–226, 2007.
10. T. Várady and R. R. Martin. Reverse engineering. *Handbook of Computer Aided Geometric Design*, pages 651–681, 2002.
11. N. Werghi, R. Fisher, C. Robertson, and A. Ashbrook. Modelling objects having quadric surfaces incorporating geometric constraints. In *Computer Vision—ECCV’98*, pages 185–201. Springer, 1998.