B-spline vonalfelületek előállítása egyeneshalmazokból neurális háló segítségével, komputergrafikai vonatkozások

Doktori (PhD) értekezés

Kovács Emőd

Debreceni Egyetem
Természettudományi Kar
Debrecen, 2003.

Tartalomjegyzék

Ábrák jegyzéke

1. Bevezetés

A dolgozat témája a komputergrafika egyik központi területe, a görbe- és felületmodellezés. E terület egyik feladata, hogy egy előre megadott ponthalmazhoz bizonyos feltételeket kielégítő görbét vagy felületet illesszen. Az így létrehozott, úgynevezett szabadformájú felületek széleskörű alkalmazásra találnak a számítógéppel segített tervezésben az autógyártástól a tudományos adatok szemléltetéséig.

A klasszikus geometria jól ismert területei a vonalfelületek és a kifejthető felületek, mindemellett a komputer grafikában és a számítógéppel segített gyártásban (Computer Aided Manufacturing) szintén sok alkalmazása van e felületeknek. Minden ilyen alkalmazás alapvető kérdése a következő. Ha adott egy felület, akkor hogyan lehet közelíteni vonalfelülettel. Ha a szükséges pontossággal meg tudjuk ezt tenni, akkor a vonalfelülettel tudjuk helyettesíteni az eredeti felületet, és egyszerűsödik a gyártás vagy növekszik a pontosság. Sok esetben csak egy ponthalmazt kapunk a felületről, amit például egy lézer szkenner adott meg nekünk. Megtehetjük, hogy egy ponthalmaz (pontfelhő) legyen az inputja az általunk tárgyalt algoritmusoknak. A felületek megadásának szokásos módja, a különböző típusú spline felületek használata. Ilyen lehet például a Bézier, B-spline vagy a NURBS felület. Tehát nagyon kívánatos eredmény lenne, ha a vonal és kifejthető felületeket, mint spline felületeket tudnánk kezelni. A szakirodalomban számos módszert találunk, amelyekben ilyen típusú felületeket használnak felület approximáció vagy felület interpoláció esetén. Mindegyikben azonos azonban, hogy eredetileg rendezett adatok - azaz görbe esetén pontsorozat, felület esetén pontrács - modellezésére alkották őket. Rendezetlen adatok kezelésére ezek a standard algoritmusok eredetileg alkalmatlanok. Ezen dolgozat célja, hogy bemutasson egy olyan eljárást, mely rendezetlen adatok esetén is lehetővé teszi a standard görbe- és felületmodellezési algoritmusok használatát. A bemutatandó eljárás egy előfeldolgozási részből és a tulajdonképpeni felületmodellezésből áll. Az előfeldolgozás során a rendezetlen adatokból olyan pontsorozatot illetve pontrácsot állítunk elő, amelyet az említett modellezési eljárások már inputként használhatnak. A rendezetlen adatok kezeléséhez Kohonen-féle mesterséges neurális hálózatot és annak dinamikus változatát használjuk.

A mesterséges neurális hálózatok elméletének és lehetséges számítás-

technikai alkalmazásainak vizsgálata a 40-es, 50-es években kezdődött W. McCulloch, W. Pitts és Neumann János munkásságával, széleskörű elterjedése azonban az utóbbi két évtizedre tehető. A mesterséges neurális háló, mint az algoritmikus problémamegoldás alternatívája, főként olyan feladatoknál használható, ahol pontos algoritmikus megoldás nem ismert, vagy az nagyon lassú lenne.

A dolgozat első szakaszában –a 2. pontban– a B-spline görbék és felületek modellezését tekintjük át. Terjedelmi okoból csak a legfontosabb definíciókat és fogalmakat tárgyaljuk.

A következő részben áttekintjük az eddigi irodalmat, ahol rendezetlen adatok modelleznek B-spline vagy Bézier felülettel. Bevezetjük a Plücker-koordinátákat, amelyet a jobb megértés kedvéért később még 5.3.1. pontban is kifejtünk. A mérnöki visszafejtés (reverse engineering) kutatási területet megalapozó munka Várady Tamás, és szerzőtársai (R. Martin és J. Cox) 1997-es (lásd [85]) cikkéhez fűződik. Weiss, Andor, Renner és Várady 2002-es cikke alapján tudunk beszámolni a legfrissebb eredményekről (lásd [88]). Annak ellenére, hogy igen intenzív és kiterjedt kutatások folynak paraméteres felületek illesztésére, a megjelent megoldások gyakran hibáznak. A felület rekonstrukció kulcskérdése jó paraméterértékek hozzárendelése az adatpontokhoz, azaz, hogyan találjunk olyan pontokat a sima felületen, amelyek közel vannak az adatpontokhoz. Szabálytalan távolságban lévő véletlen eloszlású adatpontok esetén, a parametrizáció inicializálására szokásos módszer létrehozni egy alap felületet (base surface), amely felület egy durva közelítése az input adatok pontfelhőjének. Ez a felület lehet egy sík, egy henger vagy egy bilineárisan súlyozott Coons-folt (lásd Hermann és szerzőtársai [35]), amely a pontfelhőből nyert geometriai információk alapján állítható elő. A 5.1. pontban ismertetett, általam és szerzőtársaim által kifejlesztett módszerrel egyéb formájú bázis felület is nyerhető.

A 4. fejezetben áttekintést adunk a neurális hálózatokról, továbbá a neurális hálók eddigi alkalmazásairól. Röviden ismertetjük a Kohonen-féle neurális hálózatot.

Az 5. fejezetben olyan görbe- és felületmodellezési módszereket vizsgálunk, melyek a neurális hálózat alkalmazása után rendezetlen adatok modellezésére is használható. Az 5.1. pontban azzal a neurális háló típussal foglalkozunk részletesen, amely a fent vázolt problémához leginkább megfelel, ez pedig a Kohonen féle neurális háló. Az 5.2. pontban a Kohonen-háló

dinamikus változatát vezetjük be, amellyel az algoritmus hatékonyságát sikerült növelni. Az 5.3. pontban vonalfelületet állítunk elő, itt használjuk a Plücker-koordinátákat. Az 5.4. pontban kifejthető felületeket állítunk elő Kohonen háló segítségével. Felhasználjuk a dualitás elvét és hivatkoztunk Pottmann és társainak mostanában megjelent munkáira (lásd [71]), amelyben a szerzők függvényt vezetnek be síkok távolságának értelmezésére.

A dolgozat önálló eredményeken alapuló részei az 5.2. és 5.3., illetve a 5.4. fejezetekben található. Új eredménynek számít a 2.3. pontban ismertetett elemzés is, annyiban, hogy vizsgálatot folytattunk, hogyan lehetséges interpolációs B-spline görbét előállítani Kohonen-háló segítségével. A Plücker-féle koordináták használata és részletes ismertetése annyiban új eredmény, hogy kellő matematikai alapot biztosít a Kohonen-háló használatához. A dolgozat ötödik fejezetében található ábrák mind saját készítésű programok ouputjai, tehát az elméleti eredményeket a gyakorlatban is kipróbáltuk. A dolgozat terjedelmi korlátok miatt nem tartalmaz programokat, program részleteket. Meg kell említenünk, hogy a programozás rengeteg energiát és időt emésztett fel. Egyfelől úgy gondoltuk, hogy mindenképpen szükséges az elmélet gyakorlati alkalmazása, másfelől az a véleményünk, hogy informatikai szempontból a programozás során elért eredmények is értékesek.

A dolgozat szerzőjeként ezúton szeretnék köszönetet mondani témavezetőmnek, Dr. Szabó Józsefnek, akitől mérhetetlen türelmet, szakmai útmutatást és bátorítást kaptam. Köszönet illeti meg Dr. Hoffmann Miklóst is, aki segített a szakirodalom felkutatásában, továbbá szakmai segítsége nélkül a dolgozat.nem készülhetett volna el.

2. B-spline görbe és felület

Ebben a fejezetben áttekintjük a Computer Aided Geometric Design (CAGD) talán az egyik legnépszerűbb görbéjét, a B-spline-t. Irodalomként felhasználom egyrészt G. Farin [24] és Juhász Imre [45], [52] munkáit, másrészt jól használhatónak tartjuk még F. Yamaguchi [89], Rogers és Adams [76], és természetesen Piegl és Tiller [67] könyvét. A B-spline görbék alapjául szolgáló úgynevezett B-spline alapfüggvények már régóta ismertek. N. I. Lobacsevszkij már a XIX. században vizsgálta azok egy esetét (speciális csomóérték sorozat fölött definiált függvényeket, lásd Rényi [74], 165. o.). Ezeket J. Schoenberg 1946-ban statisztikai adatok simítására használta, az ő cikkétől [79] származtatjuk a spline approximáció modern elméletének kezdetét. A B-spline görbék tervezése a hetvenes évek elején válik használhatóvá. Ehhez hozzájárult a B-spline alapfüggvények rekurzív tulajdonságát felfedező C. de Boor [11], M. Cox [19] és L. Mansfield. Továbbá W. Gordon és R. Riesenfeld [29] munkásságának köszönthetjük , hogy a csak approximációs szempontból vizsgált függvényeket paraméteres görbék előállítására és görbék tervezésére is használják. Jelentős munkásságot fejtett ki ebben a témakörben W. Boehm [10] is.

Az interpolációs és approximációs görbék valamint felületek előállítására két különböző stratégiát ismerünk. Az egyik lehetőség, hogy az összes pontot és esetleges egyéb adatot (pl. érintők, második deriváltak stb.) figyelembe véve egyetlen görbét vagy felületet határozunk meg. Ilyen például a Lagrange, Newton vagy a Hermit interpoláció. A másik lehetőség az, amikor egymáshoz kapcsolódó görbeívekből állítjuk elő a görbét. Az utóbbi esetben előállított görbét szplájnnak (spline) nevezzük. A spline eredetileg hajóépítők által használt flexibilis vonalzót, gyakorlatilag egy hajlékony lécet vagy pálcát jelentett. A B-spline görbe a Bézier görbéhez hasonlóan approximáló görbe. Mivel végtelen sok megoldása lehet egy approximációs feladatnak, és nem tudunk megadni univerzálisan, minden szempontból optimális módszert, ezért létjogosultsága van több módszernek. A B-spline görbe jóval több szabadsági fokkal rendelkezik, mint a Bézier görbe. A B-spline görbe használatának előnye az is, hogy kevesebb kontrollpontra van szükség, mint a Bézier görbe esetében, azaz kevesebb adatot kell tárolni a számítógépen, és a Bézier görbe egy speciális esete a B-spline görbének.

2.1. Definíció

A dolgozatban a B-spline görbe megadására az alábbi definíciókat használjuk:

1. Definíció. *Tekintsük az* $u_i \leq u_{i+1}$, $u_i \in \mathbb{R}$ $(i = 0, \ldots, n+k)$ *skalárokat, ahol* $n, k \in \mathbb{N}$ *és* $n \geq k$. *Az*

$$N_i^1(u) = \begin{cases} 1, & ha\ (u_i \leq u < u_{i+1})\,^1 \\ 0 & egyébként, \end{cases}$$

$$N_i^k(u) = \frac{(u - u_i)}{(u_{i+k-1} - u_i)} N_i^{k-1}(u) + \frac{(u_{i+k} - u)}{(u_{i+k} - u_{i+1})} N_{i+1}^{k-1}(u) \qquad (1)$$

rekurzióval definiált függvényt normalizált B-spline alapfüggvénynek nevezzük, az u_i *skalárokat pedig csomóértékeknek (knot values) vagy csomóvektornak (knot vector). Az előforduló* $\frac{0}{0}$*-t definíció szerint* 0*-nak tekintjük.*

Az $N_i^k(u)$ normalizált B-spline alapfüggvény tehát egy legfeljebb $(k-1)$-edfokú polinom (a rendje k). A $\frac{0}{0}$ akkor fordulhat elő, ha a szomszédos csomóértékek egybeesnek. Megjegyezzük, hogy némely irodalomban a definícióban szereplő indexhatár nem 1-től, hanem 0-tól indul, ekkor a nevezőkben k index helyére $k+1$ kerül, továbbá a függvény fokszáma k lesz.

A Bézier[2] görbéhez hasonlóan határozzuk meg a B-spline görbét is.

2. Definíció. *Azt a görbét, amely*

$$\mathbf{r}(u) = \sum_{i=0}^{n} \mathbf{d}_i N_i^k(u), 2 \leq k \leq n+1,$$

alakban felírható, $k-1$*-ed fokú (vagy* k*-adrendű) B-spline görbének nevezzük, ahol* $N_i^k(u)$ *a* $k-1$*-ed fokú normalizált B-spline alapfüggvény*

$$\mathbf{U} = (u_0, u_1, \ldots, u_n, u_{n+1}, \ldots, u_{n+k})$$

csomóvektorral. A $\mathbf{d}_i$ $(i = 0, \ldots, n)$ *pontokat kontroll- vagy de Boor-pontoknak, az általuk meghatározott poligont pedig kontroll- vagy de Boor-poligonnak nevezzük.*

[1] Nem megengedett az egyenlőség, mert ha $u_i = u_{i+1}$ akkor $N_i^1(u) \equiv 0$.

[2] A Bézier görbéket terjedelmi okokból a dolgozatban részletesen nem ismertetjük, lásd P. Bézier könyvét [7].

Terjedelmi okokból nem tárgyaljuk a B-spline görbe tulajdonságait, viszont megemlítjük, hogy a terület alapos tanulmányozása elengedhetettlen a megfelelő programok elkészítése során.

2.2. Cox-de Boor algoritmus

A görbe alőállítására most már készíteni tudunk programot. Mivel a polinominális alakban megadott rekurzív függvény számítása rosszul kondicionált, ezért javasolt a Bézier görbénél tanult lineáris műveleteket tartalmazó de Casteljau algoritmushoz hasonló Cox-de Boor algoritmus használata.

de Boor javasolt algoritmust arra a problémára, hogy B-spline görbe pontot állítsunk elő rekurzívan, lineáris műveletek segítségével. Kezdjük az (1) egyenletben definiált rekurzív formula átalakításával, feltéve, hogy $u \in [u_i, u_{i+1})$:

$$\begin{aligned}\mathbf{r}(u) &= \sum_{i=0}^{n} \mathbf{d}_i N_i^k(u) = \\ &= \sum_{i=0}^{n} \mathbf{d}_i \frac{(u-u_i)}{(u_{i+k-1}-u_i)} N_i^{k-1}(u) + \sum_{i=0}^{n} \mathbf{d}_i \frac{(u_{i+k}-u)}{(u_{i+k}-u_{i+1})} N_{i+1}^{k-1}(u).\end{aligned}$$

Indextranszformációt hajtunk végre a második tagon, $i = i-1$,amely a következőt eredményezi

$$\mathbf{r}(u) = \sum_{i=0}^{n+1} \frac{\mathbf{d}_i (u-u_i) + \mathbf{d}_{i-1}(u_{i+k-1}-u)}{(u_{i+k-1}-u_i)} N_i^{k-1}(u) =: \sum_{i=0}^{n} \mathbf{d}_i^{[1]} N_i^{k-1}(u)$$

ahol, $\mathbf{d}_{-1} = 0$ és $\mathbf{d}_{n+1} = 0$. Az újraindexelést folytatva kapjuk

$$\mathbf{r}(u) = \sum_{i=0}^{n+j} \mathbf{d}_i^{[j]}(u) N_i^{k-j}(u), \quad j = 1, \ldots, k-1,$$

ha

$$\mathbf{d}_j^{[0]}(t_s) = \mathbf{d}_i \quad (i = 0, ..., n)$$

akkor

$$\mathbf{d}_i^{[j]}(u) = (1-\lambda)\mathbf{d}_{i-1}^{[j-1]}(u) + \lambda \mathbf{d}_i^{[j-1]}(u), \quad j > 0$$

konvex lineáris kombináció, ahol

$$\lambda = \frac{u-u_i}{u_{i+k-j}-u_i}.$$

Ha az algoritmust folytatjuk addig, hogy $j = k-1$, akkor megkapjuk az $N_r^1(u)$ bázisfüggvényt, azaz $u \in [u_r, u_{r+1})$ esetén az $\mathbf{r}(u) = \mathbf{d}_r^{k-1}(u)$ függvényértéket.

Ezt a felosztási algoritmust *Cox-de Boor algoritmusnak* nevezzük. Abban az esetben, ha az u_i és u_{i+k-1} $(u_i < u_{i+k-1})$ multiplicitása $k-1$, azaz

$$u_i = u_{i+1} = \ldots = u_{i+k-2} \text{ és } u_{i+k-1} = u_{i+k} = \ldots = u_{i+2k-3},$$

akkor speciális esetként a de Casteljau algoritmust kapjuk, következésképpen a B-spline görbe speciális eseteként a Bézier görbét. Bizonyítást lásd [52].

Figyelembe véve azt a tényt, hogy adott $u \in [u_r, u_{r+1})$ paraméter esetén minden $N_k^i(u)$ eltűnik kivéve azon bázisfüggvény, ahol az indexekre igaz, hogy $r-(k-1) \le i \le r$. A Cox- de Boor algoritmus a következő sémával adható meg,

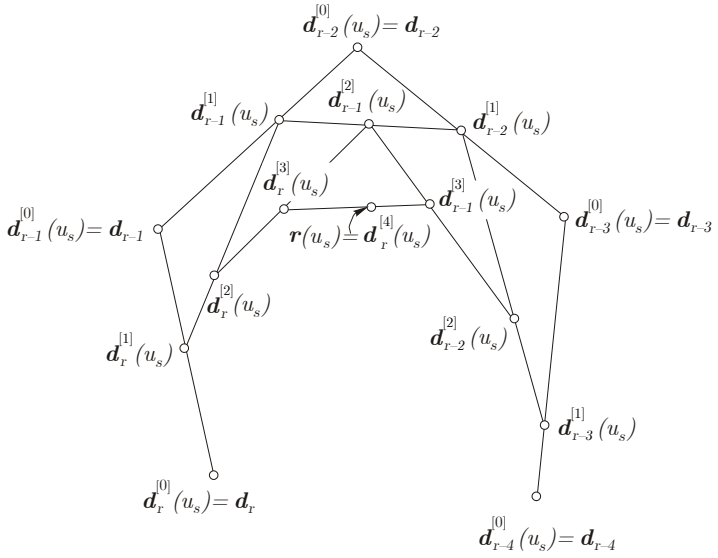
$$\begin{array}{lllllll}
\mathbf{d}_{r-k+1} & = \mathbf{d}_{r-k+1}^0 & & & & & \\
 & & \mathbf{d}_{r-k+2}^1 & & & & \\
\mathbf{d}_{r-k+2} & = \mathbf{d}_{r-k+2}^0 & & \mathbf{d}_{r-k+3}^2 & & & \\
 & & & & \ddots & & \\
 & & & & & \mathbf{d}_{r-1}^{k-2} & \\
\vdots & & & & \cdots & & \mathbf{d}_r^{k-1} = r(u). \\
 & & & & & \mathbf{d}_r^{k-2} & \\
 & & & & \vdots & & \\
\mathbf{d}_{r-2} & = \mathbf{d}_{r-2}^0 & & \mathbf{d}_{r-1}^2 & & & \\
 & & \mathbf{d}_{r-1}^1 & & & & \\
\mathbf{d}_{r-1} & = \mathbf{d}_{r-1}^0 & & \mathbf{d}_r^2 & & & \\
 & & \mathbf{d}_r^1 & & & & \\
\mathbf{d}_r & = \mathbf{d}_r^0 & & & & &
\end{array}$$

Ez alapján a szerkesztés is könnyen elvégezhető, csak itt a szakaszok osztási aránya függ a csomóértékektől.

Mivel programozási szempontból nagyon fontos a Cox-de Boor algoritmus, ezért részletesen megadjuk az algoritmust:

Adatok:

- $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_n$, ahol $n \ge k-1$, $(n+1)$ de Boor pont,
  k a B-spline görbe rende (fokszám $= k-1$)



1. ábra. B-spline görbepont szerkesztése de Boor algoritmussal, $k = 5$

- Zárt görbe esetén, a (**??**) szerint megadott csomóértékeket érdemes kiterjeszteni jobbra és balra is, azaz bevezetni

$$\begin{aligned} u_{-1} &= u_0 + (u_{n+1} - u_n), \quad u_{-2} = u_{-1} + (u_n - u_{n-1}), \ldots, \\ u_{n+2} &= u_{n+1} + (u_1 - u_0), \quad u_{n+3} = u_{n+2} + (u_2 - u_1), \ldots, \end{aligned}$$

  értékeket

- Nyilt görbe esetén, a csomóértékek megadása legyen

$$\mathbf{U} = (u_0 = u_1 \cdots = u_{k-1}, u_k, u_{k+1}, \ldots, u_n, u_{n+1} = u_{n+2} = \cdots = u_{n+k}) .$$

A számítás menete:

Határozzuk meg a $u = u_s$ paraméter értékhez tartozó $\mathbf{r}(u_s)$ görbe pontot.

1. Keressünk olyan i értéket, amelyre igaz $u_i \leq u_s < u_{i+1}$.
2. Legyen $\lambda^1_j = (u_s - u_j) \,/\, (u_{j+k-1} - u_j)$,
   for $j = (i - k + 2)$ to i do
$$\mathbf{d}^{[1]}_j = (1 - \lambda^1_j)\mathbf{d}_{j-1} + \lambda^1_j \mathbf{d}_j$$

3. Ismételten alkalmazzuk a formulát:

   for $m = 2$ to $(k-1)$ do
   for $j = (i-k+m+1)$ to i do

$$\begin{aligned} \lambda_j^m &= (u_s - u_j) / (u_{j+k-m} - u_j), \\ \mathbf{d}_j^{[m]} &= (1-\lambda_j^m)\mathbf{d}_{j-1}^{[m-1]} + \lambda_j^m \mathbf{d}_j^{[m-1]}. \end{aligned}$$

   Ha uniform módon adjuk meg a csomóértékeket és a lépésköz 1, akkor

$$\lambda_j^m = \frac{u_s - u_j}{k-m}.$$

4. Végül számítsuk ki az $\mathbf{r}(u_s) = \mathbf{d}_i^{[k-1]}$ görbepontot.

2.3. Interpoláció B–spline görbével

A B-spline görbe approximáló görbe, de használhatjuk interpolációra is. Farin [24], Piegl [67] munkái alapján jómagam is elemeztem ezt a problémát [56]-ben és [37]-ben. Tehát keresünk egy olyan adott fokszámú B-spline görbét, azaz határozzuk meg kontrollpontjait és csomóértékeit, amely az adott paraméterértéknél az előre megadott ponton áthalad. Ha $\mathbf{Q}_r,\ r = 0, ..., j$ pontok illeszkednek egy k-ad fokú nem racionális B-spline görbére, akkor kielégítik a következő $(j+1)$ egyenletből és $(n+1)$ ismeretlenből álló egyenletrendszert

$$\mathbf{Q}_r = \mathbf{Q}(\overline{u}_r) = \sum_{i=0}^{n} \mathbf{d}_i N_{i,k}(\overline{u}_r), \tag{2}$$

ahol igaz az, hogy $2 \le k \le n+1 \le j$, és az $\overline{u}_r$ paramétereket hozzárendeljük minden $\mathbf{Q}_r$-hez, továbbá választunk egy megfelelő $\mathbf{U} = u_0, ..., u_m$ csomóvektort. A $\mathbf{d}_i$ kontrollpontok lesznek az egyenletrendszer ismeretlenei. Az egyenletrendszernek egy együttható mátrixa van és természetesen s megoldáshalmaza, ha $\mathbf{d}_i$-kontrollpontoknak s koordinátája van, azaz s a $\mathbf{Q}_r$ pontok koordinátáinak a száma 2, 3 vagy 4.

A (2) egyenletrendszert kifejtve a következőt kapjuk

$$\begin{aligned}
\mathbf{Q}(\overline{u}_0) &= N_{0,k}(\overline{u}_0)\,\mathbf{d}_0 + N_{1,k}(\overline{u}_0)\,\mathbf{d}_1 + \ldots + N_{n,k}(\overline{u}_0)\,\mathbf{d}_n \\
\mathbf{Q}(\overline{u}_1) &= N_{0,k}(\overline{u}_1)\,\mathbf{d}_0 + N_{1,k}(\overline{u}_1)\,\mathbf{d}_1 + \ldots + N_{n,k}(\overline{u}_1)\,\mathbf{d}_n \\
&\vdots \\
\mathbf{Q}(\overline{u}_j) &= N_{0,k}(\overline{u}_j)\,\mathbf{d}_0 + N_{1,k}(\overline{u}_j)\,\mathbf{d}_1 + \ldots + N_{n,k}(\overline{u}_j)\,\mathbf{d}_n
\end{aligned}$$

amelyet átírhatunk (a szokásos módon) mátrixalakba

$$[\mathbf{Q}] = [\mathbf{N}]\,[\mathbf{d}]\,.$$

Ha $2 \leq k \leq n+1 = j$, azaz $(n+1)$ egyenletből és $(n+1)$ ismeretlenből áll az egyenletrendszer, akkor $[\mathbf{N}]$ mátrix kvadratikus, és az eredmény mátrixszorzással nyerhető, azaz

$$[\mathbf{d}] = [\mathbf{N}]^{-1}\,[\mathbf{Q}]\,. \tag{3}$$

Tehát a probléma megoldása $\overline{u}_r$ és $\mathbf{U}$ meghatározására korlátozódik, mivel megválasztásuk befolyásolja a görbe alakját. Tegyük fel, hogy a paramétertartomány $u \in [0,1]$. A paraméterek kiválasztására több módszer ismert, például ekvidisztáns (egyenlőközű), ívhossz szerinti és a centripetális mód. Számítási igényeket és a kapott görbe alakját is figyelembe véve a centripetális módszer a javasolt. Legyen

$$l = \sum_{r=1}^{n} \sqrt{|\mathbf{Q}_r - \mathbf{Q}_{r-1}|},$$

tudva, hogy $\overline{u}_0 = 0$ és $\overline{u}_n = 1$,

$$\overline{u}_r = \overline{u}_{r-1} + \frac{\sqrt{|\mathbf{Q}_r - \mathbf{Q}_{r-1}|}}{l}, \quad r = 1, ..., n-1.$$

Ez a módszer jobb eredményt ad, amikor a pontok hirtelen élesen változnak, tehát éles "kanyart" várunk a görbétől. Ajánlott még a centripetális módszer kombinálása az átlagolás technikájával, azaz

$$u_0 = ... = u_k = 0, \quad u_{m-k} = ... = u_m = 1$$

$$u_{j+p} = \frac{1}{p} \sum_{i=j}^{j+p-1} \overline{u}_i, \quad j = 1, ..., n-p.$$

Ez a kombináció vezet a (2) egyenletrendszerhez, és biztosítja a következő előnyös tulajdonságot: $N_{i,k}(\overline{u}_r) = 0$ ha $|i - r| \geq k$. Az egyenletrendszert megoldhatjuk (3) szerint is, de a mátrix inverzének a meghatározása nagy dimenziószám esetén rosszul kondicionált. Javasolt a teljes főelemkiválasztássos Gauss-Jordan elimináció használata. Természetesen érdemes az LU dekompozíció technikájával alsó és felső háromszögalakú komponensekre bontani az együtthatómátrixot (lásd [72]), vagy valamilyen iterációs technikát alkalmazni (pl. Seidel-iteráció).

Bár az eredmény görbe mindenhol C^{k-2}-folytonos, előfordulhat, hogy nem lesz elég szép sima és tartalmazhat váratlan hullámokat és kilengéseket. A nem várt kilengések elkerülése végett keressünk kevesebb $\mathbf{d}_i$ kontrollpontot mint $\mathbf{Q}_r$ input pontot, azaz $2 \leq k \leq n + 1 < j$. Mivel ebben az esetben $[\mathbf{N}]$ már nem kvadratikus, ezért a (2) egyenletrendszer átírva kompakt mátrix alakba a következőt eredményezi:

$$[\mathbf{Q}] = [\mathbf{N}]\,[\mathbf{d}]$$

$$[\mathbf{N}]^T\,[\mathbf{Q}] = [\mathbf{N}]^T\,[\mathbf{N}]\,[\mathbf{d}]$$

$$[\mathbf{d}] = \left[[\mathbf{N}]^T\,[\mathbf{N}]\right]^{-1}\,[\mathbf{N}]^T\,[\mathbf{Q}]$$

Természetesen nem minden esetben kapunk megoldást, mivel a probléma túlhatározott, de ilyenkor numerikus úton kezelhető a probléma.

2.4. Racionális B-spline görbe

A racionális Bézier görbékhez hasonlóan lehet bevezetni a racionális B-spline görbét is. Tehát térbeli parabolák vetületeként (pl. a $z = 1$ síkra) állíthatunk elő kúpszeleteket, azaz, kúpszeletek térbeli másodrendű Bézier görbék centrális vetületei lesznek. Ezen megközelítés általánosítása a racionális Bézier görbe.(Hoschek[46] és Juhász [45] alapján.)

A jelenlegi tervezési rendszerekben a nem uniform racionális B-spline görbék, amelynek szokásos rövidítése NURBS (nonuniform rational B-spline), biztosítják a legtöbb szabadságot, és a legtöbb alakváltoztatási lehetőséget a tervezők számára. Az előző pont alapján teljesen analóg módon megadhatjuk a racionális Bézier görbéhez hasonlóan a racionális B-spline görbét is. Szemléletesen tárgyalva a következőt mondhatjuk, a négydimenziós térben egy x, y, z, w koordinátarendszerben adott egy B-spline görbe $\begin{bmatrix} w_i\mathbf{d}_i \\ w_i \end{bmatrix}$

kontrollpontjaival. Ezt a görbét centrálisan levetítjük az origóból a $w = 1$ hipersíkra (3 dimenziós) és vetületként racionális B-spline görbét kapunk. Dimenziószámokat változtatva egyéb esetekhez jutunk, pl eggyel csökkentve síkbeli racionális B-spline görbét kapunk.

3. Definíció. *Az*

$$\mathbf{s}(u) = \frac{\sum_{i=0}^{n} w_i \mathbf{d}_i N_i^k(u)}{\sum_{j=0}^{n} w_j N_j^k(u)}$$

kifejezéssel definiált görbét $(k-1)$*-edfokú racionális B-spline görbének nevezzük. A* $\mathbf{d}_i$ $(i = 0, 1, \ldots n)$ *pontokat a görbe kontrollpontjainak, az általuk definiált poligont kontrollpoligonnak, a* w_i *skalárokat pedig súlyoknak nevezzük. Az* $N_i^k(u)$ *függvény* $(k-1)$*-edfokú normalizált B-spline alapfüggvény, amelyhez meg kell adnunk az* $\mathbf{U} = (u_0, u_1, \ldots, u_n, u_{n+1}, \ldots, u_{n+k})$ *csomóvektort is, amelyben a csomóértékekre mint skalárokra igaz, hogy* $u_i \leq u_{i+1}$ *és* $u_i \in \mathbb{R}$ $(i = 0, \ldots, n+k)$.

A w_i súlyokat a szingularitások elkerülése végett pozitívnak szokták választani, mivel ebben az esetben nem csak a nem racionális B-spline görbék kedvező tulajdonságait tartja meg (local support, konvex burok , folytonosan differenciálhatóság), hanem még újabb alakváltoztatási lehetőséghez jutunk. A racionális Bézier görbéhez hasonlóan a w_i súlyokat alakparamétereknek nevezzük, mivel ha növeljük egy w_i értékét akkor a görbe a megfelelő $\mathbf{d}_i$ kontrollpont felé fog mozogni. Ez a tulajdonság hasonló ahhoz, amelyet a csomóérték multiplicitással érünk el, de nem egyezik meg azzal. Megjegyezzük, hogy egyszerű átalakításokkal megkaphatjuk a racionális Cox- de Boor algoritmust is (lásd [45]).

2.5. B-spline felület

Ebben a pontban a B-spline felületet definiáljuk, mivel további kutatásaink szempontjából a geometria tervezésnél a legnépszerűbb NURBS felületekkel törődünk. Kindulásként tekintsük a következő felületek osztályát

$$\mathbf{s}(u, v) = \mathbf{M}(u)\mathbf{g}(v), u \in [u_0, u_1], v \in [v_0, v_1],$$

ahol $\mathbf{g}(v)$ egy tetszőleges görbe, $\mathbf{M}(u)$ pedig egy 4×4-es mátrix. Az így kapott felület nem más, mint az u paramétertől fuggő egyparaméteres görbesereg által súrolt felület. A $\mathbf{g}(v)$ görbe térbeli mozgása során egy felületet

ír le. Megengedett, hogy a görbe alakja is változzon a mozgás során. Pontosan az utóbbi tulajdonsággal származtatható Bézier és B-spline felület is.

Tegyük fel, hogy az $\mathbf{a}_i$ $(i = 0, 1, \ldots, n)$ kontrollpontjával adott B-spline görbét mozgatjuk, úgy, hogy feltételezzük, hogy a B-spline görbe kontrollpontjai ugyancsak B-spline görbén mozognak. Jelöljük $\mathbf{b}_{ij}$-vel az $\mathbf{a}_i$ kontrollpontok pályáját meghatározó B-spline görbe kontrollpontjait $j = 0, 1, \ldots, m$, azaz $\mathbf{b}_{i0} = \mathbf{a}_i$, $(i = 0, 1, \ldots, n)$. Ebben az esetben a mozgó B-spline görbe

$$\mathbf{a}(u) = \sum_{i=0}^{n} \mathbf{a}_i N_i^{k_1}(u),$$

amelynek kontrollpontjai az

$$\mathbf{a}_i(v) = \sum_{j=0}^{m} \mathbf{b}_{ij} N_j^{k_2}(v)$$

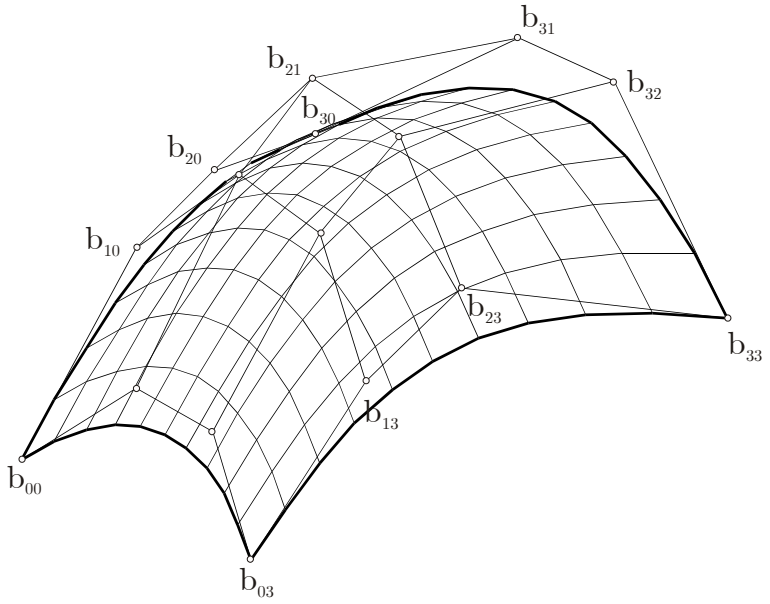
B-spline görbén mozognak. A két alakot kombinálva a következő felületet kapjuk:

$$\mathbf{s}(u, v) = \sum_{i=0}^{n} \left(\sum_{j=0}^{m} N_j^{k_2}(v) \right) N_i^{k_1}(u) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{b}_{ij} N_i^{k_1}(u) N_j^{k_2}(v).$$

Ezt a felületet B-spline felületnek, vagy tenzori szorzattal előállított B-spline felületnek nevezzük. A $\mathbf{b}_{ij}$ pontokat a felület *kontrollpontjainak*, az általuk meghatározott hálót pedig *kontrollhálónak* nevezzük. Megjegyezzük, hogy az elöbbihez hasonlóan más görbék tenzori szorzataként is származtathatunk felületet. Az

$$\mathbf{s}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{b}_{ij} F_i(u) G_j(v)$$

felület tenzori szorzattal előállított felület (tensor product surface), ahol a $F_i(u)$ és a $G_j(v)$ különböző bázisfüggvények, például Bézier felület esetén Bernstein polinomok, de lehetnek például Lagrange, racionális Bézier vagy racionális B-spline alapfüggvények is.



2. ábra. B-spline felület és kontrollháló

Racionális B-spline felület

A racionális felület származtatása pontosan ugyanúgy történik mint a nem racionális esetben. Tehát a racionális B-spline felület racionális B-spline görbék tenzori szorzataként előállított felületek osztályába tartozik. Ennek megfelelően

$$\mathbf{s}\left(u,v\right)=\sum_{i=0}^{n}\sum_{j=0}^{m}\mathbf{d}_{ij}w_{ij}\frac{N_i^k(u)}{\sum_{r=0}^{n}w_{rj}N_r^k(u)}\frac{N_j^l(v)}{\sum_{r=0}^{m}w_{ri}N_r^l(v)}$$

ahol a $\mathbf{d}_{ij}$ pontokat a felület kontrollpontjainak az általuk meghatározott hálót pedig kontrollhálónak nevezzük. A $0<w_{ij}\in\mathbb{R}$ skalárokat súlyoknak nevezzük. Az $N_k^i\left(u\right)$ és $N_j^l\left(v\right)$ pedig a normalizált alapfüggvényeket jelöli, az $u_i\leq u_{i+1}$ és $v_j\leq v_{j+1}$ csomóértékekkel. A racionális felületek tulajdonságai megegyeznek a nem racionális felületek tulajdonságaival, csak az alakváltoztathatóság eszköztára bővül a felületek kontrollpontjainak súlyaival.

3. Rendezetlen adatok modellezése B-spline felülettel

3.1. Alapprobléma

Fontos tárgyalnunk néhány kutató eredményeit, hogy tisztábban lássuk a saját eredményeinket. A reverse engineering kutatási területet megalapozó munka Várady Tamás, és szerzőtársai (R. Martin és J. Cox) 1997-es (lásd [85]) cikkéhez fűződik. Weiss, Andor, Renner és Várady 2002-es cikke alapján be tudunk számolni a legfrissebb eredményekről is (lásd [88]). Annak ellenére, hogy igen intenzív és kiterjedt kutatások folynak paraméteres felületek illesztésére, a megjelent megoldások gyakran hibáznak. A felületillesztés problémájának megfogalmazása ([88]) nyomán a következő:

Adottak a $\mathbf{p}_r \in \mathbb{R}^n$ véges ponthalmaz, (amelyet például lézer szkenneléssel nyertünk) és az $\varepsilon_r \in \mathbb{R}_+$ tűrésértékek (toleranciaértékek), keresünk olyan megfelelő $\mathbf{S}$ felületet, amely megközelíti az adott $\mathbf{p}_r$ pontokat megadott ε_r hibaértéken belül

$$\|\mathbf{S}_{\mathbf{p}_r} - \mathbf{p}_r\| \leq \varepsilon_r, \quad r \in \{1, \ldots, m\}\,. \tag{4}$$

$\mathbf{S}_{\mathbf{p}_r}$ jelöli azt a pontot a felületen, amelyet hozzárendelük $\mathbf{p}_r$-hez. Ez a formula természetesen nem vezet a megoldáshoz, mert számos lehetőség van $\mathbf{S}$ kiválasztására, és a megoldás száma lehet sok, de lehet nulla is.

A szabadformájú felületek előállítására talán a legnépszerűbb választás a NURBS felületek használata. Feltételezzük, hogy minden $\mathbf{p}_r$ adatponthoz hozzárendeltünk egy felületi pontot, amely a paramétertartomány felett (u_r, v_r) paraméterpárral és w_r súllyal definiált. A w_r skalár a pont relatív fontosságát fejezi ki. Ezekkel a feltételekkel a súlyozott legkisebb négyzetek módszerének formulája:

$$\sum_{r=1}^{m} w_r^2 \left\|\mathbf{S}\left(u_r, v_r\right) - \mathbf{p}_r\right\|^2 \to \min\,. \tag{5}$$

Ha Euklideszi metrikát használunk a pontok távolságának meghatározására, és ha rögzítjük a csomóértékeket és a paraméterértékeket, akkor a (5) általában megoldható. Természetesen a B-spline felület kontrollpontjai lesznek az ismertlenek. A (5) minimumának a meghatározása nem feltétlenül egyezik

meg a (4)-ben meghatározott tolerancia feltétellel és nincs garancia a megfelelő simaságú felületre sem. A megfelelő esztétikus felület biztosítására a fenti (5) kifejezést ki kell bővíteni úgynevezett *simasági feltételekkel*.

Bár a megoldás egyszerűnek tűnik, de a (4)-ben megadott szoros tolerancia feltétellel az *automatikus felületillesztés* nehéz feladat. Az első probléma az, hogy a tenzori szorzattal előállított B-spline felületek négyszögháló fölött vannak értelmezve, viszont az input adatok pontfelhője gyakran nem konvex n-oldalú tartományokból származik, amely ráadásul számos lyukat is tartalmazhat a belsejében. A második probléma az, hogy mért értékekkel dolgozunk. A mérési pontatlanságból adodóan az adatok gyakran „zajosak" és egyenlőtlen eloszlásúak. A harmadik probléma a parametrizálásból ered, a paraméterek mesterségesek, de lényeges belső mennyiségei a felületnek. A felületillesztés lépései alatt úgy kell megbecsülnünk a parmetrizálást, hogy a felület még valójában ismeretlen számunkra. A távolsági feltétel (tolerancia küszöb) és a simasági feltétel között általában feloldhatatlan ellentét húzódik. Ha „túl sima" a felület, akkor az távolról sem felel meg a (5)-as feltételnek, ha viszont a felület „túl pontos", akkor váratlan és nem kívánatos hullámzások fordulnak elő. Tehát ezen feltételek között navigálva kell meghatároznunk a kívánt felületet.

A feltételek szokásos megadása

$$F_{comp}(s) = F_{lsq}(s) + \lambda F_{smooth}(s),$$

ahol az első tag a pontossági a második tag a simasági feltétel. A λ a simaságot súlyózó tényező, amely meghatározza a két feltétel viszonyát. A nagyobb λ simább felületet biztosít nagyobb hibával, amíg a kisebb λ-val számított felület hullámzobb lesz és pontosabb. Gyakori megadása (lásd [88]) a simasági függvénynek a következő

$$F_{smooth}(s) = \iint\limits_{\Omega} \left(\mathbf{S}_{uu}^2 + 2\mathbf{S}_{uv}^2 + \mathbf{S}_{vv}^2\right) dudv,$$

ahol $\mathbf{S}(u,v)$ 2.5. pontban definiált NURBS felület, azaz

$$\mathbf{S}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} \mathbf{s}_{ij} N_i^{k_1}(u)\, N_j^{k_2}(v).$$

Összefoglalva a következő követelményeket fogalmazhatjuk meg: (i) a keresett felületnek adott hibahatáron belül kell közelítenie az input ponto-

kat, esztétikusnak (simának) és kiszámíthatónak kell lennie, azaz, görbüljön ahol szükséges, de sehol máshol; (ii) az is elvárható, hogy elfogadható módon kiterjeszthető, bővíthető legyen a felület olyan régiókba is, ahol már nincs megadva input pont; (iii) olyan redundancia nélküli automatikus felület-reprezentációt kell kapnunk, amely csak elfogadható számú kontrollponttal rendelkezik.

A CAD-ben már régóta ismertek olyan matematikai módszerek, amelyek input pontok felhőjére szabad formájú felületek illesztésével foglalkoznak (lásd Hayes és Halliday 1974-ben publikált cikkét [32]). Mindemellett csak az utolsó 10-12 évben születtek hatékony és stabil módszerek. Nem célunk ennek az igen széles irodalomnak a részletes ismertetése, amelynek összefoglalása –kitérve a rekonstrukicó alap statégiáira, a simasági feltételekre és a kezdeti parametrizációra– megtalálható Weiss, Andor, Renner és Várady 2002-es cikkében [88]. Ebben a dolgozatban rövid áttekintését adjuk a kezdeti parametrizálás kérdésének.

3.2. Paraméterezés

A felületrekonstrukció kulcskérdése jó paraméterértékek hozzárendelése az adatpontokhoz, azaz, találjunk olyan pontokat a sima felületen, amelyek közel vannak az adatpontokhoz. A problémának két aspektusa van: (i) a folyamat elején elfogadhatóan jó paraméterértékeket, azaz kezdeti parametrizálás kell találni; (ii) a folyamat során nem csak a felületet kell optimalizálni, hanem a paraméterértékeket is. Ez útobbi, azaz a paraméterkorrekció, különösen fontos szerepet játszik a szigorú feltételekhez kötött approximációk esetében. Létezik mindkét aspektust megoldó módszer is, amely azonban csak bizonyos kedvező körülmények teljesülése esetén alkalmazható.

Szabálytalan távolságban lévő véletlen eloszlású adatpontok esetén, a parametrizáció inicializálására szokásos módszer létrehozni egy alap felületet (base surface), amely felület egy durva közelítése az input adatok pontfelhőjének. Ez a felület lehet egy sík, egy henger vagy egy bilineárisan súlyozott Coons-folt (lásd Hermann és szerzőtársai [35]), amely a pontfelhőből nyert geometriai információk alapján állítható elő. Itt említem meg, hogy a 5.1. pontban ismertetett szerzőtársaimmal együtt kifejlesztett módszerrel egyéb formájú bázisfelület is nyerhető. Ma és Kruth [60] a bázisfelület előállítására a négy határgöbével együtt interaktívan definiált görbe részeket használt.

Az így nyert felület szintén az input pontok egy durva közelítése. Ezek után a kezdeti paraméterértékeket úgy kapjuk meg, hogy az input pontokat rávetítjük a bázisfelületre. Az említett módszerek esetében számos probléma merül fel a bázisfelület megkonstruálása során, például szabálytalan élesen görbülő részekből épül fel, vagy az input pontokat nem tudjuk egyértelműen rávetíteni a bázisfelületre. A Coons folt módszer csak akkor alkalmazható, ha a pontfelhőből meg tudjuk határozni a négy megfelelő határgörbét, mely igen nehéz feladat egy n oldalú ponttartomány esetén.

A paraméterek inicializálásának másik módszere az input pontfelhő térbeli háromszögelésén alapul. Általában a topologikusan azonos síkbeli háromszögelés előállítható interaktív lépéseken keresztül. Az így kapott síkbeli háromszögelés során nyert paramétertartomány pontjainak koordinátáiból nyerhetőek a kezdeti paraméterértékek. A publikált módszerek a tér síkra való leképezésében különböznek. Harmonikus parametrizációk minimalizálják az élektől való távolságot (Eck és Hadenfeld [23]), a Floater-alakot megtartó elképzelés baricentrikus leképezés használatával megtalálja a belső háromszögek új csúcsait (lásd Floater [25]). Mindkét módszer esetén először szükség van egy megfelelő konvex tartományra, amely a térbeli pontok sarokpontjainak leképezésével nyerhető. Greiner és Hormann több cikkben is foglalkoznak a problémával (lásd [30] és [44]). A 2000-ben megjelent cikkükben már egy tökéletesített technikát mutatnak be – the Most Isometric Parametrisation – , ahol a határokat már nem kell előre rögzíteni és természetesebb után alkulnak ki. Ezek a tecnikák amellett, hogy nagyszámú input pont esetén lassúnak tűnnek, meglehetősen korlátozottak, ha lyukakat vagy konkáv részeket tartalmaz az input pontfelhő.

Végezetül azt a következtetést vonhatjuk le, hogy nincs biztos, univerzális módszer paraméterinicializálás problémájára. Hoffmann M. és Várady L. szerzőtársaimmal (lásd [43] és [86]) adunk egy újabb módszert, amellyel pontfelhőre lehet illeszteni felületet Kohonen neurális háló segítségével, amely akár a bázisfelület (base surface) szerepét is betöltheti. Véleményünk szerint, a neurális hálók előnyös tulajdonságait kihasználva gyorsan lehet jó bázisfelületet előállítani. A neurális halók technikáját alkalmazó módszert a dolgozat későbbi fejezeteiben találhatjuk meg. A klasszikus változatot a 5. pontban, a dinamikus változatot pedig a 5.2. pontban ismertetem.

3.3. Vonalfelületek modellezése

Az előző fejezetben általánosan kezeltük a pontfelhőre történő felületillesztési problémát, azaz szabad formájú felületeket állítottunk elő. Most azzal foglalkozunk, hogyan lehet térbeli ponthalmazban egyeneseket keresni, tehát vonalfelületet szeretnénk illeszteni a pontfelhőre.

A klasszikus geometria jól ismert területei a vonalfelületek és a kifejthető felületek, mindemellett a számítógéppel segített tervezésben (Computer Aided Design) és a számítógéppel segített gyártásban (Computer Aided Manufacturing) szintén sok alkalmazása van e felületeknek. Minden ilyen alkalmazás alapvető kérdése, hogy ha adott egy felület, akkor hogyan lehet közelíteni vonalfelülettel? Ha a szükséges pontossággal meg tudjuk ezt tenni, akkor a vonalfelülettel tudjuk helyettesíteni az eredeti felületet, és egyszerűsödik a gyártás vagy növekszik a pontosság. Sok esetben csak egy ponthalmazt kapunk a felületről, amit például egy lézer szkenner adott meg nekünk. Megtehetjük, hogy egy ponthalmaz (pontfelhő) legyen az inputja az általunk tárgyalt algoritmusoknak. A felületek megadásának szokásos módja, a különböző típusú spline felületek használata. Ilyen felületek lehetnek például a Bézier, B-spline vagy a NURBS felületek. Tehát a kívánatos eredmény az lenne, ha a vonal és kifejthető felületeket, mint spline felületeket tudnánk kezelni. A szakirodalomban számos módszert találunk, amelyekben ilyen típusú felületeket használnak felület approximáció vagy felület interpoláció esetén.

A továbbiakban e módszereket ismertetjük, majd később rámutatunk a hiányosságokra is, és megoldást adunk a problémára.

Számos cikk foglakozik vonal és kifejthető felületek létrehozásával CAD környezetben. Egy síkgörbe, és egy párhuzamos síkban lévő két pont között Aumann [3] konstruált kapcsolódó kifejthető felületet, alkalmas kubikus és kvadratikus Bézier görbék segítségével. Kifejthető felületek projektív geometriai megközelítése Ravani és társszerzője Bodduluri [8]–[9] nevéhez fűződik. A duális Bézier és B-spline felületek használatát javasolta Hoschek [50], amelyek igen hasznos eszközök kifejthető felületek megadására, interpolálására és approximálására [68], [47]. Frey [28] összehasonlító áttekintést ad az alkalmazásokról, és kiterjeszti Aumann [3] megközelítését.

Ebben a fejezetben részletesen áttekintjük a [16]-ban Horng-Yang Chen és Helmut Pottmann által tárgyalt módszert. Az algoritmus három lépés-

ben ad módszert arra, hogyan állíthatunk elő tenzori szorzattal megadott B-spline felülettel approximáló vonalfelületet:

1. Keressük meg az egyenesek diszkrét rendszerét a ponthalmazzal megadott felület közelében.

2. Konstruáljunk az egyenesekből approximáló B-spline felületet.

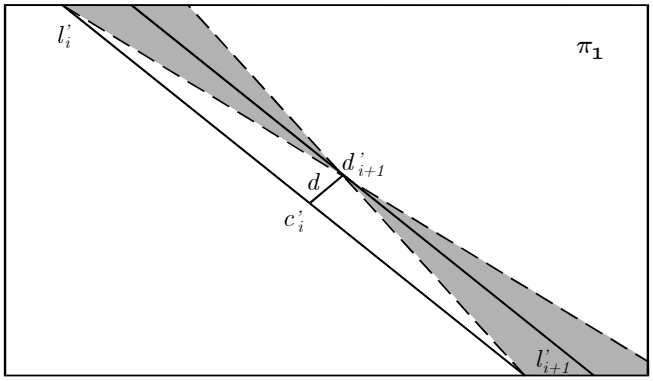
3. Ha lehetséges, akkor valamilyen ismert módszer segítségével javítsunk az approximáción. Kombináljuk a legkisebb négyzetek módszerét a függvény minimalizálással és a paraméterkorrekcióval.

Ezen lépések egy részét tárgyaljuk a következő részben. Az 1. lépés hasonló Hoschek és Schneider [48] által közölt algoritmushoz. A 2. lépés elvezet az egyenesek terében végzett approximációhoz a Klein leképezés segítségével. A 3. lépéssel nem foglakozunk, mert a később leírt új eredményeinket már az ebben a lépésben megadott módszerek nem befolyásolják. Hoschek által felvetett probléma, hogy adott a P_j pontfelhő (amely például lézer szkenneléssel keletkezett), és keressünk olyan kifejthető vagy vonal felületet, amelynek alkotói olyan közel vannak a megadott pontokhoz amilyen közel csak lehetséges. A probléma megoldásának egyik kritikus pontja a megfelelő hibamérés. Egyenesek távolságának a mérésére felhasználhatjuk a Horng-Yang Chen és Helmut Pottmann által használt módszert.

Keressük meg az egyenesek diszkrét rendszerét

Az algoritmus inputja a paraméteres vagy implicit módon megadott $\Phi \in \mathbb{R}$ felület. Ha csak szétszórt pontok adatai ún. pontfelhő adott, akkor először felületet kell a pontokra illeszteni. Feltéve a pontok elégséges sűrűségét, a szakaszonként lineáris modell is megfelelő számunkra. Valójában minden esetben érdemes ilyen modellel dolgozni, mert így az algoritmus első részében növelni tudjuk a számítási sebességet.

Először foglalkozzunk azzal az esettel, amikor létezik egy olyan $\mathbf{e}_3$ vektor és egy $0 < \gamma_0 < \pi/2$ szög, hogy minden felületi normálisnak az $\mathbf{e}_3$-mal bezárt γ szögére igaz, hogy $0 < \gamma < \gamma_0$. Legyen a Descartes koordinátarendszer z-tengelye párhuzamos az $\mathbf{e}_3$ vektorral. Ekkor a felület megadható egy kétváltozós függvénnyel, amely az $[x, y]$ sík valamely D' tartománya fölött van értelmezve. Nem szükséges a függvény explicit meghatározása.



3. ábra. Az árnyékolás a választható területet mutatja

Az $[x, y]$ sík elemeit jelöljük vesszővel. Tekintsük a sík azon $l^{'}$egyeneseit, amelyek metszik a $D^{'}$tartományt. Minden ilyen $l^{'}$-höz hozzárendelünk egy térbeli l egyenest, nevezetesen azt, amelyik azt a görbét közelíti, amely az $[x, y]$ síkra merőleges és $l^{'}$-re illeszkedő sík metsz ki a felületből. Az említett görbét meghatározhatjuk a legkisebb négyzetek módszerével. Továbbá minden egyes $l^{'}$-höz hozzárendeljük az átlagos eltérés (hiba) négyzetét, amely jól tükrözi, hogy mennyire közelíti meg a megadott felületet.

Az egyenesek síkbeli rendszerének meghatározásához tekintsük a $D^{'}$ tartomány $d_0^{'}$ belső pontját. Ha a $D^{'}$tartomány konvex, akkor tekintsük a középpontját, egyéb esetben ügyeljünk arra, hogy $d_0^{'}$ ne legyen a határvonal közelében. Elég nagy szakaszokban metszik a $D^{'}$tartományt azon $l^{'}$ egyenesek, amelyek illeszkednek $d_0^{'}$-ra. Jelöljük $l_0^{'}$-val azt az egyenest, amelyik illeszkedik $d_0^{'}$-ra és a legkisebb hibával rendelkezik. Ezek után az algoritmusnak $l_0^{'}$ lesz a kiinduló szegmense, és a kezdő iránya pedig, az $l_0^{'}$-ra merőleges irány. Az $l_i^{'}$-ből úgy határozhatjuk meg $l_{i+1}^{'}$-et, hogy figyelembe vesszük az összes olyan egyenes szegmenst, amely illeszkedik az $l_i^{'}$ szegmens $c_i^{'}$középpontjától merőleges irányban d távolságra lévő $d_{i+1}^{'}$-re, és nem metszi az $l_i^{'}$ szegmenst. (3 ábra)

Azt az egyenest választjuk a kapott egyenesseregből, amelyik a legkisebb hibával rendelkezik, ha több ilyen is van, akkor azt választjuk, amelyiknek legkisebb az előző egyenessel közbezárt szöge. A másik irányba is megismételve az előző eljárást egyenes szegmens sereget kapunk A d távolság értéke

az egyenesek sűrűségét határozza meg. A vonalfelület kialakításához volt szükséges az egyenes szegmensek metszésének elkerülése.

Abban az esetben, ha a kapott egyenesek rendszere nem megfelelő, azaz nem közelíti meg eléggé az adott felületet, akkor a másik irányba is megkonstruáljuk a rendszert. Az algoritmus csak szalagszerű felülettel foglalkozik. A felület lehet nyílt, irányítható zárt, illetve Möbius szalag. Az utóbbi esetet kizárhatjuk, mivel a gyakorlati haszna elhanyagolható. Irányítható zárt szalagszerű felület esetén az algoritmusnak nem szükséges biztosítani a közel záródó rendszert, ha a vonal felülettől való eltérés túl nagy. Ebben az esetben ajánlott, a mindkét irányban végrehajtott algoritmus során kapott sorozatok végső átlagolása.

A kezdeti megszorító feltétel feloldására csak megfelelően kell mozgatnunk, változtatnunk a referencia keretet az algoritmus alatt. A jó (tényleges vagy becsült) pozíció a mozgó sík számára az előző vonalszegmens középpontjának érintősíkja.

Konstruáljunk az egyenesekből approximáló B-spline felületet

Az előző lépés eredménye, az $l_0, ..., l_N$ egyenessereg közelíti a megadott ponthalmazt. A következő lépésben ezt a rendszer approximálja

$$x(u, v) = c(u) + vg(u), \quad v \in [-1, 1], \quad u \in [a, b] \tag{6}$$

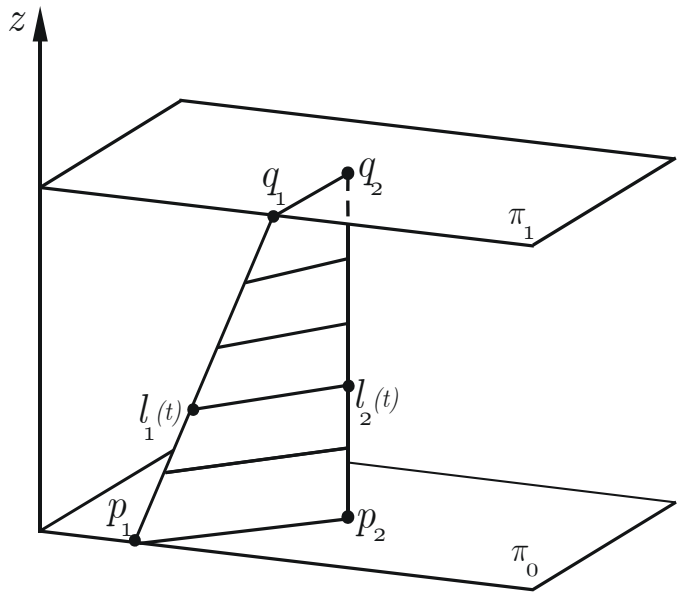
szalagszerű vonal felület, ahol $g(u)$ a generáló görbe, tehát az éppen elmetszett vagy érintett alkotó egy irányvektora, $c(u)$ a vezérgörbe vagy direktrix.

A jelenlegi approximációs problémát az irányított egyenesszegmensek (szakaszok) L halmazán értelmezzük a háromdimenziós euklideszi térben, $\mathbb{R}^3$-ban. Helyes meglátás az, ha a problémát úgy kezeljük, mint pusztán ennek a struktúrának a kiszámítási feladata.

Egy irányított l egyenes szakasz megadható a végpontjainak (p, q) rendezett párosával, amely megad egy természetes leképezést a valós hatdimenziós affin térre $\mathbb{R}^6 = \mathbb{R}^3 \times \mathbb{R}^3$,

$$\sigma : L \mapsto \mathbb{R}^6, \quad (p, q) \mapsto X = (x_1, ..., x_6) = (p_x, p_y, p_z, q_x, q_y, q_z). \tag{7}$$

A pontokat amelyek szakaszokat határoznak meg $\mathbb{R}^3$-ban leképeztük egy háromdimenziós altér $P : x_1 = x_4, x_2 = x_5, x_3 = x_6$ pontjaira. Minden szakasz irányításának megváltoztatása olyan mint egy affin tükrözés P-n.



4. ábra. Az $l_1\,(g_0, g_1)$ és az $l_2\,(h_0, h_1)$ egyenesek

Értelmezzük két szakasz távolságát

Ahhoz, hogy meghatározzuk két szakasz $l_i = (p_i, q_i)$ távolságát figyelembe vesszük azon szakaszpontok távolságát, amelyek a vizsgált tartományba esnek. Úgy választunk lokális koordinátarendszert, hogy a vizsgált tartomány a következő síkok közé esik:

$$\pi_0 : z = 0, \quad \pi_1 : z = 1.$$

A koordinátarendszer z-tengelyét úgy választjuk, hogy a vizsgált egyenesek és a z-tengely γ_0 szöge a lehető legkisebb legyen. Ekkor a két szakasz pontjai között a következő megfeleltetést tehetjük:

$$(1-\lambda)\,p_1 + \lambda q_1 \mapsto (1-\lambda)\,p_2 + \lambda q_2, \quad \lambda \in [0,1]\,.$$

A megfelelő pontokat összekötve bilineáris felületet (hiperbolikus paraboloidot vagy síkot) kapunk.

Ezekután tekinthetjük két egyenes szakasz, l_1 és l_2 távolságának négyzetét (az egyszerűsítés kedvéért megszorozzuk 3-mal)

$$d\left(l_1, l_2\right)^2 \ := 3 \int_0^1 \left[\left(1-\lambda\right)\left(p_1-p_2\right)+\lambda\left(q_1-q_2\right)\right]^2 d\lambda$$
$$= \left[\left(p_1-p_2\right)^2+\left(q_1-q_2\right)^2+\left(p_1-p_2\right)\left(q_1-q_2\right)\right]. \qquad (8)$$

Az ily módon definiált távolság egyszerűen két pont távolsága $L_i := \sigma(l_i) \in \mathbb{R}^6$ euklideszi metrikában. Ha a vizsgált tartomány fölött, nem teszünk különbséget egy X egyenes és $X = (x_1, x_2, x_3, x_4, x_5, x_6)$ koordinátái között, akkor a (9) egy pozitív kvadratikus forma, a következő kordináta reprezentációval

$$\langle X, X\rangle = x_1^2 + \ldots + x_6^2 + x_1x_4 + x_2x_5 + x_3x_6 \qquad (9)$$

1. Megjegyzés. *A (8)-ben definiált távolság csak $faktor \leq \cos\gamma_0$ értékben különbözik az euklideszi merőleges távolságtól (pl. egyenes és pont távolsága). Tehát ha γ_0 relative kis érték, akkor ellenőrzésünk alatt tudjuk tartani a két távolságértelmezés közötti különbséget.*

Tehát egyenesek interpolációját vagy approximációját $\mathbb{R}^3$-ban megfeleltethetjük pontok interpolációjának vagy approximációjának $\mathbb{R}^6$-ban.

1. Lemma. *A (7)-ben megadott σ leképezés meghatároz egy bijektív leképezést a háromdimenziós euklideszi tér irányított szakaszai és a valós hatdimenziós euklideszi tér pontjai között. Mivel oly módon definiálható két szakasz közötti távolság (8), hogy az két pont távolságának felel meg $\mathbb{R}^6$-ban, s amely pozitív kvadratikus formával adható meg (9).*

Tehát a felület approximációs problémát visszavezettük görbe approximációs problémára. Megjegyezzük, hogy a hatdimenziós térben lévő B-spline görbe fokszáma k, a háromdimenziós térben lévő tenzori szorzattal előállított B-spline felület fokszámai pedig $(1, k)$. Tehát alkalmazhatóak a standard görbe approximációs technikák, de a (8) egyenlet szerinti tiszta euklideszi távolságot be kellett vezetnünk.

Az approximáció javítása

Az eredmények javításához, a legkisebb négyzetek módszerét gyakran kombinálják függvény regularizáció minimalizálásával (ún. simasági feltétel, lásd 3.pontbeli fejezetet) Az (6)-ben megadott vonalfelület paraméter független, azonban jó választás Chen és Pottman [16] szerint az ún. plate-spline regularizált függvény

$$F(x) = \int_a^b \int_{-1}^1 (x_{uu}^2 + 2x_{uv}^2 + x_{vv}^2) dv du \tag{10}$$

használata. Alkalmazva (6) összefüggést, az előző dekompozicóját kapjuk

$$F(x) = 2 \int_a^b c_{uu}^2 du + \frac{2}{3} \int_a^b (g_{uu}^2 + 6g_u^2) du, \tag{11}$$

azaz $c(u)$ és $g(u)$ felhasználásával két kezelhető függvényt kaptunk. A (10) és a (11) egyenletek további használatáról részletesen olvashatunk a [16]-ban és az [46]-ben. (lásd még Weiss, Andor, Renner és Várady 2002-es cikkében [88])

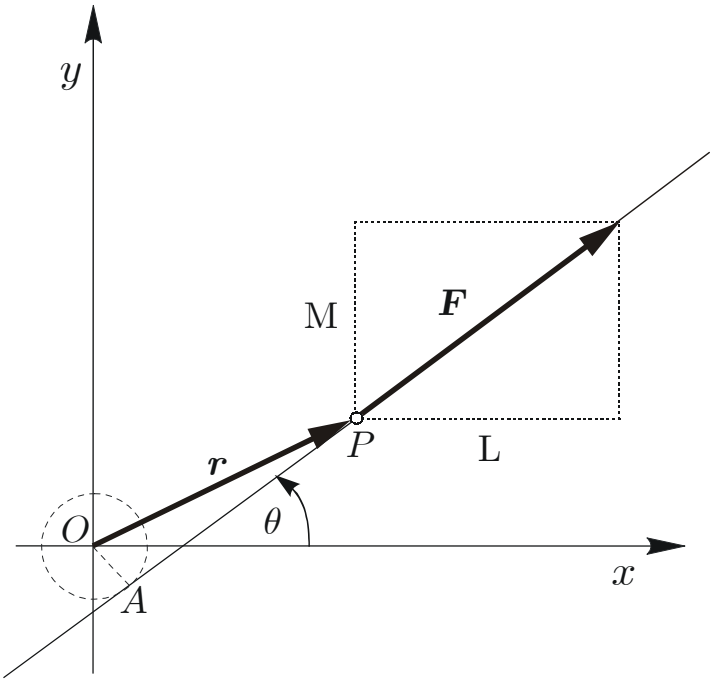
A paraméterkorrekció technikájáról szintén az [46]-ben találunk leírást.

Összegezve

Az algoritmus, három lépésben megfelelő vonalfelület ad számunkra, de nagy matematikai apparátust kellett felhasználnunk, és az első két lépésében ismertetett nagyon jó kezdeti illesztést kíván.

A mérnöki visszafejtésben foglalkozni kell a kiugróan eltérő adatokkal is, amelyek vagy a mérési hibákból, vagy olyan pontok adataiból keletkeznek, amelyek a modell geometriailag eltérő részéről származnak. A legkisebb négyzetek módszere nagyon érzékeny az ilyen eltérésekre. Megoldásként használhatjuk Rousseeuw és Leroy által 1987-ben publikált nagy teljesítményű regressziós módszert [78], ahol a legkisebb négyzetek rendszerében iteratív súlyozással kiszűrhetjük a nagy eltéréseket okozó adatokat, majd ezekhez megfelelő súlyokat rendelünk. Minden tárgyalt módszer élvezheti ezen módosítás, előszűrés előnyeit.

A fent tárgyalt módszer – egyenes szakaszok approximációja – természetesen vezet minket $\mathbb{R}^3$-ban lévő egyenesek $\underline{L}$ terének az approximációjához.



5. ábra. Forgatónyomaték az O pontra vonatkozólag

A dolgozat későbbi részében felhasználjuk, sőt a magyar terminológiának megfelelően újra tárgyaljuk a Plücker-koordinátákat a tárgyaljuk. A térbeli egyenesek kezeléséhez nagyon hasznos segédeszköz a Plücker-koordináták felhasználása.

Plücker-koordináták

A jobb megértés kedvéért segítségül hívjuk a fizikában használt **F** erő tengelyre számított forgatónyomatékát (lásd Budó [14]).

Tekintsünk egy jobbsodrású koordináta rendszert, ahol a z tengely a lap síkjából kifelé mutat az olvasó felé. Az egyenes **F** irányvektorának tengelymenti összetevői L és M.

$$\begin{aligned} L &= x_2 - x_1 = \mathbf{F}\cos\theta \\ M &= y_2 - y_1 = \mathbf{F}\sin\theta \end{aligned} \tag{12}$$

A z tengelyre vonatkozó (a z tengelyre merőleges síkban ható) **F** erő

forgatónyomatéka $R = \mathbf{F} \cdot \overline{OA}$ előjeles mennyiség.

Ha az $\mathbf{F}$ erő P támadáspontjának $\overrightarrow{OP} = \mathbf{r}\,(x_1, y_1)$ helyvektora $\mathbf{F}$ irányával ϑ szöget zár be, ekkor, $\overline{OA} = r * \sin\vartheta$ miatt

$$R = \mathbf{F} \cdot \overline{OA} = |\mathbf{r}|\,|\mathbf{F}| \sin\vartheta = |\mathbf{r} \times \mathbf{F}| = x_1 M - y_1 L = \begin{vmatrix} x_1 & y_1 \\ L & M \end{vmatrix} \tag{13}$$

összefüggés adódik pozitív irányú forgatás esetén. A levezetés megadja a forgatónyomaték következő meghatározását is. Egy P pontban támadó $\mathbf{F}$ erő O pontra számított forgatónyomatéka az $\overrightarrow{R} = \mathbf{r} \times \mathbf{F}$ vektormennyiség, ahol $\mathbf{r}$ az O pontból a P pontba mutató helyvektor.

$$\overrightarrow{R} = \mathbf{r} \times \mathbf{F} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_1 & y_1 & 0 \\ L & M & 0 \end{vmatrix} = (x_1 M - y_1 L)\,\mathbf{k}.$$

Ha behelyettesítjük (12)-t (13)-ba, akkor

$$R = \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix}.$$

A következő mátrixból –balról jobbra aldeterminánsokat képezve– nyerhető három determináns értéke egyértelműen meghatározza az l egyenest.

$$\{L, M, R\} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \end{bmatrix} \tag{14}$$

Meghatározható az egyenes x tengellyel bezárt θ szöge, és az $\overline{OA}$ szakasz hossza,

$$\theta = \arctan\frac{M}{L}, \quad \overline{OA} = \frac{R}{\sqrt{(L^2 + M^2)}}.$$

Az (14) alapján könnyen áttérhetünk a homogén koordinátás alakra. A jelölésben, a magyar terminológiában nem szokásos módon, a w_i homogén koordinátákat az alak elejére írjuk.

$$\begin{bmatrix} w_1 & x_1 & y_1 \\ w_2 & x_2 & y_2 \end{bmatrix}$$

Jobbról kezdve a következő összefüggéseket kapjuk

$$L = \begin{vmatrix} w_1 & x_1 \\ w_2 & x_2 \end{vmatrix} = w_1 x_2 - w_2 x_1$$
$$M = \begin{vmatrix} w_1 & y_1 \\ w_2 & y_2 \end{vmatrix} = w_1 y_2 - w_2 y_1$$
$$R = \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} = x_1 y_2 - x_2 y_1$$

Ha most általánosítva háromdimenzióban adjuk meg az egyenes iránytényezőit, akkor

$$L = x_2 - x_1$$
$$M = y_2 - y_1$$
$$N = z_2 - z_1$$

ahol az egyenes harmadik iránytényezője legyen $N = z_2 - z_1$. Az x, y és a z tengelyre vonatkozó nyomatékok rendre legyenek P, Q és R, és kiszámításuk a következő:

$$P = y_1 z_2 - y_2 z_1 = y_1 N - z_1 M = y_2 N - z_2 M$$
$$Q = z_1 x_2 - z_2 x_1 = z_1 L - x_1 N = z_2 L - x_2 N$$
$$R = x_1 y_2 - x_2 y_1 = x_1 M - y_1 L = x_2 M - y_2 L$$

Egy térbeli egyenes reprezentálható a következő homogénkoordinátás alakkal

$$\begin{bmatrix} w_1 & x_1 & y_1 & z_1 \\ w_2 & x_2 & y_2 & z_2 \end{bmatrix}$$

Az egyenes *Plücker-koordinátái* pedig

$$\left.\begin{aligned} l_{41} &= w_1 x_2 - w_2 x_1 \\ l_{42} &= w_1 y_2 - w_2 y_1 \\ l_{43} &= w_1 z_2 - w_2 z_1 \end{aligned}\right\} = \omega$$

$$\begin{aligned} l_{23} &= y_1 z_2 - y_2 z_1 \\ l_{31} &= x_2 z_1 - x_1 z_2 \\ l_{12} &= x_1 y_2 - x_2 y_1 \end{aligned} = V$$

Ezekután megadhatjuk az egyenes hat koordinátáját az

$$(L, M, N, P, Q, R)$$

alakban, vagy hangsúlyozva a homogenitás az

$$(l_{41}, l_{42}, l_{43}, l_{23}, l_{31}, l_{12})$$

alakban. Legyen a két ponton áthaladó egyenes ω iránytényezővel és a V forgatónyomatékkal megadva. Mivel a forgatónyomaték-vektor merőleges az egyenesre, ezért

$$\omega \cdot V = 0. \tag{15}$$

A Plücker-koordinátákat felhasználva adódik

$$l_{41}l_{23} + l_{42}l_{31} + l_{43}l_{12} = 0 \tag{16}$$

összefüggés, amelyet *Plücker-relációnak* nevezünk. A Plücker-reláció kifelyezi, az egyenes irányvektorának, és a forgatónyomaték-vektorának a merőlegességét.

Normalizált Plücker-koordináták

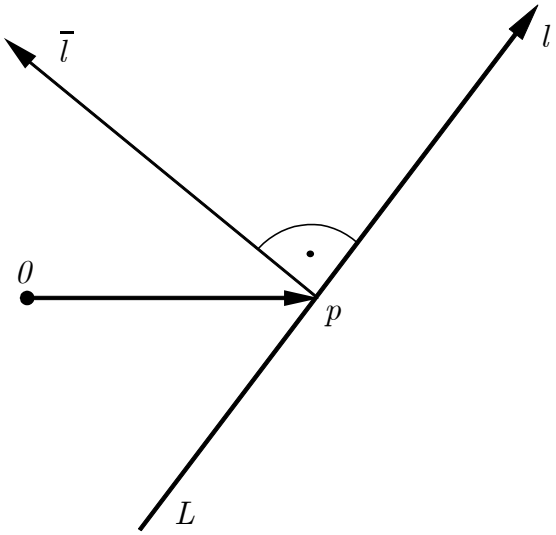
A forgatónyomaték-vektor vagy másnéven momentum vektor fizikában használt levezetésétől eltérően egyszerűen kiindulhatunk a következőkből is (lásd [69]). Egy L egyenes megadható $\mathbf{p} \in L$ pontjával és normalizált $\mathbf{l}$ irányvektorával, azaz $\|\mathbf{l}\| = 1$. A momentum vektor fizikában szokásos értelmezését az előző részben már rendbe raktuk, tehát

$$\bar{\mathbf{l}} = \mathbf{p} \times \mathbf{l},$$

Az $\bar{\mathbf{l}}$ független $\mathbf{p}$ választásától, hiszen helyettesíthető az L egyenes bármely $\mathbf{q} = \mathbf{p} + \lambda\mathbf{l}$ pontjával. Az $\mathbf{l} = (l_1, l_2, l_3)$ és $\bar{\mathbf{l}} = (l_4, l_5, l_6)$ koordinátákat nevezik *normalizált Plücker-koordinátáknak,* amelyek megfelelnek egyrészt a normalizáció feltételének, másrészt $\mathbf{l}$ és $\bar{\mathbf{l}}$ merőlegessége miatt, kielégítik a

$$\mathbf{l}\cdot\bar{\mathbf{l}} = 0$$

Plücker-relációt, másnéven *Plücker-egyenletet* is.



6. ábra. Az egyenes irány és a momentum vektora

Bármely $\left(\mathbf{l}, \bar{\mathbf{l}}\right)$ számhatos egy egyenest reprezentál $\mathbb{E}^3$-ban, ha kielégíti a $\mathbf{l}\cdot\bar{\mathbf{l}} = 0$ Plücker-egyenletet és $\|\mathbf{l}\| = 1$. Nem vesszük figyelembe az egyenes irányítását, $\left(\mathbf{l}, \bar{\mathbf{l}}\right)$ és $\left(-\mathbf{l}, -\bar{\mathbf{l}}\right)$ ugyanazt az L egyenest reprezentálja.

Klein-féle leképezés

Vezessük be a következő jelölést

$$(l_1, \ldots, l_6) := (l_{41}, l_{42}, l_{43}, l_{23}, l_{31}, l_{12})\,.$$

Ekkor a homogén Plücker-koordináták alkalmazásával értelmezhetünk egy kölcsönösen egyértelmű megfeleltetést a valós számok rendezett homogén hatosa $(l_1, \ldots, l_6) \neq (0, \ldots, 0)$ és a $\mathbb{P}^3$ tér egyenesei között. Az $(l_1, \ldots, l_6)$ számhatos felfogható az ötdimenziós $\mathbb{P}^5$ projektív tér pontjainak koordinátájaként. A kapott bijektív leképezés eredményez egy újabb kölcsönösen egyértelmű leképezést a $\mathbb{P}^3$ tér egyenesei és az ötdimenziós $\mathbb{P}^5$ projektív tér azon pontjai között amelyek kielégítik a Plücker-egyenletet. A (16) egyenletet átalakítva kapjuk

$$l_1 l_4 + l_2 l_5 + l_3 l_6 = 0 \tag{17}$$

Az ily módon definiált γ *Klein-leképezés* egy bijekciót hoz létre a $\mathbb{P}^3$ tér egyenesei és a $\Omega \subset \mathbb{P}^5$ Klein kvadrik pontjai között. Természetesen a $\mathbb{P}^5$ projektív tér nem minden pontja reprezentál egyenest.

A (16) egy másodfokú homogénkoordinátás egyenlet, amelynek a megoldása olyan pontok halmaza, amelyek egy hiperfelületre, egy négydimenziós kvadrikra illeszkednek. Ezt a felületet nevezzük *Klein-kvadriknak* [53]. Azon pontok $\mathbb{P}^5-$ben, amelyek nem illeszkednek a *Klein-kvadrikra,* nem reprezentálnak egyenest $\mathbb{P}^3$-ban. Másrészről $\mathbb{R}^3$-ban nincs olyan egyenes, amelyre $\omega = 0$ teljesülne ($V = 0$ teljesülhet az origón átmenő egyenesek esetében), viszont $\mathbb{P}^3$ -ban ezek éppen a végtelen távoli egyenesek, amelyeknek megfelelnek a *Klein-kvadrik* $\omega = 0$-val definiált kétdimenziós síkjának.

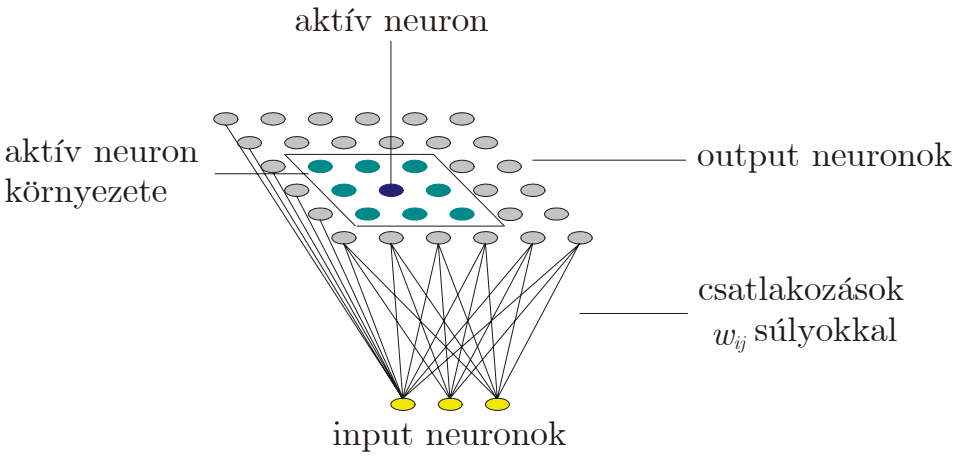
4. Neurális hálók és alkalmazásai a CAGD-ban

4.1. Néhány szó a neurális hálókról általában

A mesterséges intelligencia neurális hálói biológiailag inspirált modellek, az emberi agy kutatásában tárgyalt, vizsgált neuronok elméletén, működésük struktúráján alapszik. A neurális hálózat számos egyszerű feldolgozóelemből (neuronból vagy csomópontból) áll, amelyek nagyon összetett és szoros kapcsolatban vannak egymással. A működés során az egyes elemek közötti kapcsolatot súlyok jellemzik. A súlyok hozzá vannak rendelve minden kapcsolathoz. Általában a csomópontok (nodes) két réteget alkotva szerveződnek. A tanulási folyamat során biztosítjuk az input vektorokat az inputréteg számára, úgy, hogy a kívánt outputot vagy előírjuk, vagy nem. Ez a különbség klasszifikálja a neurális hálókat a felügyelettel vagy a felügyelet nélküli változatok között. A neurális hálókat csoportosíthatjuk az input értékek alapján is (bináris vagy folytonos). A tanulási folyamat önmagában három fő lépést tartalmaz, az input minta biztosítása, az output előállítása számítások alapján, és a súlyok módosítása speciális szabályok és függvények segítségével. Ezek a lépések ismétlődnek többször, amíg azt nem mondhatjuk a hálózatra, hogy tanult vagy kiképzett. A mesterséges neurális hálózatok részletes leírását megtaláljuk Freeman és Skapura [27] és Rojas [77] könyvében. Magyar irodalomként meg kell említenem Borgulya István [12] összefoglaló könyvét.

A Kohonen-háló egy kétrétegű, felügyelet nélküli és folytonosan kiértékelt neurális hálózat, amelyet T. Kohonen publikált [54]-ban. A Kohonen-háló kiváló eszköz a sorrend meghatározására bármilyen típusú ponthalmaz esetén. A hálózat nagyon erős öntanuló, önszervező képességgel rendelkezik. A modell kiviteli rétege, vagy Kohonen-rétege általában egy- vagy kétdimenziós (lásd 7. ábra). Ez azt jelenti, hogy a réteg feldolgozó elemei egydimenzióban láncot, kétdimezióban rácsot alkotnak. Az n elemű beviteli réteg minden eleme a Kohonen-réteg minden elemével össze van kötve és a Kohonen -réteg minden eleme egy $\mathbf{w}_j(w_{1j}, w_{2j}, w_{3j}), (j = 1, ..., n)$, súlyvektorral jellemezhető.

A modell olyan szomszédsági környezetet definiál, amely a közvetlen szomszédos elemek kapcsolatát erősíti, a távolabbi elemekét pedig csökkenti (on-center/off surround kapcsolat). Így létrejöhet a topológia tartó leképe-



7. ábra. Kohonen háló topológiája

zés, azaz, a modell a tanulás során, ha egy input adatsorhoz a Kohonen-réteg $\mathbf{q}_j$ elemét hozzárendeli, akkor további hasonló input adatsorhoz (mintához) a $\mathbf{q}_j$ közeli szomszédos elemeket rendeli. A háló versengő tanulást folytat, azaz, győztes feldolgozó elemet keres. A legaktívabb elem lesz a győztes, és ezen elem súlyvektorát kell legerősebben módosítani, míg a szomszédos elemekét kisebb mértékben. A szomszédságon kívüli, azaz, a szomszédos környezetet meghatározó sugáron kívül eső elemek, neuronok súlyvektorai változatlanok maradnak. A környezetet meghatározó sugár (radius) függvényről és a nyerő vagy aktivitási (gain term) függvényről 5.1.2. pontban olvashatunk részletesen. A Kohonen-háló topológiatartó tanulását egy rácsozattal is szemléltethetjük. Tekintsük a kimeneti réteg elemeit térbeli pontoknak, a pontok koordinátáját, pedig a súlyvektorok határozzák meg. Ha a térbeli pontokat összekötjük, akkor rácsozatot kapunk, amely a tanulási folyamat során végig megtartja a topológiáját, és az input adatsoroktól függően változtatja az alakját.

4.2. Eddigi alkalmazások a CAGD-ban

A következő négy alfejezetben a mesterséges intelligencia néhány alkalmazását ismerhetjük meg. Az ismertetett cikkek mind friss eredmények, amely jól mutatja a témában rejlő lehetőségeket. További eredményként meg lehet

említeni, hogy a neurális hálók közül a backpropagation algoritmus is alkalmazható a CAGD-ban. Erre jó példa az általam publikált [55] cikk, ahol lineáris leképezéseket tud megtanulni vagy előállítani a neurális hálózat. Az előállítás olyan esetekben hasznos, amikor egy ponthalmaz és képe között (pl. torzult fénykép pontjai és a valóság pontjai) az egzakt leképezés már nem lenne lineáris, de egy igen jól közelítő lineáris leképezést találhatunk a neurális háló segítségével.

Genetikus algoritmus használata görbe illesztési problémák esetében

Elsőként meg kell említenünk Márkus, Renner és Váncza cikkét. A szerzők a genetikus algoritmusok (genetic algortihms, GA) áttekintését, és szabad formájú görbék tervezésében való alkalmazhatóságát írják le. Nyilvánvaló, hogy vannak a CAGD-ban olyan esetek, amikor a determinisztikus algoritmusok nem vagy nehezen használhatóak, vagy hibás eredményre vezetnek, ilyenkor javasolt valamilyen nem determinisztikus, valószínűségen alapuló módszer alkalmazása.

A CAGD-ban sok esetben szükséges megadni görbéket, vagy felületeket, oly módon, hogy szigorú geometriai feltételeknek megfeleljenek (görbék interpoláljanak előre megadott pontokat, felületek áthaladjanak előre megadott görbéken). Gyakran a megoldás nem kívánatos kilengéseket, éles kanyarokat és hirtelen változásokat tartalmaz geometriai jellemzőikben. A problémák kiszűrése a szabad paraméterek felhasználásával is sokszor igen nehéz feladat. Sokszor olyan paramétert kell megadni, amelynek nincsen közvetlen geometriai jelentése, de szükséges a matematikai reprezentációhoz. A problémák megfogalmazása gyakran mérnöki szemléletű, nem pedig matematikai. A szerzők javasolják a GA alkalmazását, amely nem ad ugyan egzakt megoldást, de jól kezelhető megoldáshalmazt biztosít.

A GA ötlete a természetből jön: ahelyett hogy egy egyedi probléma egyetlen optimális megoldását keressük, a lehetséges megoldások populációja párhuzamos módon fejlődik ki. A populáció egyede rendelkezik genetikus kóddal (génnel vagy kromoszómával), amely jellemzi örökítő tulajdonságát. Az új egyed egy vagy két szülőtől származhat, nemi jelleg nélküli (asexual) vagy nemi jellegű (sexual) szaporodással. Az első esetben véletlenszerűen változik a genetikus kód, a második esetben a két szülő génjeinek keresztezéséből alakul ki az új kód. Feltéve, hogy az új egyedek lecserélik a létezőket,

a leszármazottak öröklik az ősök jellemzőit, a valamely célnak megfelelőbb egyedeknek nagyobb esélyük van örökíteni, akkor a populáció egyre és egyre jobb, megfelelőbb egyedeket eredményez. A genetikus folyamat három jellemzője van, a populáció méretet, a kereszteződések mértéke, és a mutációk száma. Ezen elven működő genetikus algoritmus részletes leírását megtaláljuk, az említett cikkben (lásd [61]). A szerzők ezen algoritmust ültették át görbeillesztési problémára, és nagyon hasznosnak találták olyan problémák megoldására, amikor a hagyományos algoritmusok nem működnek.

Funkcionális hálózatok használata B-spline felületekről származó adatpontokra illesztett Bézier felület előállítására

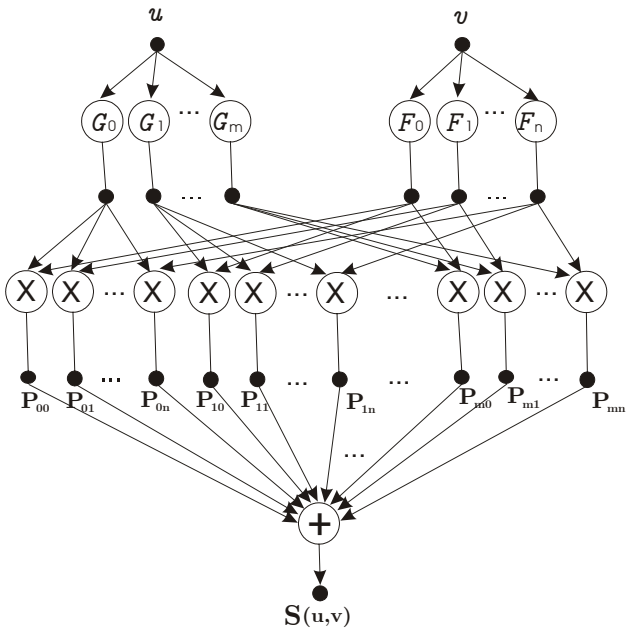
A neurális hálózatok erőteljes kiterjesztése a funkcionális hálózatok, amelyet Castillo ír le [15] cikkében. Ezen típusú hálózatok nagyobb sokoldalúságot mutatnak mint a hagyományos neurális hálózatok, ebből következőleg sikeresen lehet használni őket a CAGD-ban. Iglesias és Gálvez [51] cikkében példát adnak arra, hogyan lehet funkcionális hálózatok segítségével előállítani tenzori szorzattal megadott felületeket, amelyek nagyon fontosak a CAGD-ban. Az új algoritmus hatékonyságát egy Bézier felület approximációs problémán keresztül mutatják be, ahol a approximálandó pontokat, a kontrollpontokat, B-spline felületről nyerik. Természetesen a B-spline felületről nyert pontok határozzák meg a Bézier felület fokszámát is. A szerzők több B-spline felületet is felhasználnak az algoritmus hatékonyságának az igazolásaira. A hagyományos neurális hálózatokkal szemben két fő különbséget említ a cikk:

1. A neurális hálózat neurális függvényei azonosak, a funkcionális hálózatban lehetnek különbözőek.

2. A neuron outputok a neurális hálózatban különbözőek, a funkcionális hálózatban lehetnek azonos output neuronok is.

Tekintsük a szerzők példáját. Keressünk olyan $\mathbf{S}(u,v)$ felületet, amely kielégíti a következő összefüggést

$$\mathbf{S}(u,v) = \sum_{j=0}^{n} \alpha_j(u) F_j(v) = \sum_{i=0}^{m} \beta_i(v) G_i(u),$$



8. ábra. Parametrikus felület reprezentálása funkcionális hálózattal

ahol az $\alpha_j(u)$, $(j = 0, 1, \ldots, n)$ és a $\beta_j(v)$, $(i = 0, 1, \ldots, m)$ eggyüthatók, az $F_j(v)$ és a $G_i(u)$ lineárisan független függvények halmaza. A szerzők a következő megoldást adják

$$\mathbf{S}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{P}_{ij} G_i(u) F_j(v),$$

ahol $\mathbf{P}_{ij}$ térbeli pontokat jelöl, azaz, a $\mathbf{S}(u, v)$ egy tenzori szorzattal megadott felület. A 8. ábra mutatja a példa alapján előállított funkcionális hálózatot.

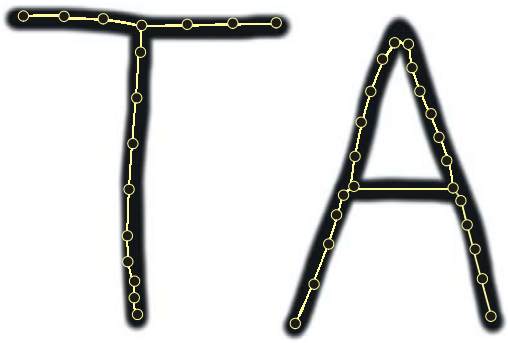
Vázkijelölés TASONN segítségével

A vázkijelölés (skletonization) olyan folyamat, amely során több pixel széles objektumot transzformálunk egy pixel széles objektumba, úgy, hogy a topológiai tulajdonságokat többé-kevésbé megőrizzük. A keletkezett objektum a váz vagy más néven a szkeleton (skeleton). Ezek a szkeletonok nagyon hasznosak például karakter vagy kromoszóma felismeréskor, sőt az eredeti objektum adattömörítésének is tekinthetjük. (lásd 9. ábrát) A szkeleton fogalmát Blum [6] vezette be, amelyet folytonos esetben középső tengely transzformációnak is nevezik (medial axis transformation, MAT). Formálisan azon körök középpontja határozza meg a középső tengelyt, amelyek teljes egészében az objektumon belül helyezkednek el, és érintik az objektum határát kettő vagy több pontban. Nagyon sok módszer ismert a probléma megoldására.

A vektor szkeltonizáció esetében azzal foglalkozunk, hogy keresünk egy gráfot az input minta szakasz rész approximációjával. Ebben a dolgozatban irreleváns a raszteres szkeletonozáció témaköre. 2001-ben a neurális hálók segítségével Datta és szerzőtársai (lásd [22]) adnak új módszert A cikkük jó példa arra, hogyan lehet Kohonen-hálót használni approximációs problémákra. Az általuk használt háló nemcsak egy önszervező (self-organizing neural network, SONN) neurális háló, hanem ennél több, azaz, topológialilag alkalmazkodó hálózat, angolul topology-adaptive self-organizing neural network, TASONN. Szemben a hagyományos Kohonen-hálóval a TASSON háló topológiája automatikusan alkalmazkodik a megfelelő alak felé, azaz, dinamikusan változik, miközben megtartja öntanuló, önszervező képességét is. A cikkben részletezett algoritmus a minta alapján előállítja a vektor vázat, amelyből a rasztervász is nyerhető. Meg kell említenünk, hogy a Kohonen-háló említett hiányosságával jómagam és szerzőtársaim is szembekerültünk. A [86] cikkben publikált megoldást a 5.2. pontban ismertetem, amely a Kohonen-háló dinamikus kiterjesztését tartalmazza.

Térbeli ponthalmazok paraméterezése és rekonstrukciója neurális háló és parciális differenciálegyenletek (PDE) segítségével

Barhak és Fischer (lásd [4]) cikkükben szintén a mérnöki visszafejtésben (reverse engineering) gyakori problémára kínálnak megoldást neurális háló és PDE (Partial Differential Equation) segítségével. Lézer szkennerrel gyorsan lehet 3D adatokhoz jutni, de a kapott digitális adatok olyan nagy



9. ábra. Szkeletonizáció eredménye

mennyiségben és teljesen szabálytalanul állnak rendelkezésre, hogy azokat mindenképpen intenzív rekonstrukciós eljárásoknak kell alávetni. Szabadformájú felületek rekonstrukciójának két fontos lépése van, a parametrizáció és a felületillesztés.

A jelenlegi parametrizálási módszerek topológiai problémákkal küzdenek, pl. zajos önmetsző felületek esetében. A szerzők erre a problémára adnak megoldást, felhasználják Ma és Kruth [60] eredményét, azaz térbeli bázis felületet (base surface) használnak a parametrizálása sík helyett. Valójában a bázis felület egy durva approximálása a végső felületnek, amely majd illeszkedik a digitalizált pontokra. Ezután a kezdeti parametrizálás után újabb és újabb felületeket állítunk elő a paraméter értékek és a input pontok alapján, s az így kapott felületsorozat fog konvergálni a kívánt illeszkedő felülethez.

A szerzők két új parametrizálási módszert adnak meg. A PDE módszer önmetszés nélküli paraméterhálót biztosít. A neurális hálót (SOM, Self Organizing Maps, amely valójában Kohonen-háló lásd [54]) a síkbeli paramétertartományon alkalmazzák. Előnye, hogy nemcsak a határpontok, hanem az összes pont részt vesz a kontrollháló kialakításában, ezáltal uniform sűrűségű paraméter-tartomány keletkezik. A szerzők cikkükben össze is hasonlítják a két módszert, és számos gyakorlati példán keresztül mutatják be eredményeiket.

5. Ponthalmaz geometriai modellezése Kohonen-háló segítségével

5.1. Szabad formájú felületek illesztése pontfelhőre neurális hálók segítségével

Pontfelhőre illesztett felület approximálásának vagy interpolálásának a problémájára számtalan különböző módszer és applikáció készült a CAD/CAM alkalmazásokban (lásd Boehm [10], Foley és Hagen [26], Hoschek és Lasser [46]). Hoffmann és Várady [43] cikkükben a Kohonen-hálót alkalmazzák az előbbi probléma megoldására. A későbbi eredmények erre a módszerre épülnek, ezért ezt bővebben ismertetem.

A 4.1. pontban leírt Kohonen-hálót használhatjuk felületillesztési problémák megoldására is. Vagy addig futtatjuk az algoritmust, amíg egy négyszög kontrollháló előállításával megfelelő B-spline vagy egyéb approximáló vagy interpoláló felületet kapunk, vagy a 3.2. pontban meghatározott base surface-t állítjuk elő, amely egy lehetséges mód a paraméter inicializálásra. A Kohonen-háló előnyös tulajdonságait felhasználva (kétrétegű, felügyelet nélküli és folytonosan kiértékelt neurális hálózat) kiváló eszközt nyerünk bármilyen típusú ponthalmaz esetén a sorrend meghatározására. A hálózat nagyon erős öntanuló, önszervező képességgel rendelkezik. A neurális hálók előnyeiről teoritikus eredményeket olvashatunk Alder cikkében (lásd [1]), továbbá szabad formájú felületek előállítása neurális hálókkal szintén érdekes kutatási terület (még Gu és Yan [31]).

A mi esetünkben a Kohonen-háló erős öntanuló képességét használjuk fel. Ezt a képességet gyakorlatilag arra használjuk, hogy egy előre megadott struktúrát vonszoljunk, mozgassunk, úgy hogy az majd keresztül menjen a megadott pontokon, azaz illeszkedjen a ponthalmazra. Görbék esetében ez az előre megadott struktúra poligon-, felületek esetében pedig négyszöghálózat. Az úgynevezett tanulási folyamat után az előre definiált hálózat követni fogja az input pontok struktúráját és eloszlását, vagy ha relatívan kicsi az input pontok száma, akkor illeszkedi fog minden pontra. Ennek a technikának fő előnye az, hogy egyaránt kezelni tudunk olyan problémákat, amelyek kevés ponttal, valamint amelyek nagy számú ponttal, pontfelhővel adottak. Az alkalmazás másik előnye, hogy a Kohonen-háló végig megtartja topológiáját (jelen esetben a négyszöghálót) a tanulási folyamat során. Mi-

után a Kohonen-háló megtanulta a pontfelhőt előállít egy négyszöghálót, amely kontrollháló alapja lehet, valamely standard approximációs vagy interpolációs módszernek (pl. NURBS felületek).

Mivel a Kohonen-hálót már ismertettük a 4.1. pontban, ezért itt már csak a tanulási algoritmust adjuk meg. A standard – azaz nem dinamikus – módszer teljes részletes leírását megtaláljuk a [43]-ben.

5.1.1. A neurális háló tanulási folyamata

A megadott ponthalmaz pontjait jelöljük a következőképpen :

$$\mathbf{p}_i(x_{1,i}, x_{2,i}, x_{3,i}), \quad (i = 1, ..., m),$$

ahol m az input pontok számát jelöli. A Kohonen-háló input rétege három neuront tartalmaz, amelyek bemenetként szolgálnak az input koordináták számára. Az output réteg $n(> m)$ neuront tartalmaz. Az n értéke függ az m értékétől, általában $n = 4 \times m$. Ha az input pontok száma nagy, vagy az input eloszlással definiált, akkor kényelmesebb n értékét az input pontok számától függetlenül megválasztani. Az m értéke mellett ne feledjük, hogy négyszög-háló topológiát kell hogy alkosson az output réteg. Az output réteg minden neuronja (pontja) összeköttetésben áll az input réteg összes neuronjával (pontjával) (lásd 7. ábrát), azaz, az input réteg i-edik neuronja hozzá van kapcsolva az output réteg j-edik neuronjához, és w_{ij} súly van hozzárendelve minden egyes összeköttetéshez. Ezeket a súlyokat tekintjük az output pontok koordinátáinak,

$$\mathbf{q}_j(w_{1j}, w_{2j}, w_{3j}), \quad (j = 1, ..., n),$$

amelyek majd az eredmény poligont szolgáltatják a tanulási folyamat után:

1. Inicializáljuk a $w_{sj}, (s = 1, 2, 3, j = 1, ..., n)$ súlyokat, mint kicsi véletlen számokat az input koordináták átlagainak környezetében. Inicializáljuk a t tanulási időt is, $t = 1$.

2. Adjunk új input $(x_{1,i_0}, x_{2,i_0}, x_{3,i_0})$ értékeket, amelyek egy random módon kiválasztott $\mathbf{p}_{i_0}$pont koordinátái.

3. Határozzuk meg minden output csomóérték $d_j(\mathbf{p}_{i_0}, \mathbf{q}_j), (j = 1...n)$ euklidészi távolságát az input ponttól

$$d_j = \sum_{s=1}^{3} (x_{si_0} - w_{sj}) .$$

4. Találjuk meg a $\mathbf{q}_{j_0}$ nyerő értéket, amelynek minimális a távolsága az input ponttól, azaz. $d_{j_0} = min(d_j)$.

5. Határozzuk meg a $N(t) = (j_0, j_1, ..., j_k)$ környezetet.

6. Az $N(t)$ környezetben frissítsük a csomópontok súlyait (azaz a koordinátákat) a következő képlet alapján:

$$w_{sj}(t+1) = w_{sj}(t) + \eta(t)(x_{s,i_0} - w_{sj}(t)), \quad \forall j \in N(t),$$

   ahol $\eta(t)$ az úgynevezett nyerő kifejezés (gain term), amely időben csökkenő Gauss-függvény, például:

$$\eta(t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2}, \quad t \geq 0.$$

7. Növeljük a tréning (tanulási) időt, $t = t + 1$. Ismételjük a 2.-től 7.-ig a lépéseket amíg a hálózatot tanultnak (kiképzettnek) tekintjük.

Viszonylag kevés pont esetén, akkor tekintjük befejezettnek az algoritmust, ha minden input pont rajta van a kontrollhálón, azaz minden $\mathbf{p}_i$ $(i = 1, ..., m)$ ponthoz rendeltünk egy olyan $\mathbf{q}_j$ output vektort, amely olyan, hogy egy bizonyos idő elteltével $\mathbf{q}_j$ távolsága $\mathbf{p}_i$ -től egy előre meghatározott értéknél kisebb. Erősebb, pontosabb konvergencia nyerhető, ha megköveteljük, hogy azon output vektor, amely nem konvergál az input vektorhoz illeszkedjen a két szomszédos output pont által meghatározott egyenesre. Az ilyen konvergencia különösen nagy szerepet kap a későbbiekben nyert felület simaságában. Amennyiben az input pontok számossága nagy vagy végtelen (eloszlással definiált), akkor nevezhetjük a hálózatot tanultnak, ha az output háló változása egy előre definiált határértéknél kisebb.

5.1.2. A sugár és a nyerő feltétel megadása

A sugár (radius), mint a környezet meghatározásának fontos tényezője és a nyerő feltétel vagy kifejezés (gain term) a számítási időnek igen fontos tényezői. Mindkét érték csökken az idő függvényében. A konvergencia elérése miatt a sugár értékét nulláig kell csökkenteni, mert egyébként az aktuális input olyan outputokat is magához vonzana, amelyek más inputokhoz kell hogy konvergáljanak. Ugyanakkor a gain term-nek is csökkennie kell, különben az aktuális input olyan outputokat is magához húzna, amely közelebb van más inputokhoz, de viszonylag messze van az aktuális inputtól. Meg kell jegyeznünk, hogy a gain term-nek nem szabad sokkal gyorsabban csökkennie mint a sugárnak, mert különben túl kevéssé húzná magához az input a környezetében lévő outputokat. Programokban az alábbi gyakorlati értékekkel dolgozhatunk. Gain term függvény pl:

$$Gain(t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{t}{q})^2}, \quad radius(t) = \frac{m}{2} e^{-\frac{1}{2}(\frac{t}{s})^2}$$

ahol t jelőli az iterációk számát, q és s skálázó értékek az idő tengelyén, m jelöli az input pontok számát. A q és s értékek tetszőleges kis számok, de az algoritmus sebességének növelése érdekében érdemes előre meghatározni. A folyamat kritikus pontja a t_0 iterációszám, amikor a sugár nullára csökken. Ezután a pontok sorrendje már determinált lesz, és csak a nem approximált input pontok vonzódnak a legközelebbi ouputhoz. A gain term és a radius függvény vizsgálatát megtaláljuk Várady L. [87] és Hoffmann M. [41] cikkében.

5.1.3. A súlyok inicializálása

A folyamat legelején a neurális háló súlyainak kezdőértéket kell adni, azaz, inicializálni kell a súlyokat. Az inicializálás meghatározza a kontrolháló pontjainak kezdeti térbeli helyzetét. Különböző típusú input ponthalmaz esetén különböző inicializálásra van szükség. A legegyszerűbb megoldás, hogy kis véletlen számokat adunk meg. Ebben az esetben az output pontok távol lehetnek az input pontoktól. Az algoritmus hatékonyságát növeli, ha a súlyokat az input pontok környezetében (esetleges középpontja körül) inicializáljuk.

Ha nagy számú input ponttal rendelkezünk, vagy eloszlása ismert a pontoknak, akkor a súlyokat abban a pontban kell inicializálni, ahol az input

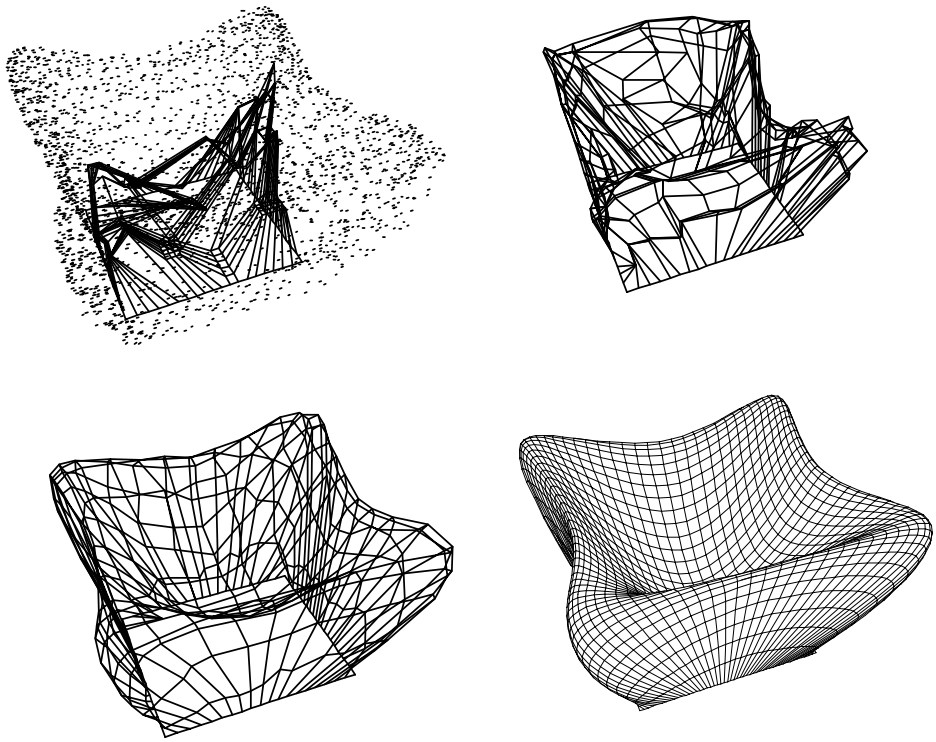
pontok sűrűsége a legnagyobb. Az input pontok osztályozásával (klaszterizációjával) ez a probléma megoldható.

Meg kell említenünk, hogy ha az input pontokat véletlenszerűen adjuk meg, akkor a kontrollháló kisimulhat, és néhány pont a kontrollhálón kívül lesz. Ezen pontokat már a tanulási folyamat során kiszűrhetjük, amivel csökkenthetjük a végrehajtási időt.

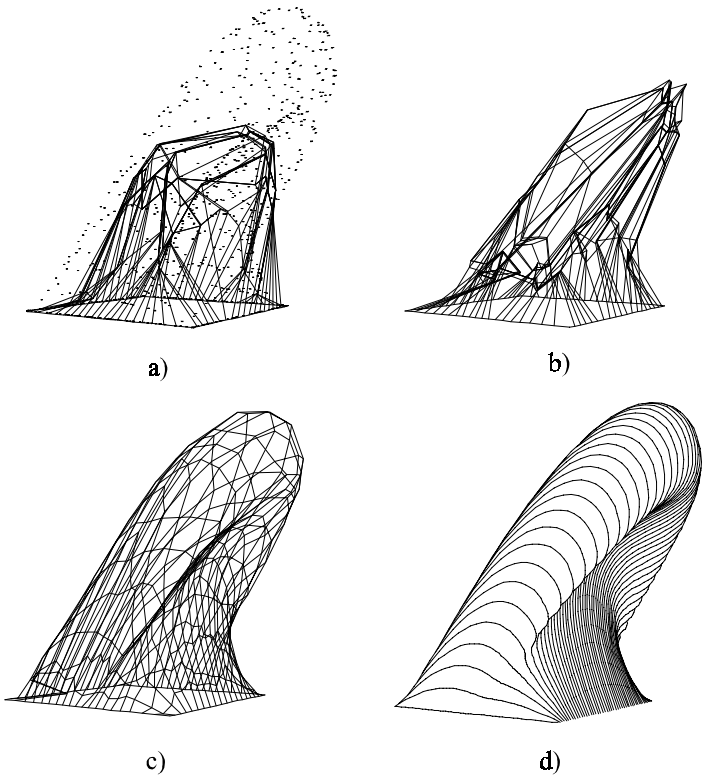
5.1.4. A felület

A tanulási folyamat után a Kohonen-háló előállít egy négyszögrácsot (grid), amely rácsháló figyelembe vehető valamely későbbi felület kontrollhálójaként. A felületet előállító technika kiválasztása (NURBS, Bézier felület, stb.) az algoritmus szempontjából nem lényeges. A kiindulási ponthalmaz rendezetlen volt (scattered), de a tanulási folyamat után mint egy rendezett kontrollháló pontjainak tekinthetjük. A 3.2. pontra visszautalva, akár base surface szerepet is betölthet a kapott felület, amely megoldhatja a felületillesztési problémák egyik kulcskérdését, a paraméter-inicializálás problémáját. A Kohonen-háló alkalmazására láthatunk konkrét példákat a 10. és a 11. ábrákon.

A 11. ábrán példát láthatunk arra, hogy konkáv input ponthalmaz esetén is működik az algoritmus. Az előző fejezetekben említett módszerek ilyen esetekben vagy hibás eredményre vezetnek, vagy rendkívül megnövekedik a számítási idő.



10. ábra. A Kohonen féle háló által előállított kontrollháló és felület



11. ábra. Egy speciális konkáv eset

5.2. A dinamikus Kohonen-háló alkalmazása ponthalmazra illesztett szabad formájú felületek előállítására.

Az előző fejezetben megismerkedhettünk a Kohonen-hálóval. A Kohonen-féle neurális hálózat kiválóan alkalmazható szabad formájú felületek approximációjában. Hoffmann és Várady adtak [43]-ben megoldást az alkalmazhatóságra. Korábbi módszerek általában háromszög alapú kontrollhálót konstruálnak a pontokból, pedig a klasszikus NURBS vagy Bézier felületek négyszög alapú kontrollhálót alkalmaznak. Ha adott a térben egy ponthalmaz (pontfelhő), akkor először az a feladatunk, hogy a neurális háló segítségével rendezzük sorba a pontokat és formáljunk négyszög topológiájú kontrollhálót, ezután standard approximációs vagy interpolációs módszerek alkalmazhatóak a felület előállítására. A módszer előnye, hogy egyaránt alkalmazható nagyszámú és kisszámú adat (input pont) esetén is.

Hoffmann, Várady és jómagam az előbbi módszer továbbfejlesztését adtuk meg [86]-ben. Amint már említettem, a Kohonen-háló előállít egy négyszög topológiájú kontrollhálót az inputként megadott pontfelhőből. A tanulási folyamat előtt meg kell adni egy kiindulási hálózatot, amely majd a folyamat során az input pontok felé fog mozogni, követve azok térbeli elhelyezkedését. Ennek a módszernek, az a hátránya, hogy jól kell megbecsülni a kiindulási hálózat neuronjainak, azaz későbbi kontrollháló csúcspontjainak a számát. Ha ez a szám túl kicsi, akkor néhány input pont kívül eshet az eredmény kontrollhálón, ha szükségtelenül nagy, akkor lényegesen megnövekedhet a számítási idő. A [43]-ben megadott módszer $4n$ neuront használ n db input pont esetén, amely megbízható választás, de nagy n esetén meglehetősen lassú az algoritmus. "Végtelen" input pont esetén bizonytalan a megfelelő neuronszám megválasztása Mivel az eredeti algoritmus [43]-ben, pontosan definiált, itt csak az új eredménynek számító módosítást közlöm.

5.2.1. A dinamikus Kohonen-háló

Az eredeti Kohonen-háló a neuronok két rétegéből áll. Az input réteg három neuronból áll, ezen neuronok kapják inputként a térbeli pontok koordinátáit. Az output réteg neuronjainak a száma m függ az input pontok n számától, általában $m = 4n$. Nagyon nagy számú input pont esetén alkalmas m-et kell választani. A két réteg neuronjai teljes összeköttetésben vannak egymással, tehát minden input neuron minden output neuronnal, és a kapcsolatokhoz

súlyokat rendelünk. Ezeket a súlyokat tekinthetjük az előre definiált topológiájú négyszög kontrollháló csúcspontjainak a térbeli koordinátáiként. Szemléletesen is látható, hogy m értékének megválasztása kritikus pontja az algoritmusnak.

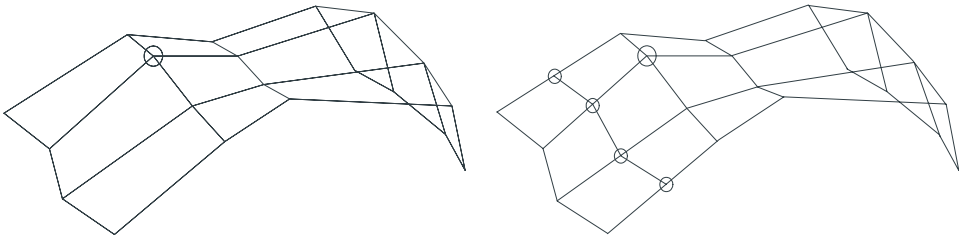
A problémára [86]-ben megoldást adtunk, azaz a Kohonen-háló dinamikus változatát vezettük be. Az alapötlet az, hogy az output neuronok száma, ennél fogva a súlyok száma, azaz a kontrollháló csúcspontjainak száma növekedjen dinamikusan a tanulási folyamat során. Minden iterációban kiválasztunk egy aktív neuront, amely a legközelebb van az input ponthoz. Ezt a csúcspontot és a szomszédjait mozgatjuk az input pont felé. Feltételezzük, hogy a kiválasztott neuront vagy a neuron környezetében lévő neuronokat igen sűrűn aktiváljuk. Ez geometriailag azt jelenti, hogy ha a háló ezen része nem elég sűrű, vagyis nagyon sok input pont található ezen neuron vagy a neuron környezetében lévő neuronok közelében, akkor megoldásként extra neuronok beszúrásával próbálkozunk a leggyakrabban választott neuron mellett. Mivel egy neuron beszúrása deformálná a kontrollháló szerkezetét, ezért vagy neuronok egy sorát, vagy egy oszlopát szúrjuk be. Az aktív neuron azon oldalára szúrjuk be az új neuronokat, amely oldalon a szomszéd neuronok legkevésbé aktívak. Ezzel a módszerrel a kontrollháló dinamikusan fog nőni, és a tanulási algoritmus végére megfelelő méretű lesz.

5.2.2. Új neuronok beszúrása

A következőkben bemutatjuk a beszúrás algoritmusának formális megadását. Legyen az output neuronok száma $k \times l$ (az eljárás elején $k = 2, l = 2$). Rendeljünk $\lambda_{ij}, (i = 1, ..., k; j = 1, ..., l)$ aktivitási változót minden output neuronhoz. Az (i_0, j_0) aktivitási változó értékét növeljük 1-gyel, amikor (i_0, j_0) output neuron aktív. Ezenkívül meg kell adni a λ_{ij} aktiválási változó Λ felső korlátját is. Ha valamelyik aktiválási változó értéke egyenlő Λ-val (azaz ez a neuron Λ-szor volt kiválasztva), akkor egy sort, vagy egy oszlopot szúrunk be a kiválasztott neuron mellé.

A korábbi tanulási lépések után (amelyet már a 5.1.1.pontban ismertettem, lásd még [43]), a következő lépéseket kell végrehajtani:

1. Minden aktivitási változót összehasonlítunk a határértékkel. Ha bár-



12. ábra. Új neuronok beszúrása a legaktívabb neuron mellé

milyen ij indexpár esetén igaz, hogy

$$\lambda_{ij} = \Lambda,$$

akkor

2. Válasszuk ki a neuron legkevésbé aktív szomszédját, azaz határozzuk meg a
$$\min\left(\lambda_{i-1j}, \lambda_{ij-1}, \lambda_{i+1j}, \lambda_{ij+1}\right).$$
számot. (Ha a neuron a rácsháló határán helyezkedik el, akkor előfordulhat, hogy a zárójelen belül valamelyik érték nem létezik.)

3. Tegyük fel, hogy a megtalált neuron az $(i, j-1)$ neuron. Ekkor egy oszlopot szúrunk be a $(j-1)$ és a j oszlopok közé. A keletkező új oszlop minden egyes neuronjához tartozó súlyok értéke egyenlő lesz a két szomszédos neuronhoz tartozó súlyok átlagával. A súlyok átlagolását az indokolja, hogy a kapott új neuronok a szomszédos neuronok szakaszfelezőpontjaiban vannak. A k és l értéke megfelelően növekszik, jelen esetben $l = l + 1$.

4. Az összes aktiválási változót alaphelyzetbe állítjuk vissza
$$\lambda_{ij} = 0, \quad (i = 1, ..., k; j = 1, ...l).$$

További előnye a beszúrásnak, hogy k és l értéke, azaz a kontrollháló alakja automatikusan változik a tanulási folyamat során. Az eredeti módszer (lásd [43]) megállási feltételét is megváltoztathatjuk. Ha az input pontok száma viszonylag kicsi, akkor az eredeti algoritmusban akkor mondjuk,

hogy a neurális háló tanult, vagy kiképzett, ha minden input pont illeszkedik a kontrollhálóra. Ebben az esetben olyan interpolációs módszert és felületet találhatunk, amelyre az összes input pont illeszkedik. Ha az input pontok száma százas nagyságrendű, vagy az adatok bizonyos eloszlással vannak megadva, amelyek pontfelhős (scattered data) problémáknál fordulnak elő, akkor a neurális hálót kiképzettnek mondjuk, ha a kontrollháló változása egy előre megadott értéknél kisebb. Az utóbbi esetben a dinamikus Kohonen-háló tanulási folyamata akkor is befejeződhet, ha az output neuronok száma elér egy előre meghatározott értéket, sőt ez az érték az input pontok számánál kisebb is lehet.

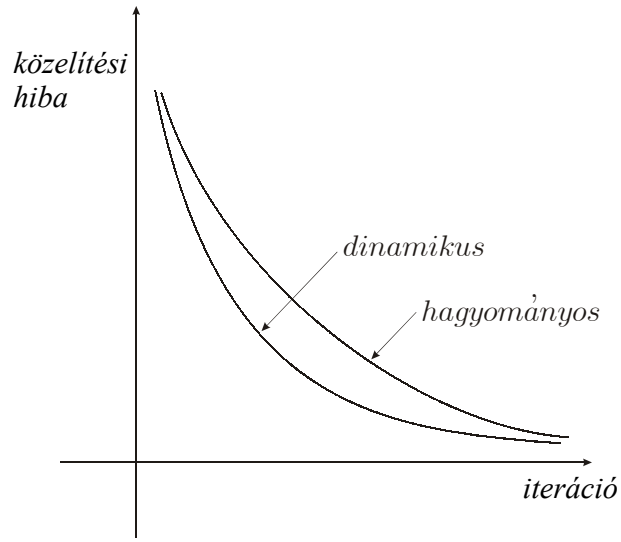
5.2.3. Az új algoritmus hatékonysága

Az új algoritmusnak két előnye van. Az első az, hogy lényegesen csökken a tanulási iterációk száma, azaz, hogy hányszor adjuk meg az input pontok koordinátáit. Igaz, hogy az új neuronok beszúrása plusz feladat, de ez elhanyagolható számítási időt igényel, így a dinamikus változat gyorsabb, mint az eredeti. A második előny az, hogy a tanulás után megkapott kontrollháló csúcspontjainak száma jóval kevesebb lesz, és ez általában simább felületet eredményez. Az alábbi táblázat az iterációk számának változását mutatja (ezerszer ismételt futtatás utáni átlagértékek).

Input pontok	*eredeti*		*dinamikus*	
	iteráció	*neuronok*	*iteráció*	*neuronok*
10	3200	40	2300	30
100	5500	400	3600	250

Még egyszer meg kell jegyeznünk, hogy a Kohonen-háló egy kontrollhálót állít elő pontfelhőből, azaz bizonyos rendezést hajtunk végre a pontfelhő pontjain. Tehát az eljárás szempontjából teljesen irreleváns, hogy később milyen standard felület approximációs vagy interpolációs módszert alkalmazunk, amely a kontrollháló alapján előállítja a szabad formájú felületet.

Felix Hausdorff (1868 -1942) a metrikus tér két részhalmazán értelmezett metrikus függvényt definiált. Két halmaz Hausdorf-távolsága akkor és csak akkor r ha az egyik halmaz tetszőleges pontjának a távolsága a másik halmaz valamely pontjától kisebb egyenlő mint r. Formálisan: Tekintsünk



13. ábra. A dinamikus és hagyományos technika összehasonlítása

egy X metrikus teret a d távolságfüggvénnyel. $x \in X$ pont és $A \subseteq X$ nemüres részhalmaz esetén a következőképpen értelmezzük x *távolságát* A-tól

$$\delta_H(x, A) := \inf_{a \in A} \delta_X(x, a).$$

Ekkor A és B nemüres halmazok *Hausdorff-távolsága*

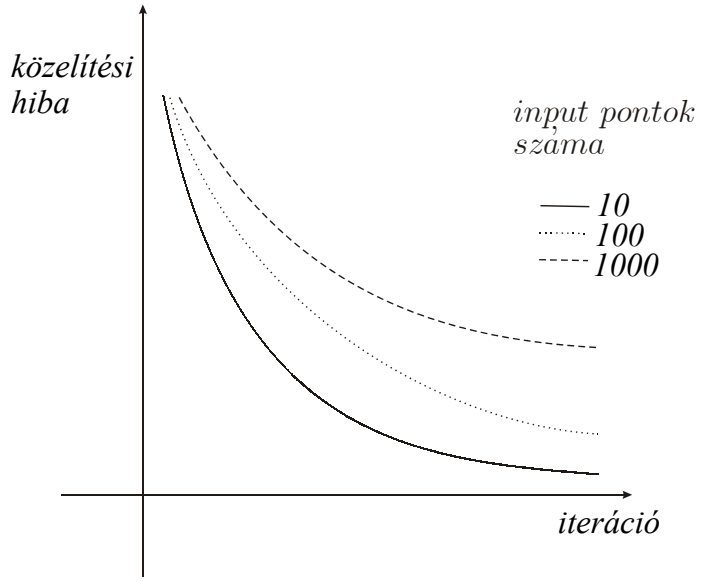
$$\delta_H(A, B) := \max(\delta_{asym}(A, B), \delta_{asym}(B, A)), \quad \text{ahol} \tag{18}$$

$$\delta_{asym}(A, B) := \sup_{a \in A} \delta_H(a, B), \tag{19}$$

az ily módon megadott távolságot gyakran az A és B halmaz *aszimetrikus Hausdorff-távolságának* is nevezik.

Ezt a Hausdorff távolságot alkalmaztuk, a ponthalmaz és a Kohonen-hálóval kapott approximáló felület közelítési hibájának a meghatározására. A 13. ábrán jól látható, hogy a dinamikus változat közelítési hibája meredekebben csökken az iteráció számának növekedésével.

A (18)- ben definiált Hausdorff-távolságot alkalmazva vizsgálhatjuk az input rögzítésével kapott eredményeket is. A 14. ábrán jól látható, hogy rögzített inputszám esetén hogyan változik a közelítési hiba az iteráció számának növekedésével.



14. ábra. A közelítési hiba változása rögzített input szám esetén

5.3. B-spline vonalfelületek konstruálása Kohonen-háló segítségével

A következőkben olyan eredményeket közlünk, amelyek megtalálhatóak a [38] cikkben, amelyet Hoffmann Miklóssal közösen készítettünk. A vonalfelületek és speciális részhalmazaik, a kifejthető felületek, jólismert területei a klasszikus geometriának. Erről a területről számos komputergeometriai és CAD-CAM alkalmazást ismerünk. Az utóbbi területen a felületek megadására szokásos eljárás a különböző típusú spline felületek, mint például a Bézier, B-spline vagy a NURBS felület alkalmazása. Ebből következik az is, hogy megoldható a vonal és kifejthető felületeket megadása, illetve konstruálása spline felületek segítségével is. Lefejtés alatt érthetünk olyan izometrikus leképezést is, ahol az egyik felületet képezzük a másik felületre, ahol az utóbbi felület nem feltétlenül sík. Számos eredményt ismerünk a szakirodalomból hasonló approximációk és interpolációk létrehozására (lásd [49], [68]).

Bármilyen típusú spline felület szokásos megadása négyszögalapú kontrollhálóval történik, amely a vonalfelületek esetében $(2, n)$ típusú. Az is-

mert módszerek legfontosabb feladata éppen e kontrollháló létrehozása. Az általunk használt módszerben Kohonen-féle neurális hálót használunk az input adatok előfeldolgozása során, hogy előállítsuk ezen kontrollhálót. A Kohonen-hálót, mint a mesterséges intelligencia egyik igen hasznos eszközét, szerzőtársaimmal együtt már sikeresen alkalmaztuk más approximációs problémák esetében is (Lásd [42], [86]).

A mi esetünkben az eredeti input adatstruktúra adott egyenesek halmazát tartalmazza, amelyet alkotóknak (rulings) nevezünk. Előnyős számunkra ezen egyenesek és a neurális háló megadása homogén koordinátákkal, amely egy jól ismert eljárás a projektív geometriában. Hasonló komputergeometriai megközelítés találunk Bodduluri és Ravani cikkében [8]. Projektív egyenesek Plücker-koordinátákkal való megadásának felhasználását megtaláljuk Hlavaty [36] munkájában, amelyet Chen és Pottmann ([16]) szintén felhasználtak cikkükben.

Vonal és kifejthető felületek definícióját megtaláljuk a dolgozat későbbi 5.4. fejezetében. Itt csak hivatkozunk ezen definíciókra.

5.3.1. Plücker-koordináták (vonalkoordináták) megadása

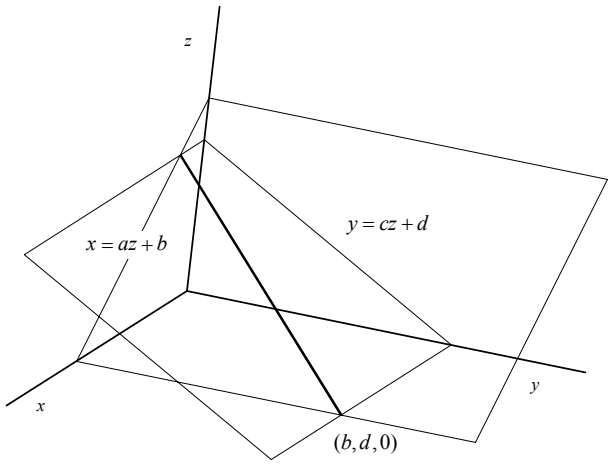
Egy korábbi fejezetben már levezettük Plücker-koordinátákat. A most következő levezetésben azt mutatjuk meg, hogy a Plücker-koordináta megegyezik a vonalgeometriában használt vonalkoordináta definiciójával.

Plücker 1868-ban a Neue Geometrie des Raumes, gegründet auf die Betrachtung der geraden Linien als Raumelement c. munkájában vezette be a vonalkoordinátákat. A matematikai fogalmak általában a magyarázatukat tekintve folyton alakulnak, egyszerűsödnek, új oldalról nyernek megvilágítást. Ahogyan most ezt leírjuk, azt elsősorban a dolgozatunk szempontjából tesszük így.

Az Euklideszi térben felveszünk egy Descartes-féle koordinátarendszert. Az általános helyzetű egyenest két, a koordinátasíkokra eső vetületével adjuk meg.

$$\begin{aligned} x &= az + b \\ y &= cz + d \end{aligned}.$$

Az egyenleteket vetítősíkok egyenleteként is felfoghatjuk. Ekkor e síkok metszésvonala lesz az egyenes. Az egyenes az x, y síkot a $(b, d, 0)$ koordinátájú pontban metszi.



15. ábra. Egyenes megadása vetítősíkok metszeteként

Az egyenletpár nem állítja elő az $[x, y]$ síkkal párhuzamos egyeneseket. Ilyenkor az egyik egyenest felcseréljük egy $[x, y]$ síkon lévő egyenesre pl. $y = ex + f$-re, azaz z irányú vetítősíkot vezetünk be. Hogy a tér melyik egyeneséről van szó azt kizárólag négy paraméter határozza meg, azaz a tér összes egyenese egy négyparaméteres sokaságot alkot.

Ha a paraméterek közül nem mind szabad paraméter, akkor valamilyen egyenessokaságot kapunk. Nevezetesen, ha három szabad paraméter van, akkor *egyeneskomplexus*, vagy röviden *komplexus,* ha kettő, akkor *kongruencia*, ha pedig csak egy, akkor *vonalfelület* az egyenessokaság.

Ha homogén koordinátákra térünk át, érdekes dolgokat kapunk. A projektív térben a pontot, vagy duálisát a síkot, számnégyessel jellemezzük. A számnégyesek arányosság erejéig tartoznak a ponthoz vagy síkhoz, és mind nem lehet nulla. Amikor egyenest adunk meg, azt számnyolcassal tehetjük tehát meg a következő módon:

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{pmatrix}.$$

Az pedig lényegtelen, hogy a mátrix egy sorát pontnak, vagy síknak tekintjük. Fontos, hogy a mátrix rangja 2, mert egyébként a két pont, illetve a két sík azonos lenne. Két egyenes pedig akkor megegyező, ha a

koordinátákból képezett mátrix rangja 2.

$$rang \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ u_1 & u_2 & u_3 & u_4 \\ v_1 & v_2 & v_3 & v_4 \end{pmatrix} = 2.$$

Az egyenes koordinátáiból 2×2 -es determinánsokat képezünk az összes lehetséges módon. Minden determináns nem lehet nulla, hiszen a 2×4 -es mátrix rangja 2.

$$p_{12} = \begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix}, \quad p_{13} = \begin{vmatrix} x_1 & x_3 \\ y_1 & y_3 \end{vmatrix}, \quad p_{14} = \begin{vmatrix} x_1 & x_4 \\ y_1 & y_4 \end{vmatrix},$$

$$p_{23} = \begin{vmatrix} x_2 & x_3 \\ y_2 & y_3 \end{vmatrix}, \quad p_{24} = \begin{vmatrix} x_2 & x_4 \\ y_2 & y_4 \end{vmatrix}, \quad p_{34} = \begin{vmatrix} x_3 & x_4 \\ y_3 & y_4 \end{vmatrix}.$$

Figyelembevéve, hogy egy oszlopcsere a determináns értékének -1 -szeresét adja, azaz $p_{ik} = -p_{ki}$, rövid számolás után megállapíthatjuk, hogy

$$p_{12}p_{34} + p_{13}p_{42} + p_{14}p_{23} = 0.$$

Ha a fenti mennyiségeket nem egy általános projektív térben, hanem a végtelen távoli elemekkel kibővített euklideszi térben írjuk fel, akkor érdekes geometriai tulajdonságokat látunk. Tehát inhomogén koordinátákra térve:

$$X_i = \frac{x_i}{x_4}, \quad Y_i = \frac{y_i}{y_4}.$$

Az egyenes kiindulási alakja ekkor így alakul:

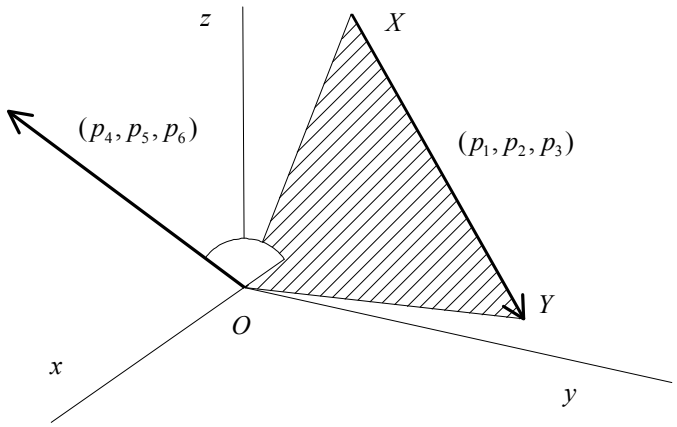
$$\begin{pmatrix} X_1 & X_2 & X_3 & 1 \\ Y_1 & Y_2 & Y_3 & 1 \end{pmatrix}.$$

Bevezetjük még az alábbi jelöléseket is:

$$p_1 = p_{41}, \quad p_2 = p_{42}, \quad p_3 = p_{43}, \quad p_4 = p_{23}, \quad p_5 = p_{31}, \quad p_6 = p_{12}.$$

Ekkor

$$p_1 = Y_1 - X_1, \quad p_2 = Y_2 - X_2, \quad p_3 = Y_3 - X_3,$$



16. ábra. Az egyenes vonalkoordinátái

azaz ez pontosan az egyenes irányvektorának három koordinátája. A következő

$$p_4 = \begin{vmatrix} X_2 & X_3 \\ Y_2 & Y_3 \end{vmatrix}, \quad -p_5 = \begin{vmatrix} X_1 & X_3 \\ Y_1 & Y_3 \end{vmatrix}, \quad p_6 = \begin{vmatrix} X_1 & X_2 \\ Y_1 & Y_2 \end{vmatrix}$$

értékek még így is írhatók:

$$\begin{vmatrix} i & j & k \\ X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \end{vmatrix}.$$

Vagyis a p_4, p_5, p_6 mennyiségek vektoriális szorzat eredményeként egy másik vektor koordinátái, éspedig az origóból az egyenes irányvektorának kezdőpontjába illetve végpontjába mutató vektorok vektoriális szorzata.

A hat p_i arányosság erejéig tartozik az egyeneshez, mivel homogén koordinátákból származtattuk őket. De a hat számból egyet rögzítve az egyenest is megkapjuk. Az első három megadja az irányvektort a hosszával együtt. Az origótól való távolság szorozva az irányvektor hosszával a sraffozott háromszög területének a duplája ami a második három számból számolható ki, mint a vektoriális szorzat abszolút értéke. A sraffozott sík, benne az egyenessel úgy áll, hogy merőleges a vektoriális szorzatra, és az origótól való távolsága a normálvektor és az irányvektor abszolút értékeinek a hányadosa. Ezzel az egyenest a hat számból visszaállítottuk.

A tér egyeneseit a fenti hat számmal mint koordinátákkal lehet jellemezni. A rendezett hat számot Plücker-féle koordinátáknak hívjuk.

Észrevételek:.

1. A Plücker-koordináták lehetővé teszik, hogy a háromdimenziós projektív tér egyeneseit modellezzük az ötdimenziós projektív tér pontjaiként.

2. A hat koordinátából öt adható meg szabadon, a hatodikkal biztosíthatjuk, hogy a két vektor merőleges legyen egymásra, a skaláris szorzatuk 0 legyen.

3. A Plücker-koordináták fenti leírása módot ad felületi egyenesek szemléletes megadására.

4. Az első három szám az irányvektor, ez a saját egyenesén bárhol lehet. A sraffozott háromszög területe egyenlő az utolsó három szám, mint a vektor abszolút értékével.

5. Egy vonalfelület akkor síkbafejthető, ha az alkotók mentén az érintősík konstans. Ha az O, X, Y sraffozott háromszög az X, Y alkotójú felület érintősíkja, és az alkotó második három koordinátája konstans, akkor az alkotó egy síkbafejthető felület alkotója.

5.3.2. Vonalfelület konstruálása előre megadott egyenesekből

Tekintsük a következő problémát. Legyen adott egyenesek tetszőleges halmaza, és keressük meg egy vonalfelületet, amely illeszkedik ezen egyenesekre, azaz a megadott egyenesek a vonalfelület alkotói lesznek.

Pontosabban, interpolálni szeretnénk a megadott egyeneseket valamilyen CAD, Bézier, B-spline, vagy NURBS felülettel. A dolgozatban B-spline felületet használunk, megemlítve, hogy minden felület kontrollpontok illetve kontrollháló által definiált, amely általában négyszögháló, és igény szerint interpolálhatjuk vagy approximálhatjuk ezen pontokat.

Használni fogjuk a következő jól ismert tulajdonságot:

1. Tétel. *Ha egy B-spline felület kontrollhálójának pontjai az egyik irányban egy egyenesre illeszkednek, azaz a háló* $(2, n)$ *típusú, akkor a konstrukció során keletkezett B-spline felület, vonalfelület lesz.*

A tétel bizonyítása a B-spline felület tulajdonságai alapján egyszerűen elvégezhető.

2. Megjegyzés. *Sajnos, a tétel általánosítása nem igaz, miszerint, ha egy B-spline felület kontrollhálójának valamelyik irányú poligonjai egyenesekre illeszkednek, akkor a felület vonalfelület. Ismereteim szerint az általános esetben ($n \times m$, $n > 2$, $m > 2$ kontrollpontra) nincs szükséges és elégséges feltétel arra nézve, hogy a B-spline felület mikor lesz vonalfelület. Természetesen vannak speciális esetek, amikor tudjuk, pl. ha párhuzamos egyenesekre illeszkednek a kontrollpontok, ekkor gyakorlatilag arról van szó, hogy egy iránnyal párhuzamosan tolunk el egy görbét, ami nyilván vonalfelület.*

A fenti tétel alapján létre kell hoznunk egy olyan $(2, n)$ típusú négyszöghálót, amelynek csúcspontjai egyenesekre illeszkednek kiindulásként előre megadott egyenesek irányában. Ha a kontrollpontokat interpoláljuk, akkor ezek az egyenesek lesznek az alkotói a későbbi felületnek, és ez a felület, vonalfelület lesz. A négyszögháló (vagy nevezhetjük négyszögrácsnak is) megkeresésére fogjuk használni a Kohonen-hálót. Ezen eszköz biztosítani fogja a megfelelő kontrollhálót, és ez a háló lesz az input adatsora a standard B-spline felületinterpolációnak.

A Kohonen neurális háló erős öntanuló tulajdonsággal rendelkezik, ez azt jelenti, hogy a tanulási folyamat alatt a topológikusan invariáns rács (grid) mozog a pontfelhő pontjai felé, és próbálja követni a pontfelhő eloszlását és struktúráját.

Ez a hálózat folytonosan kiértékelt kétrétegű hálózat. Az első réteg az input réteg, az itt lévő három input neuron értéke a térbeli pontok koordinátái. Számos neuron alkotja az output réteget, amely a négyszöghálót formázza meg. A két réteg neuronjai összeköttetésben állnak egymással, melyek súlyozottak Ezen súlyok a tanulási folyamat során módosulnak bizonyos tanító algoritmusok segítségével. Mivel minden output neuronnak van három összeköttetése három súlyozással, ezért ezen súlyok tekinthetőek azon pontok térbeli koordinátájainak, amelyek output neuronokkal megegyező topológiával alkotják a keresett pontrácsot. A pontrács mozog a megadott pontok felé, és a tanulási folyamat után illeszkedik az összes input pontra. A módszer részletes ismertetése és a Kohonen-háló bemutatása az 4.1. pontban, és a már említett cikkek mellett a [54]-ban található meg.

Mindemellett az említett módszert módosítanunk kell, mert az input struktúra pontok helyett most egyeneseket tartalmaz, és a rácsszerkezetnek a fentebb említett tulajdonsággal kell rendelkeznie. Megoldásként fel kell használnunk az 3.3. pontban részletesen ismertetett Plücker-koordinátákat. A jobb megértés kedvéért és a speciális alkalmazás miatt részletesen ismertetjük az eljárást. Meg kell említenünk, hogy a Magyarországon szokásos terminológiával ellentétben, a külföldi szakirodalomban gyakori, hogy a homogén koordinátát előre írják. Azaz, ha át akarunk térni a homogén koordinátáról Descartes-féle koordinátákra, akkor a következő összefüggést használják:

$$(x_0, x_1, x_2, x_3) \longmapsto \left(\frac{x_1}{x_0}, \frac{x_2}{x_0}, \frac{x_3}{x_0}\right) \in \mathbb{R}^3, \quad \text{ha } x_0 \neq 0.$$

A dolgozatban a nálunk szokásos terminológiát követjük

$$(x_1, x_2, x_3, x_4) \longmapsto \left(\frac{x_1}{x_4}, \frac{x_2}{x_4}, \frac{x_3}{x_4}\right) \in \mathbb{R}^3, \quad \text{ha } x_4 \neq 0.$$

Tekintsük a $\mathbb{P}^3$ háromdimenziós projektív teret. Itt minden pontnak négy homogén koordinátája van (x_1, x_2, x_3, x_4), amelyből meghatározhatóak a Descartes-koordináták, feltéve, hogy nem végtelen távoli pontról van szó. Azaz, ha $x_4 \neq 0$, akkor (x, y, z) koordináták esetén $x = x_1/x_4, y = x_2/x_4, z = x_3/x_4$. Tekintsük azt az $l \in \mathbb{P}^3$ egyenest, amely illeszkedik (x_1, x_2, x_3, x_4) és (y_1, y_2, y_3, y_4) koordinátákkal megadott pontokra. Ezen l egyenes hat Plücker-koordinátája a következőképpen adódik:

$$(l_1, ..., l_6) := (l_{41}, l_{42}, l_{43}, l_{23}, l_{31}, l_{12}), \quad l_{ij} = x_i y_j - x_j y_i. \tag{20}$$

Ezen koordináták függetlek az egyenesre illeszkedő két pont kiválasztásától, és kielégítik a Plücker-féle azonosságot:

$$l_1 l_4 + l_2 l_5 + l_3 l_6 = 0. \tag{21}$$

Ilymódon egy kölcsönösen egyértelmű megfeleltetést hoztunk létre a valós számok hatosa és a $\mathbb{P}^3$ egyenesei között, amely egy másik kölcsönösen egyértelmű megfeleltetést eredményez a $\mathbb{P}^3$ egyenesei és az $\mathbb{P}^5$ ötdimenziós projektív tér pontjai között. Az utóbbi leképezés segítségével tudjuk beágyazni az adott egyeneseket a későbbi felület kontrollhálójába. Ezzel a módszerrel a Kohonen-hálót a $\mathbb{P}^5$ tér pontjaival tanítjuk, azaz az input réteg hat

neuront fog tartalmazni, mialatt az output réteg $\mathbb{P}^5$-ben poligon lesz. Amikor a tanulási folyamat befejeződött, az eredményül kapott poligon áthalad a megadott $\mathbb{P}^5$-beli pontokon. Visszatranszformálva $\mathbb{P}^3$-ba megkapjuk az adott egyeneseket és a négyszöghálót, amely tartalmazza a kiindulásként megadott egyeneseket.

5.3.3. Az algoritmus bemutatása

Legyen adott az $l^i, i = (1, \ldots, n)$ egyenesek halmaza. Határozzuk meg ezen egyenesek *Plücker- koordinátáit:*

$$l^i \left(l^i_j\right) \quad i = 1, \ldots, n \quad j = 1, \ldots, 6.$$

Legyen adott egy $n = 6$ input neuronból és $m(= 4n)$ output neuronból álló Kohonen-háló. Jelölje w_{ij} a j-dik output neuron és az i-dik input neuron összeköttetéséhez rendelt súlyt. Azokat a $\left(w_{i_0 j}\right), (j = 1, \ldots, 6)$ súlyokat, amelyek az output neuronokhoz tartozó összeköttetéshez vannak rendelve, tekinthetjük a $\mathbb{P}^5$ projektív tér egy V_i output pontjához tartozó koordinátáinak

Elkezdve a Kohonen-háló tanítását először inicializáljuk a w_{ij} súlyokat kis véletlen értékekkel (egyéb lehetséges inicializálásról később ejtünk szót). $t = 1$ kezdőértéket rendelünk a tanulási időhöz, míg a véletlenszerűen kiválasztott $\left(l^{i_0}_j\right), (j = 1, \ldots, 6)$ input pont koordinátái adottak. Meghatározzuk minden output pont d_m euklideszi távolságát az input ponttól,

$$d_m = \left(\sum_{s=1}^{6} \left(l^{i_0}_j - w_{ms}\right)^2\right)^{\frac{1}{2}}.$$

Válasszuk ki azt az aktív neuront, amely legközelebb van az input neuronhoz, majd változtassuk meg a neuron és környezetében lévő neuronokhoz tartozó súlyait, a következő összefüggés szerint:

$$w_{ij}\left(t+1\right) = w_{ij}\left(t\right) + \eta\left(t\right)\left(l^{i_0}_j - w_{ij}\left(t\right)\right),$$

ahol az $\eta(t)$ függvény egy időben csökkenő tanulási hányados.

Miután megváltoztattuk a súlyokat, a tanulási folyamat végéig új véletlen módon kiválasztott inputtal dolgozunk tovább. Akkor tekintjük befejezettnek a tanulási folyamatot, ha minden input egyenes a kontrollhálón

található. Ha az input halmaz egyenesek százait tartalmazza, akkor nagyon megnövekedhet a futási idő, ezért hasznos leállási feltételnek tekinthetjük azt is, ha a súlyok változása egy előre meghatározott küszöbérték alá esik.

Amikor a tanulási folyamat befejeződött, akkor az eredmény egy poligon $\mathbb{P}^5$-ben, amelynek csúcspontjai

$$V_i(w_{ij}), \quad (i = 1, ..., m), \quad (j = 1, ..., 6).$$

Ezen csúcspontokat visszatranszformálva $\mathbb{P}^3$-ba a (w_{ij}) Plücker-koordinátákkal megadott v_i projektív egyeneseket kapjuk. A Plücker-koordináták definíciója alapján kiszámíthatjuk ezen egyenesek pontjait. Az ilyen módon kiszámolt egyenesek halmazát az eredetileg megadott input egyenesekből nyertük.

A tanulási periódus alatt mindig kiszámíthatjuk a teljes rácshálót minden egyes t időpontban, felhasználva a v_i egyenesek pontjait, mint csúcspontok, amelyeket összekötünk a másik irányban is. Tehát a tanulási folyamat után az alkotók mellett az alkotókat tartalmazó négyszöghálót is megkapjuk. Ezen háló csúcspontjai input kontrollpontként szolgálhatnak bármilyen standard szabad formájú felületet előállító algoritmus számára. Ennélfogva, interpolálni vagy approximálni tudjuk ezeket a pontokat és alkotókat.

5.3.4. Egyenesek előállítása a Plücker-koordinátákból

Ebben a részben azt ismertetjük, hogyan lehet előállítani az egyeneseket $\mathbb{P}^3$-ban az algoritmus eredményeként $\mathbb{P}^5$-ben kapott poligon V_i csúcspontjaiból. Tekintsük ebből a poligonból egy tetszőleges V_0 csúcspontot. Az algoritmus meghatározta a csúcspont $(l_{41}, l_{42}, l_{43}, l_{23}, l_{31}, l_{12})$ Plücker-koordinátáit, amely koordinátákra teljesül a Plücker-féle azonosság, azaz

$$l_{41}l_{23} + l_{42}l_{31} + l_{43}l_{12} = 0.$$

Tekintsük a következő négy vektort:

$$\begin{aligned}
\mathbf{s}_0 &= (\quad 0, \quad l_{41}, \quad l_{42}, \quad l_{43}), \\
\mathbf{s}_1 &= (-l_{41}, \quad 0, \quad l_{12}, -l_{31}), \\
\mathbf{s}_2 &= (-l_{42}, -l_{12}, \quad 0, \quad l_{23}), \\
\mathbf{s}_3 &= (-l_{43}, \quad l_{31}, -l_{23}, \quad 0).
\end{aligned}$$

Igaz az, hogy az $\mathbf{s}_j$ $(j = 1, \ldots, 4)$ vektorok közül bármelyik kettőt választjuk ki, akkor a belőlük a (20) összefüggés alapján előállított számhatos skalárszorosa lesz az $(l_{41}, l_{42}, l_{43}, l_{23}, l_{31}, l_{12})$ számhatosnak. A bizonyítást lásd H. Pottmann és J. Wallner könyvében, a Models of Line Space című fejezetben [70]. Ez a négy vektor megadja a Plücker-koordinátához tartozó egyenes és a koordinátasíkok metszéspontjait. Például az egyenes és az $x = 0$ sík, azaz a $[y, z]$ koordinátasík metszéspontja az $M\left(0, \frac{l_{12}}{-l_{41}}, \frac{l_{31}}{l_{41}}\right)$ pont, feltéve, hogy $l_{41} \neq 0$.

Felhasználva az előbbi eredményt vagy meg tudjuk határozni az egyenes két valós metszéspontját, vagy a keresett egyenes párhuzamos lesz valamelyik koordinátatengellyel. Az utóbbi esetben is meg tudjuk adni az egyenes két pontját.

Az egyenes két pontjával adott, de nem feltétlenül ezekkel a pontokkal érdemes továbbdolgoznunk, mivel ezek a pontok és az általuk meghatározott egyenes szakasz nem feltétlenül esik az inputként megadott szakaszok környezetébe. Viszont két pontjával megadott egyenes tetszőleges pontját meghatározhatjuk az egyenes paraméteres egyenletrendszere alapján. A P_0 (x_0, y_0, z_0) és P_1 (x_1, y_1, z_1) pontokon áthaladó egyenes egyenletének paraméteres alakja

$$\begin{aligned} x &= x_0 + (x_1 - x_0)t, \\ y &= y_0 + (y_1 - y_0)t, \\ z &= z_0 + (z_1 - z_0)t. \end{aligned} \tag{22}$$

Ez az egyenletrendszer igen kól kezelhetőnek bizonyul általában a számítógépi grafikában, s a mi esetünkben is. Ha a input egyenesek kezdőpontja ugyanarra a síkra illeszkedik, akkor megfelelő eltolás és elforgatás után tekinthetjük ezt a síkot $z = c$ síknak, ahol c egy valós konstans. Ekkor a (22) paraméteres egyenletrendszer harmadik egyenletébe behelyettesítve meghatározhatjuk t-t, majd az x és az y koordinátákat. Tehát viszonylag egyszerűen juthatunk olyan szakaszsorozathoz, amely az input egyenesek környezetében helyezkedik el.

További probléma keletkezik abból is, hogy a Kohonen-hálóval előállított szakaszok irányítása véletlenszerű. A szakaszok végpontjai szolgálnak inputként a Bézier, B-spline vagy NURBS felületet előállító algoritmus számára, mivel ezek a pontok határozzák meg a kontrollháló csúcspontjait.

Könnyen előfordulhat, hogy nem várt csavarodás keletkezik a felületen. Úgy javíthatunk ezen a helyzeten, hogy figyeljük az egymásután következő szakaszokból képzett vektorok skaláris szorzatát. Ha az i-edik és az $(i+1)$-edik vektor skaláris szorzata negatív, azaz tompaszöget zárnak be, akkor az $(i+1)$-edik vektor irányítását megfordítjuk a vektort meghatározó szakasz végpontjainak a felcserélésével. A programozás során felhasználtuk azt a jól ismert összefüggést, hogy definíció szerint a háromdimenziós térben $\mathbf{v}(vx, vy, vz)$ és $\mathbf{w}(wx, wy, wz)$ vektorok skaláris szorzata

$$(\mathbf{v}, \mathbf{w}) = |\mathbf{v}|\,|\mathbf{w}| \cos\phi,$$

ahol ϕ a két vektor hajlásszöge, és a $|\mathbf{v}|$ és $|\mathbf{w}|$ a vektorok hosszát jelöli. Továbbá igaz az, hogy a vektorok skaláris szorzata egyenlő a vektorkomponensek szorzatának az összegével, azaz

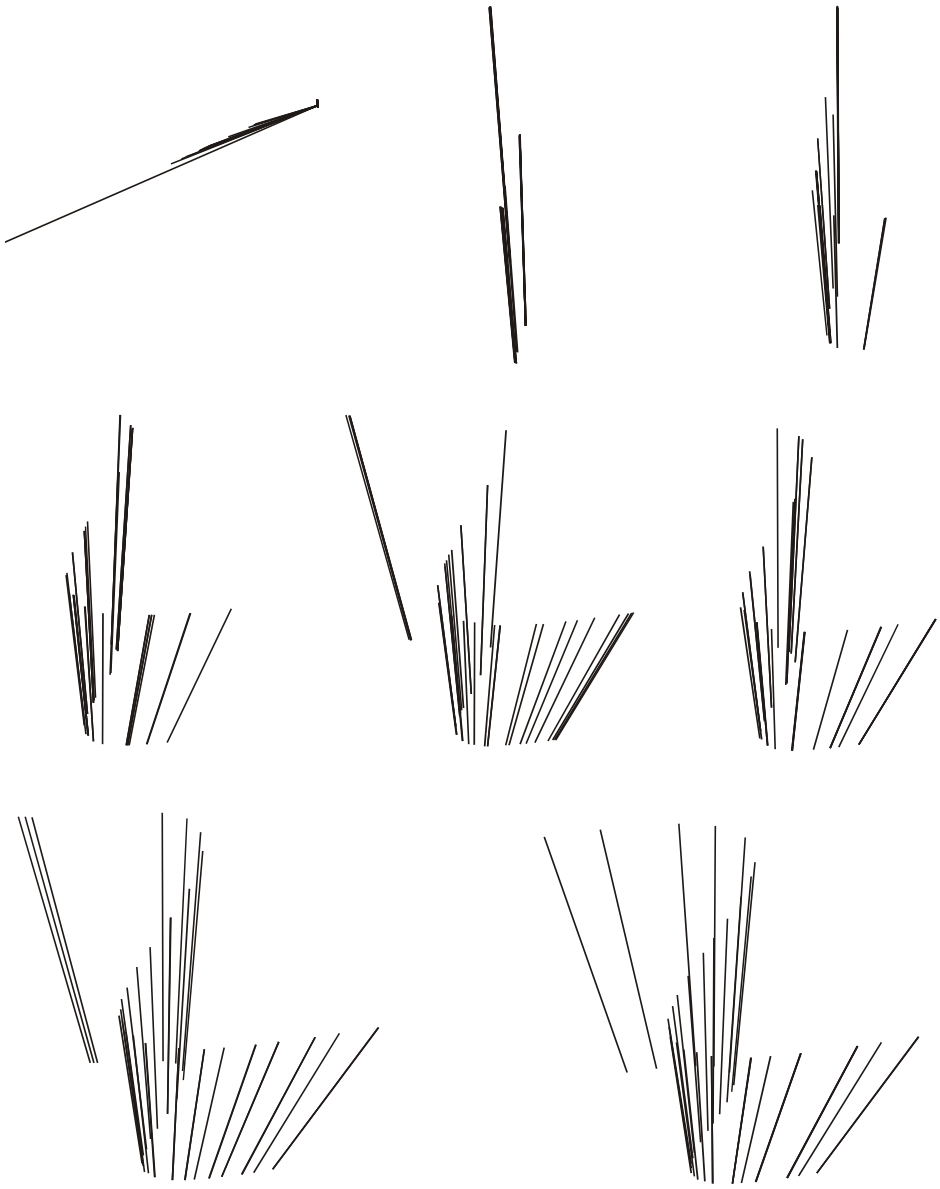
$$(\mathbf{v}, \mathbf{w}) = vx \cdot wx + vy \cdot wy + vz \cdot wz.$$

Ha ϕ tompaszög, akkor $\cos\phi$ negatív. Tehát a probléma megoldásához nem kell meghatározni a ϕ konkrét értékét, hanem elegendő az utóbbi összefüggést előjelét figyelni.

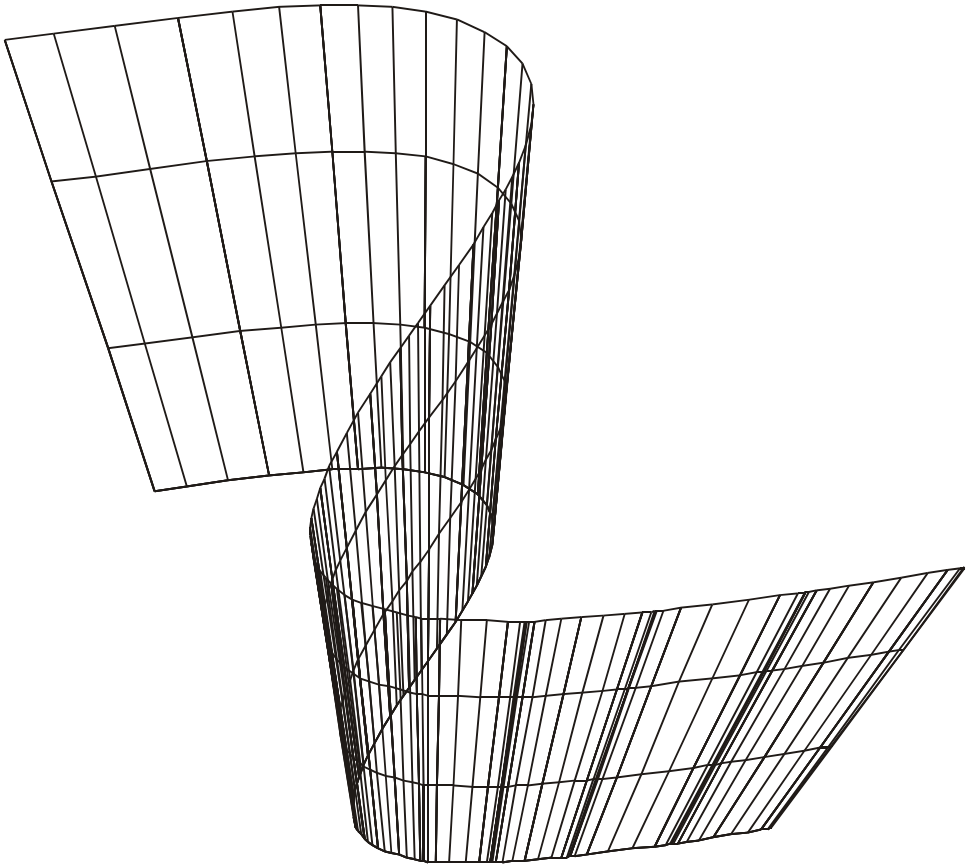
A 17. ábrán láthatjuk az algoritmus által különböző működési fázisban előállított output egyeneseket. Kezdetben néhány tíz iteráció utáni eredményt látunk, később néhány száz, végül több ezer iterációs lépés utáni outputot mutatja az ábra. A 18. ábrán a végeredményként kapott rendezett egyenes halmazra illesztett B-spline vonalfelületet láthatjuk. Úgy teszteltük az elméletet, hogy az algoritmus inputja egy szinuszosan hullámzó vonalfelület egyenesei voltak, amelyeket véletlenszerűen adtunk meg az algoritmus számára, hiszen feltételeztük, hogy nem ismert az egyenesek előzetes sorrendje. A 19. ábrán egyköpenyű hiperboloid alkotói voltak az inputegyenesek. Az ábrán összekötöttük az output egyenesek végpontjait, ezzel mutatva a kialakuló sorrendiséget.

5.3.5. Megjegyzések

Nyilvánvalóan végtelen sok vonalfelület illeszkedhet megadott térbeli egyenesek halmazára. Az is valószínű, hogy a szakirodalomban szereplő eltérő módszerek különböző felületeket eredményeznek. Összevetve az általunk kifejlesztett módszert más algoritmusokkal a következő előnyöket tudjuk megfogalmazni. Figyelembe véve a Kohonen-háló öntanuló tulajdonságát, a háló



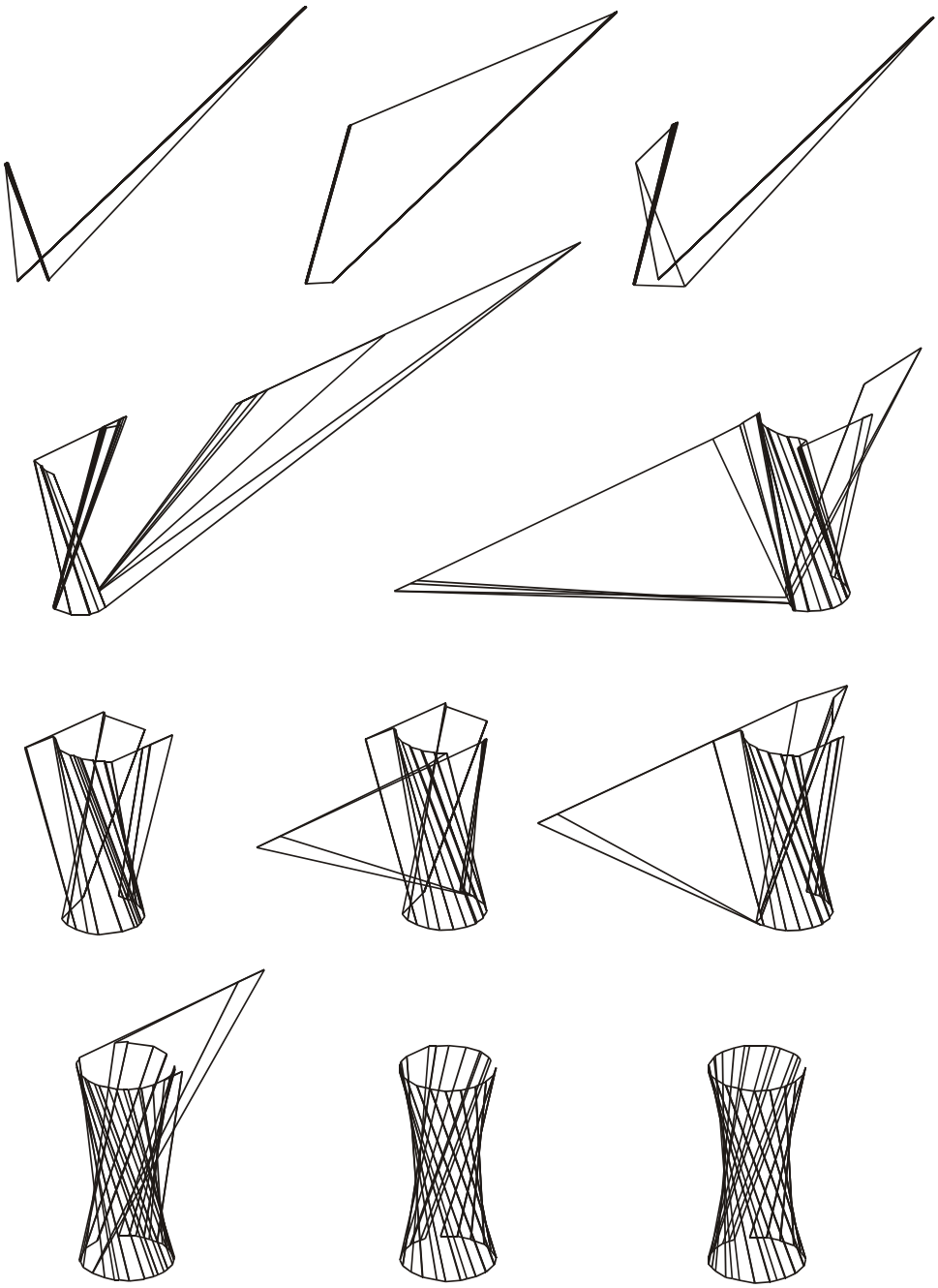
17. ábra. Munkában az algoritmus



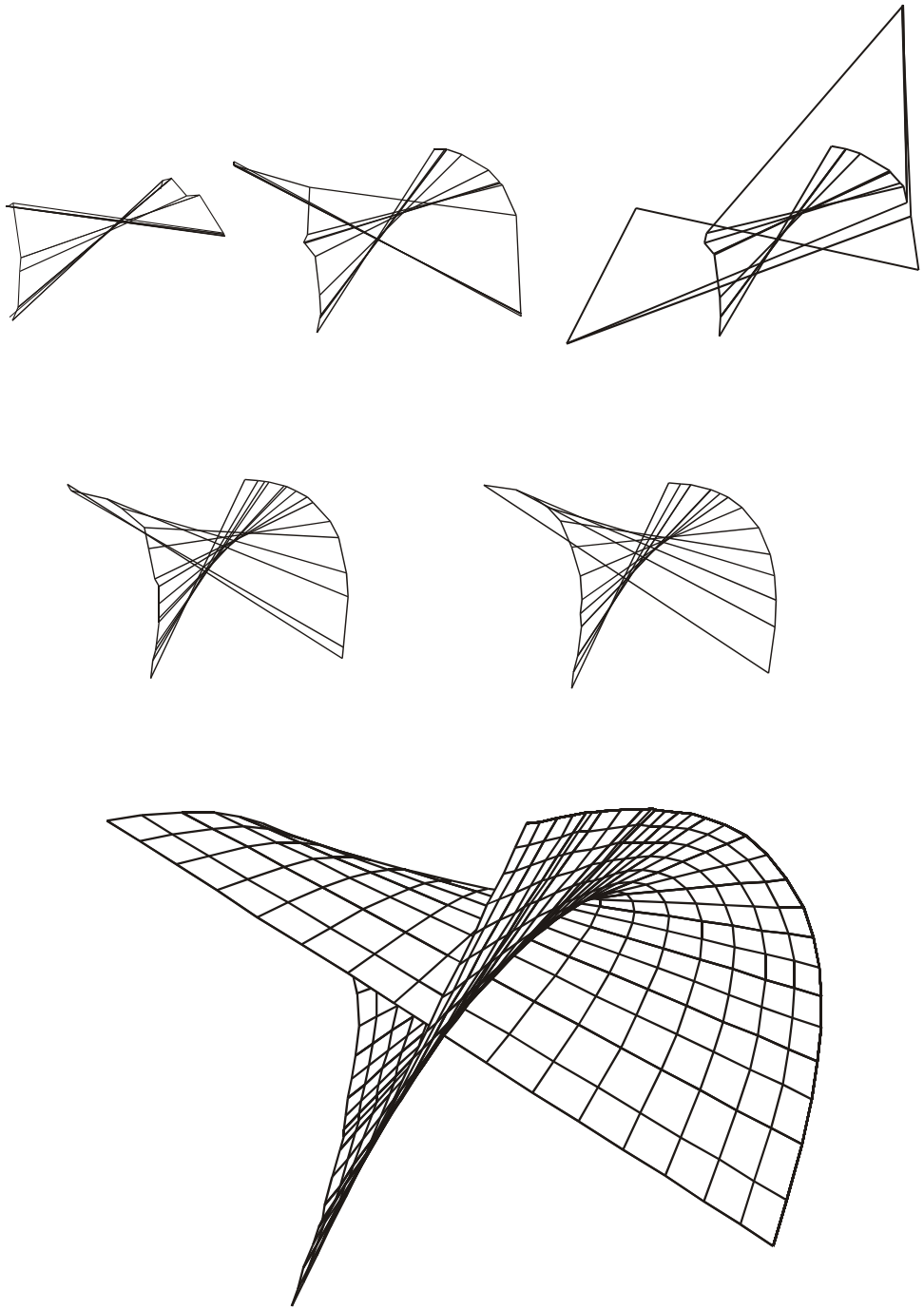
18. ábra. Az output egyenesekre illesztett B-spline felület

igyekszik az adatok nyújtotta lehetőségen belül legkisebb felületi energiával rendelkező alakot felvenni, azaz megpróbálja minél szűkebben approximálni az adatokat a legkisebb felszíni alak felé tarva. Ezen tulajdonságot szokás minimal energy propety (lásd [54]) néven említeni. Ezért a végeredményként kapott rácshálóról, kontrollhálóról úgy beszélünk mint a "legsimább" hálózatról. A háló tartalmazza az összes előre megadott alkotót, ezzel szemben más módszerek nem garantálják ezen tulajdonságot. A fent ismertetett módszer alkalmazható kifejthető felületek esetén is, amelyet a következő fejezetben ismertetünk.

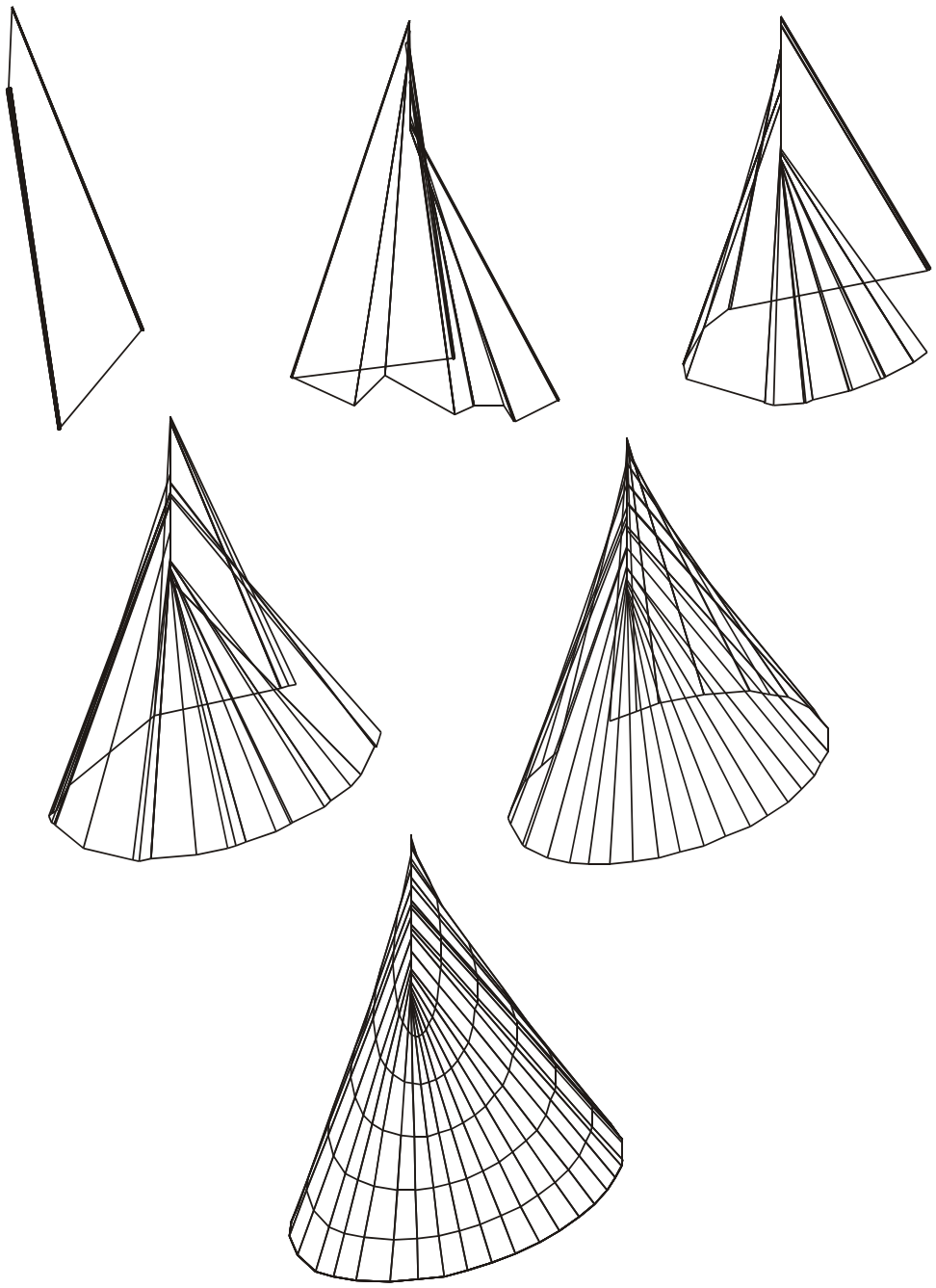
A 20. ábrán Whitney esernyőjét (Whitney's Umbrella) láthatjuk, amelyet Hassler Whitneyről, a differenciáltopológia egyik megalapítójáról neveztek el. A felület implicit egyenlete: $x^2 - y^2 z = 0$. A felületet megkaphatjuk úgy, hogy egy téglalapot önmagával metszve hajtunk össze. Szintén a Whitney felületet jutunk akkor is, ha egy vízszintes metsző egyenespárt a metszéspontjukra állított függőleges tengely mentén mozgatunk, a mozgatás alatt csökkentjük az egyenesek közbezárt szögét, egészen addig, amíg a két egyenes össze nem esik egy egyenesbe. Az ábra alján takartvonalas ábrázolással láthatjuk az output egyenessereg közelítését B-spline felülettel. A 21. és a 22. ábrákon konoidokat láthatunk, amelyek esetében szintén sikerrel használhattuk az algoritmust. Meg kell jegyeznünk, hogy a közölt ábrák tesztadatok futási eredményei. Nincs értelme teljesen véletlenszerűen elhelyezkedő egyenesek esetével foglalkozni, mivel ilyen szituáció előfordulása a gyakorlatban elenyésző.



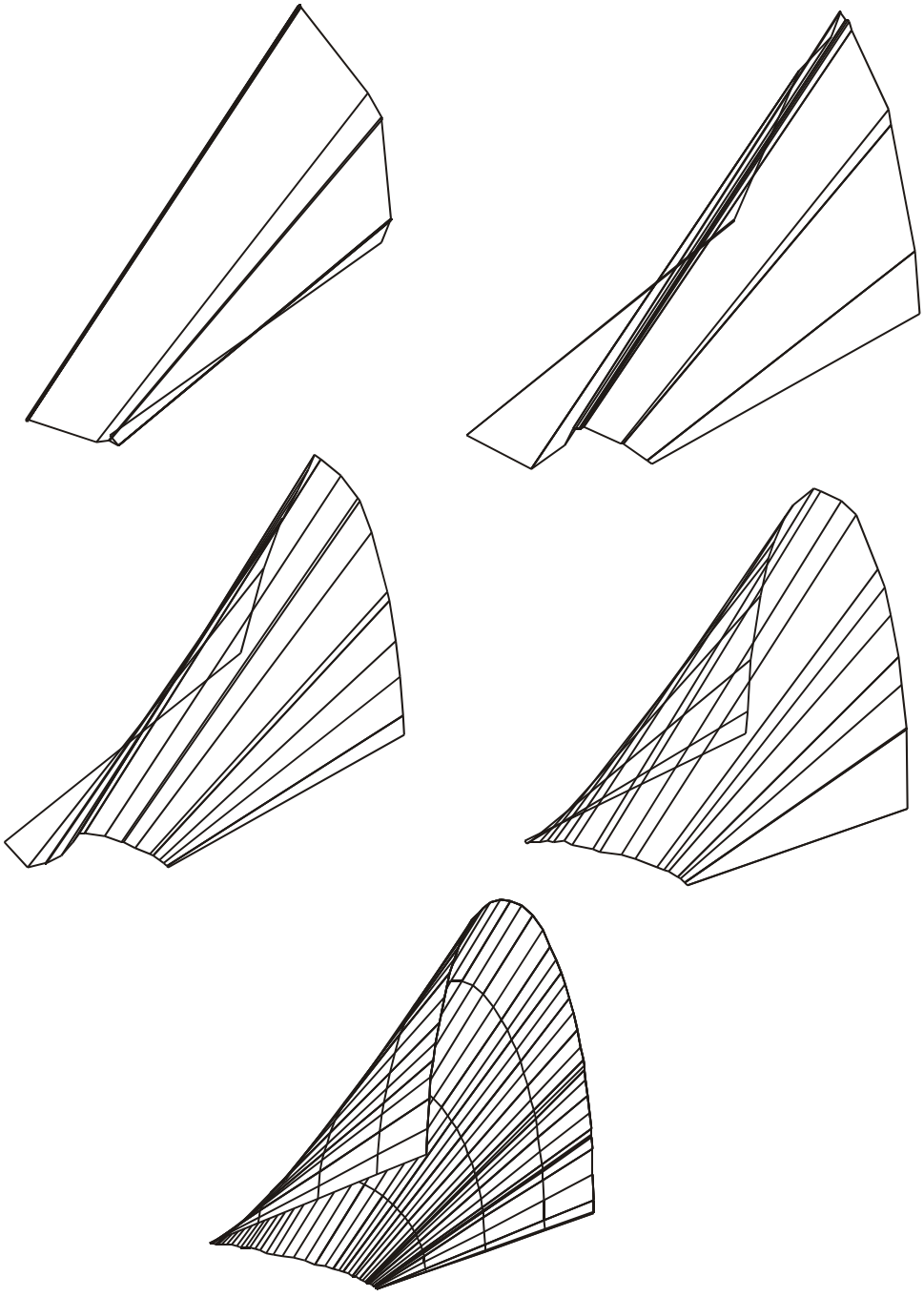
19. ábra. Egyköpenyű hiperboloid előállítása Kohonen-hálóval



20. ábra. Whitney's Umbrella



21. ábra. Ellipszis alapú konoid előállítási fázisai



22. ábra. Konoid előállítása

5.4. Kifejthető felületek konstruálása Kohonen-háló segítségével

A következökben ismertetjük azokat az eredményeket, amelyek megtalálhatóak a [39] cikkben, amelyet Hoffmann Miklóssal készítettem. A vonalfelületek és speciális részhalmazuk, a kifejthető felületek, jól ismert területek a klasszikus geometriában. Korábban említettük már, hogy erről a területről számos komputergeometriai és CAD-CAM alkalmazást ismerünk. Az utóbbi területen szokásos eljárás felületek megadására a különböző típusú spline felületek, mint például a Bézier, B-spline vagy a NURBS felület alkalmazása. Ebből következik, hogy igen hasznos a vonal és kifejthető felületek megadása, illetve konstruálása spline felületek segítségével. Néhány korábbi módszer speciális kezdőfeltételt ad meg [2], [57], de a legtöbb újabban használt módszer projektív geometriai megfogalmazású (lásd [71], [68], [8], [47], [48]). Ebben a munkában a B-spline görbét mint a kifejthető felület duálisát használjuk, ahol a dualitás elve jólismert klasszikus projektív geometriai konstrukció.

Pontfelhők (scattered data) approximációjaként előállított általános B-spline felületek neurális hálókra épülő módszerét már bevezettük a [43], [86] és az [41] cikkekben és a 5.1.1. pontban. Ezekben a már ismertetett Kohonen-háló segítségével tudjuk kezelni és feldolgozni az input adatokat, majd utána standard approximációs módszereket használva kapjuk meg a kívánt approximáló görbét, vagy felületet. Kezdetben a neurális háló az input adatok alapján négyszöghálót állít elő a felületek esetében, s poligont a görbék esetében. Az input adatok, mint csúcspontok adják a kontrollháló vagy a kontrollpoligon pontjait. A célunk az, hogy ezzel a módszerrel a fent említett dualitás elvét kihasználva kifejthető felületet konstruáljunk.

A vizsgálatok teljessége kedvéért ismertetünk néhány hasznos definíciót.

4. Definíció. *Egy felületet* $\mathbb{E}^3$ *-ban vonalfelületnek nevezünk, ha a felület minden pontjára illeszkedik olyan egyenes, amelynek minden pontja a felülethez tartozik. Az ilyen egyeneseket a felület alkotóinak nevezzük.*

A vonalfelületek egyparaméteres egyenessereggel, alkotókkal burkolt felületet jelentenek. Ezen alkotók rendelkeznek azzal a tulajdonsággal, hogy az alkotó minden pontjában a felület érintősíkjára illeszkednek. Ez a tulaj-

donság fontos a vonalfelületek egy speciális részhalmaza, a kifejthető felületek esetében. A kifejthető felületek definíciója a következő:

5. Definíció. *Egy felületet $\mathbb{E}^3$ -ban kifejthető felületnek nevezünk, ha a felület izometrikusan leképezhető a síkra.*

Két felület pontjai között létesített egy-egyértelmű leképezést akkor mondunk izometrikus leképezésnek, ha a megfelelő pontokban az első főmennyiségek egyenlők. Az egymásból izometrikus leképezéssel előállítható felületek metrikus tulajdonságai azonosak: bármely görbeív ugyanolyan hosszú, mint képe; nem változik a felületi görbék szöge, a felületdarabok felszíne stb.

Figyelembe véve a vonalfelületek fentebb említett tulajdonságát és az utóbbi definíciót, megállapíthatjuk a következő jól ismert eredményt:

2. Tétel. *A vonalfelület akkor és csak akkor kifejthető, ha a felület bármely érintősíkja a felületet egy teljes alkotó mentén érinti.*

Szemléletesen egy felület akkor kifejthető[3], ha nem rugalmas anyagból készítve és megfelelően felmetszve, síkba simán kiteríthető. A dolgozatban nem foglalkozunk nem síkba fejthető felületekkel.

Ha az érintősík változik az alkotó mentén (azaz pontról pontra más és más), akkor a felületet általános (nem kifejthető) vonalfelületnek nevezzük, Az ilyen felületre szokásos a torzfelület elnevezés használata is. Ilyen nem kifejthető vonalfelület például az egyköpenyű hiperboloid, vagy a hiperbolikus paraboloid (nyeregfelület).

Ez azt jelenti, hogy ha egy vonalfelület érintőfelületét vesszük figyelembe, akkor a kifejthető felület nem más, mint egyparaméteres síksereg burkolója (envelope). Emellett általában véve a vonalfelületeket az érintősíkok seregének van egy második paramétere is az alkotók mentén.

5.4.1. Dualitás elve a projektív geometriában

Jól ismert fogalom a projektív geometriában a dualitás elve. Coxeter [20] után a következőt mondhatjuk, a síkbeli dualitás elve azt állítja, hogy minden meghatározás érvényes marad, és minden tétel igaz marad, ha a *pont*

[3] az irodalomban a lefejthető elnevezés is szokásos

és *egyenes* szavakat felcseréljük (és ebből adódóan bizonyos szópárokat is, mint például az *összeköt* és *metsződik*, *kollineáris* és *konkurrens*, *csúcspont* és *oldal*, és így tovább). Így a projektív sík összes axiómája maga után vonja az axiómák duálisait is. E megjegyzés elegendő is ahhoz, hogy kimondjuk a kétdimenziós dualitási elv érvényességét. Mivel az axiómákat és azok következményeit alkalmazva egy adott tétel bizonyítására, azonnal kimondhatjuk a duális tételt is, hiszen a duális tételt szinte mechanikusan beláthatjuk az eredeti tétel bizonyításában szereplő lépések dualizálásával (lásd pl. Pappos tételt és duális párját, a Brianchon tételt). Egy egyenesre illeszkedő pontokat *kollineáris* pontoknak nevezzük. Egy pontra illeszkedő egyeneseket *konkurrens* egyeneseknek nevezzük. Általában a duális állítás különbözik az eredetitől, ha nem akkor *önduálisnak* nevezzük (például önduális geometriai objektum az egyenes egy ráilleszkedő ponttal, tehát síkban a sugársor, térben a sugárnyaláb) Geometriai objektumok duálisát meghatározhatjuk a eredeti definíció dualizálásával.

Pottmann [70] után megadjuk a dualitás definícióit $n-$dimenziós projektív tér esetén.

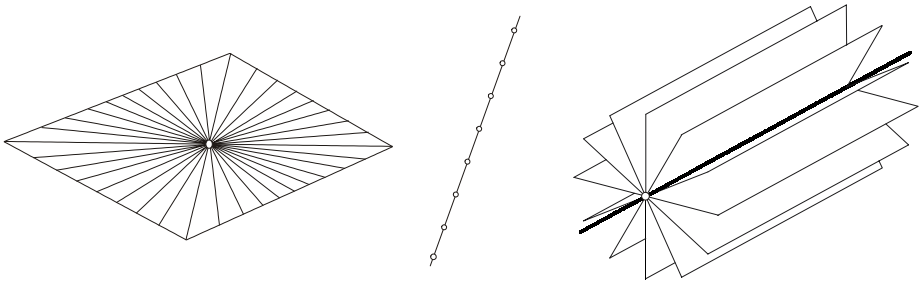
6. Definíció. *Ha egy állítás, amely tartalmaz $k-$dimenziós altereket, projektív értelemben egyértelmű meghatározást, metszést és P^n altereinek tartalmazását (beágyazását), úgy módosítunk, hogy kicseréljük ezen elemeket $(n-k-1)$ -dimenziós alterekkel, metszéssel, projektív értelemben egyértelmű meghatározással és alterek ellentétes tartalmazásaival (beágyazásaival), akkor az így kapott új állítást az eredeti duálisának mondjuk.*

Ezek után könnyen kimondhatjuk a háromdimenzióbeli dualitást elvét, amelyben pontok helyett síkok, egyenesek helyet egyenesek, síkok helyett pontok szerepelnek a definíciókban és tételekben.

Nyilvánvaló, hogy a duális állítás dualizálásával megkapjuk az eredeti állítást. Az is igaz, hogy a P^n projektív tér nem önduális, mert végtelen sok ideális pontja van, de csak egy ideális síkja.

5.4.2. Duális NURBS görbe

A dolgozat szempontjából a háromdimenziós $\mathbb{P}^3$ projektív térben alkalmazzuk a dulitás elvét. Ebben a térben a pontnak négy homogén koordinátája van (x_0, x_1, x_2, x_3), a sík pedíg $v_0x_0 + v_1x_1 + v_2x_2 + v_3x_3 = 0$ egyenlettel



23. ábra. Sugársor a síkban ,amely önduális, pontsor és síksor amelyek egymásnak duálisai

reprezentálható, azaz a (v_0, v_1, v_2, v_3) rendezett számnégyes tekinthető úgymond a sík koordinátáinak, ennélfogva $\mathbb{P}^3$ -ban analitikusan azonos módon reprezentáljuk a síkot és a pontot. Ebben az esetben a pontot és a síkot egymás *duálisainak* tekinthetjük. (lásd a (5.4.1) pontot)

A dualitás elve alkalmazható a kifejthető NURBS felületetek esetében is, amennyiben a következő definíció szerint, térbeli NURBS görbék duálisainak tekintjük őket.

7. Definíció. *A NURBS görbe megadható a következő homogén alakban*

$$\mathbf{s}(t) = \sum_{i=0}^{n} N_i^k(t)\mathbf{p}_i,$$

ahol az $N_i^k(t)$ k-adrendű normalizált B-spline alapfüggvény. A csomóvektor $t_0 = t_1 = ... = t_{k-1} < t_k < ... < t_n < t_n = ... = t_{n+k}$, ,a $\mathbf{p}_i(w_i, w_i x_i, w_i y_i, w_i z_i)$ a görbe kontrollpontjai.

Ezen görbe duál formája

$$\mathbf{U}(t) = \sum_{i=0}^{n} N_i^k(t)\mathbf{U}_i,$$

$\mathbf{U}_i$ $(w_i, w_i x_i, w_i y_i, w_i z_i)$ a $\mathbf{p}_i$-hez duális síkokat jelöli.

Ezen duál alak egyparaméteres síksereg burkolójaként értelmezhető, így ez lesz a kifejhető NURBS felület. Könnyen átalakítható ez a duál alak, kényelmesebb, tenzorszorzatos alakká (lásd [68]), de számunkra az előző forma

megfelelőbb. A fenti duál alak részletesebb leírása és további tulajdonságai megtalálhatók a [71, 68, 8] dolgozatokban.

A definíció azt eredményezi, hogy ha adott egy síksereg, akkor a duális térben végrehajtott ponthalmaz görbe approximációjával találhatunk kifejthető NURBS felületet, mint ezen síksereg burkolóját A duális térben a síkokat pontokkal helyettesítjük, mindemellett az általunk alkalmazni kívánt technikához szükségünk van síkokon értelmezhető távolság fogalomra is. Erre azért van szükségünk, mert a neurális hálózat tanulási folyamatának egy előfeldolgozó lépése a megadott és az általa előállított output adatok közötti eltérés, távolság mérése.

5.4.3. Távolságfüggvény a síkok terében

Az elméleti hátteret már korábban tárgyaltuk. Hivatkoztunk Pottmann és társainak mostanában megjelent munkáira (lásd [71]). A síkok Euklidészi távolságának meghatározása két sík szögén alapszik, amely számunkra nem megfelelő, mert nekünk csak a vizsgált tartomány felett kell meghatároznunk két sík távolságát, különbségét, amely tetszőlegesen nagy lehet, annak ellenére, hogy a két sík szöge valójában tetszőlegesen kicsi.

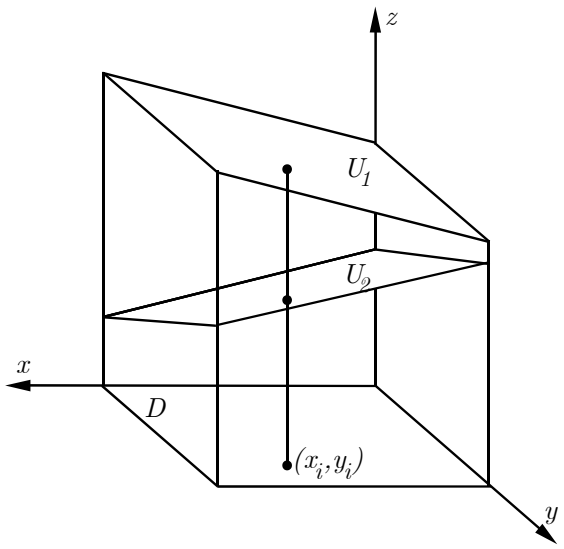
A P^* síkok duális tere izomorf $\mathbb{P}^3$-mal. Ahhoz hogy bevezessük az euklideszi metrikát szükségünk van egy legfeljebb affin térre, ezért el kell vennünk egy síkot $\mathbb{P}^3$-ból, és egy hipersíkot a P^* duális térből. Ez a hipersík azon síkok csoportja, amelyek illeszkednek egy végtelen távoli pontra, például a z tengely végtelen távoli pontjára. Ezután, ha csak olyan síkokkal foglalkozunk, amelyek nem párhuzamosak a z tengellyel, akkor ezen síkok tere egy A^* affin tér lesz. Síkok analitikus reprezentációja a

$$z = u_0 + u_1 x + u_2 y,$$

összefüggéssel adott, ahol az $(u_0, u_1, u_2, -1)$ homogén koordináták, amelyek úgy tekinthetőek, mint az A^*-beli sík (u_0, u_1, u_2) affin koordinátái. Ebben a térben a síkok nem párhuzamosak a z tengellyel, ennélfogva a vizsgált tartománynak választhatunk egy D tartományt az xy síkban. Most már definiálhatjuk két sík speciális távolságát ezen D tartomány fölött.

8. Definíció. *Ha* $\mathbf{U}_1\,(u_{0,1}, u_{1,1}, u_{2,1})$ *és* $\mathbf{U}_2\,(u_{0,2}, u_{1,2}, u_{2,2})$ *két sík* A^**-ban, akkor a távolságuk* D *tartomány fölött a következő*

$$d_\mu(\mathbf{U}_1, \mathbf{U}_2) = \left\|(u_{0,1} - u_{0,2}) + (u_{1,1} - u_{1,2})x + (u_{2,1} - u_{2,2})y\right\|_{L^2(\mu)},$$



24. ábra. Két sík távolságának a meghatározása

azaz, $L^2(\mu)$ azon lineáris függvények távolsága, amelyek gráfja $\mathbf{U}_1$-ban és $\mathbf{U}_2$-ban van, ahol μ pozitív mérték $\mathbb{R}^2$-ben. (A lineáris függvények létezését mindig feltételezzük $L^2(\mu)$-ben).

A μ mérték többféle módon is definiálható (lásd [71]). Számunkra megfelelő forma, amikor μ egyenlő az (x_i, y_i).-pontokban tömörülő számos pont összegével

$$d_\mu(\mathbf{U}_1, \mathbf{U}_2)^2 = \sum_i ((u_{0,1} - u_{0,2}) + (u_{1,1} - u_{1,2})x_i + (u_{2,1} - u_{2,2})y_i)^2. \quad (23)$$

5.4.4. Approximáció Kohonen-hálóval

A Kohonen-háló részletes ismertetése megtalálható a 4.1. és 5.1.1 pontokban. Jelen esetben mellékes, hogy a dinamikus vagy a standard Kohonen-hálót alkalmazzuk. A jobbérthetőség kedvéért ismételjük meg az algoritmust, majd rámutatunk a modosításokra.

A megadott ponthalmaz pontjait jelöljük a következőképpen:

$$\mathbf{p}_i(x_{1,i}, x_{2,i}, x_{3,i}), \quad (i = 1, ..., m),$$

A Kohonen-háló input rétege három neuront tartalmaz, az output réteg $n(> m)$ neuront tartalmaz. Az output réteg minden neuronja (pontja) összeköttetésben áll az input réteg összes neuronjával (pontja) (lásd 7. ábrát), azaz, az input réteg i-edik neuronja hozzá van kapcsolva az output réteg j-edik neuronjához, és w_{ij} súly van hozzárendelve minden egyes összeköttetéshez. Ezeket a súlyokat tekintjük az output pontok koordinátáinak,

$$\mathbf{q}_j(w_{1j}, w_{2j}, w_{3j}), \quad (j = 1, ..., n),$$

amelyek szolgáltatják majd az eredmény poligont a tanulási folyamat után:

1. Inicializáljuk a $w_{sj}, (s = 1, 2, 3, \quad j = 1, ..., n)$ súlyokat, mint kicsi véletlen számokat az input koordináták átlagainak környezetében. Legyen a tréning idő $t = 1$.

2. Adjunk új input $(x_{1,i_0}, x_{2,i_0}, x_{3,i_0})$ értékeket, amelyek egy véletlen módon kiválasztott $\mathbf{p}_{i_0}$pont koordinátái.

3. Határozzuk meg minden output csomóérték $d_j(\mathbf{p}_{i_0}, \mathbf{q}_j), \quad (j = 1...n)$ távolságát az input ponttól.

4. Találjuk meg a $\mathbf{q}_{j_0}$ nyerő értéket, amelynek minimális a távolsága az input ponttól, azaz. $d_{j_0} = min(d_j)$.

5. Határozzuk meg a $N(t) = (j_0, j_1, ..., j_k)$ környezetet.

6. Az $N(t)$ környezetben frissítsük a csomópontok súlyait (azaz a koordinátákat) a következő képlet alapján:

$$w_{sj}(t + 1) = w_{sj}(t) + \eta(t)(x_{s,i_0} - w_{sj}(t)), \quad \forall j \in N(t),$$

   ahol $\eta(t)$ az úgynevezett nyerő tag (gain term), lásd a 5.1.2. pontot.

7. Növeljük a tréning (tanulási) időt, $t = t + 1$. Ismételjük a 2-től 7-ig a lépéseket amíg a hálózatot tanultnak (kiképzettnek) tekintjük.

Most megadjuk a fenti általános algoritmus azon modosításait, amely alkalmassá teszi síkok halmazát burkolójával, azaz kifejthető NURBS felülettel történő approximációjára. Ebben az esetben, tekintsük az input adatokat az $\mathbf{U}_i(u_{0,i}, u_{1,i}, u_{2,i})$ síkok koordinátáinak. Azonban a leglényegesebb különbség a 3. lépésben lesz, ahol az eredetileg a $d_j(\mathbf{p}_{i_0}, \mathbf{q}_j)$ távolságot

használjuk. Az általános algoritmusban és alkalmazásaiban ez a távolság a szokásos Euklideszi távolság (lásd [43]), de a jelenlegi szituációban, mivel a duális P^* térben vagyunk, a speciális d_μ távolságot kell használnunk,

$$d_\mu(\mathbf{U}_1, \mathbf{U}_2) = \|(u_{0,1} - u_{0,2}) + (u_{1,1} - u_{1,2})x + (u_{2,1} - u_{2,2})y\|_{L^2(\mu)},$$

amelyet az előző részben definiáltunk (lásd (23) egyenletet). Ezekkel a változtatásokkal az algoritmus automatikusan produkálni fogja a duál NURBS görbe „kontrollpoligonját", amelyet át tudunk alakítani normál tenzorszorzattal megadott felületté. Tehát a síkok kezdeti halmazát (seregét) standard kifejthető NURBS felülettel tudtuk modellezni.

5.4.5. Következtetés

A dolgozat ezen részében egy olyan speciális módszert ismertettünk, amely kifejthető NURBS felületeket konstruál előre megadott érintősíkok halmazára. A módszer olyan speciális geometriai elvet használ, mint a projektív geometriában használatos dualitás, azaz a problémát a P^* duális térben lévő görbe modellezési problémára transzformáljuk. A Kohonen-féle neurális háló sorrendet definiál az adathalmazban, azonban ezen technika használatához alkalmaznunk kellett egy speciális távolság értelmezést. Meg kell említenünk, hogy a fő előnye a módszernek az, hogy az előállított felület az adott lehetőségen belül legkisebb felületi energiával rendelkező alakot fogja felvenni, azaz megpróbálja minél szűkebben approximálni az adatokat a legkisebb felszíni alak felé tarva, amely a legtöbb esetben fölöslegessé teszi egyéb javító technikák alkalmazását. Korábban már említettük, hogy további előnye a módszernek az, hogy egyaránt alkalmazható meglehetősen kevés input adat és eloszlással megadott nagyszámu elvileg végtelen sok input adat esetében is.

6. Összefoglalás

A dolgozatban rendszertelen adatokra történő felületillesztéssel foglalkoztunk. A Kohonen-hálózat alkalmas arra, hogy rendszertelen adatok esetén, rendezett adatokat, pontsorozatot illetve négyszög topológiájú pontrácsot állítsunk vele elő. Ezután a Bézier- vagy NURBS-felülethez hasonló standard eljárásokat tudunk alkalmazni felület interpoláció vagy approximáció céljából. Célunk az volt, hogy javítsunk a Kohonen-háló alkalmazásának módszerén illetve kiterjesszük alkalmazhatóságát vonalfelületekre.

A dolgozat első részében rövid leírást adtunk a B-spline felület approximációról és interpolációról programozási és komputergrafikai szempontból. A következő részben kritikai összegzését adtuk a szakirodalomnak, amelyekben hasonló problémákra keresnek megoldást a szerzők. Ezután ismertettük a Kohonen neurális háló alkalmazásának azon változatát, amelyet módszerként használtunk a dolgozatban.

A Kohonen-háló, amelyet T. Kohonen fejlesztett ki, erős öntanuló képességgel rendelkezik. Ennek az az előnye, hogy a Kohonen-háló végig megtartja topológiáját (jelen esetben a négyszöghálót) a tanulási folyamat során, és a térbeli ponthalmaz felé mozog követve a pontok elhelyezkedését. Miután a Kohonen-háló megtanulta a pontfelhőt, előállít egy négyszöghálót. Az outputként kapott négyszöháló olyan kontrollháló szerepét töltheti be, amely alapja lehet valamely standard approximációs vagy interpolációs módszernek A megfelelő eredmény elérése érdekében jól meg kell becsülni a kontrollháló kezdeti csúcspontjainak, azaz az output neuronok számát. Hogy elkerüljük ezt a problémát használhatjuk a dolgozat egyik eredményét, a dinamikus Kohonen-hálót, amely lehetővé teszi az output neuronok számának dinamikus változtatását a tanulási folyamat alatt. Ebben a dolgozatban szemben az eredeti módszerrel dinamikus mesterséges neurális hálózatot alkalmaztunk. A rendszertelen adatokra történő felületillesztés esetében meghatározó kezdeti folyamat, a rendezett adatok előállítása Kohonen-háló segítségével. Ezért tartottuk fontosnak, hogy a dinamikus változat használatával gyorsítsuk, és megbízhatóbbá tegyük az algoritmust.

A dolgozat következő részében a vonalfelületekre terjesztettük ki az ismertetett módszer használatát. A felületek megadására általános módszer a különböző spline-felületek használata, ezért nagyon hasznos, ha a vonalfelületeket és a kifejthető felületeket is előtudjuk állítani például Bézier, B-spline

vagy NURB felületként. Az ismert módszerek esetében a fő cél a négyszög kontrollháló előállítása. A dolgozatban az eredeti input adatok előre megadott rendezetlen egyenesek halmaza, amelyeket alkotóknak nevezünk. Ebben az esetben az alkotók megadására homogén koordinátákat használtuk, amely jól ismert technika a projektív geometriában. Szintén alkalmaztuk az egyenesek megadására a Plücker-koordinátákat is. Ilyen módon egy kölcsönösen egyértelmű megfeleltetést hoztunk létre a valós számok hatosa és a $\mathbb{P}^3$ egyenesei között, amely egy másik kölcsönösen egyértelmű megfeleltetést eredményez a $\mathbb{P}^3$ egyenesei és a $\mathbb{P}^5$ ötdimenziós projektív tér pontjai között. Az utóbbi leképezés segítségével tudjuk beágyazni az adott egyeneseket a későbbi felület kontrollhálójába. Ezzel a módszerrel a Kohonen-hálót a $\mathbb{P}^5$ tér pontjaival tanítjuk, azaz az input réteg hat neuront fog tartalmazni, mialatt az output réteg $\mathbb{P}^5$-ben poligon lesz. Amikor a tanulási folyamat befejeződött, az eredményül kapott poligon áthalad a megadott $\mathbb{P}^5$-beli pontokon. Visszatranszformálva $\mathbb{P}^3$-ba megkapjuk az adott egyeneseket és a négyszöghálót, amely tartalmazza a kiindulásként megadott egyeneseket.

A dolgozat utolsó részében egy olyan speciális módszert ismertettünk, amely kifejthető NURBS felületeket konstruál előre megadott érintősíkok halmazára. A módszer olyan speciális geometriai elvet használ, mint a projektív geometriában használatos dualitás, azaz a problémát a duális térben lévő görbe modellezési problémára transzformáljuk. A Kohonen-féle neurális háló sorrendet definiál az adathalmazban, azonban ezen technika használatához alkalmaznunk kellett egy speciális távolság értelmezést. Nagy előnye a módszernek az, hogy a végeredményként előállított felület az adott lehetőségen belül legkisebb felületi energiával rendelkező alakot fogja felvenni (minimal energy propety), azaz megpróbálja minél szűkebben approximálni az adatokat a legkisebb felszíni alak felé tartva, amely a legtöbb esetben fölöslegessé teszi egyéb javító technikák alkalmazását. További előny, hogy a módszer egyaránt alkalmazható meglehetősen kevés input adat és eloszlással megadott nagyszámú, elvileg végtelen sok input adat esetében is.

Számos programot készítettünk az ismeretett módszerek gyakorlati tesztelésére. A programok lehetőséget adtak egyrészt az eredmények vizsgálatára, másrészt a dolgozatban szereplő ábrák is ezen programok futási eredménye.

7. Summary

The purpose of this dissertation is to improve and extend the method of modeling scattered data by free-form surfaces. In that method an artificial neural network, the Kohonen-network was used for order the data and form a quadrilateral control grid from the scattered points, hence the standard free-form methods, like Bézier-surface or NURBS, could be applied to approximate or interpolate the data.

In the first part we give a short description of B-spline surface approximation and interpolation from the aspect of computer graphics and programming. In the following part we give a critical summary of the literature. We then introduce the Kohonen neural network in order to overcome the problems described.

The main advantage of this method is the preprocessing stage which produces a topologically quadrilateral grid from the scattered data. This means that the basic free-form methods can be applied without any kind of modification. For this preprocessing stage, an artificial neural network has been used. The Kohonen-network, developed by T. Kohonen, has a self-organizing capability which allows a predefined grid to keep its topology during the training procedure. When this grid moves towards the input of scattered points it manages to follow their spatial structure. This grid amounts to the input control grid for the arbitrarily chosen free-form method used to construct the desired surface. For this process, however, an accurate estimation of the number of vertices of the grid (i.e. the number of output neurons of the network) is required. To avoid this problem we use the recent developments of the dynamic Kohonen network allowing us to change the number of output neurons dynamically during the self-organizing process. This capability yields fewer neurons thus providing not only a quicker process but a more reliable method.

The next part of the dissertation contains a further application of the previous method for ruled surfaces. As the standard way of describing surfaces is through different types of spline-surfaces, like the Bézier, B-spline or NURB surface it is highly desirable to describe and construct ruled and developable surfaces as a spline surface. The main purpose of most of the known methods is to construct a quadrilateral control grid. In our method the Kohonen neural network has been applied in a preprocessing stage to

produce a control grid from the original input data. In the dissertation the original input data structure consists of a set of given lines, called rulings. A well-known technique in projective geometry, the description of these lines by homogeneous coordinates will be advantageous. The Plücker coordinates of the projective lines will be also applied.

The 1-1 correspondence between the homogeneous 6-tupels of real numbers and the lines of P^3 yields another 1-1 correspondence between the lines of P^3 and the points of the five dimensional projective space P^5. With the help of this latter mapping we can embed the given rulings and the lines of the future grid of the surface. In this way the Kohonen network will be trained by points from P^5, that is the input layer will contain 6 neurons, while the output layer will be a polygon in P^5. When the neural network is trained, the resulting polygon passes through the given points of P^5. Transforming this situation back to P^3 the given lines and a quadrilateral grid will be received, which grid contains the given lines and consists of straight lines in that direction.

A special method for constructing developable NURBS surface from a given set of scattered tangent planes has been presented in the final section of this dissertation. The method used a special geometrical principle, namely the duality of projective geometry, to transform the problem to a curve modeling in dual space. While the Kohonen neural network orders the scattered data, a technique using a special distance function is used. We notice that one of the main advantages of this method is that the neural network has a kind of minimal energy property, which yields a fairly smooth control grid and surface without any additional smoothing and fairing techniques. The other advantage is that this method can handle limited input data as well as an infinite amount of data given by a distribution.

Many computer programs have been developed for trying the above methods in practice. These programs are suitable for examining running results and constructing some attractive plotting outputs.

Hivatkozások

[1] Alder, M., Togneri, R., Lai, E., Attikiouzel, Y., Kohonen's algorithm for the numerical parametrisation of manifolds, Pattern Recognition Letters 11, pp. 313-319, 1990.

[2] Aumann, G., Approximate development of skew ruled surfaces. Comput. & Graph. 13 361-366, 1989.

[3] Aumann, G., Interpolation with developable Bézier patches. Computer Aided Geometric. Design. 8 409-420, 1991.

[4] Barhak, J., Fischer, A., Parametrization and reconstruction from 3D scattered points based on neural network and PDE techniques, IEEE Transactions on Visualization and Computer Graphics, Vol. 7, No.1, 2001.

[5] Barsky, B., Computer Graphics and Geometric Modelling Using Beta-splines, Springer-Verlag, Berlin, 1988.

[6] Blum, H., A transformation for extracting new descriptions of shape, IEEE Proceedings of the Symposium on Models for the Speech and Vision Form, Boston, pp. 362-380, 1964.

[7] Bézier, P., Numerical Control, Mathematics and Applications, Wiley 1972.

[8] Bodduluri, R., Ravani, B., Design of developable surfaces using duality between plane and point geometries, Computer-Aided Design, 10 ,pp. 621–632, 1993.

[9] Bodduluri, R., Ravani, B., Geometric Design and Fabrication of Developable Surfaces, ASME Adv. Design Autom. 2, pp. 243-250, 1992.

[10] Boehm, W., Farin, G., and Kahmann, J., A survey of curve and surface methods in CAGD, Computer Aided Geometric Design, 1, pp.1-60, 1984.

[11] de Boor, C., On calculating with B-splines, J. Approx. Theory, 6:50-62, 1972.

[12] Borgulya, I., Neurális hálók és fuzzy-rendszerek, Dialóg Campus Kiadó, Budapest-Pécs, 1998.

[13] Branhill, R. E., Representation and Approximation of Surfaces, in: Rice, J.R. (ed): Mathematical Software III., Academic Press, New Yok, 1977.

[14] Budó, Á., Kísérleti fizika, I kötet, Tankönyvkiadó, Budapest, pp 116-118, 1981.

[15] Castillo, E., Functional Networks, Neural Processing Letters 7, pp. 151-159, 1998.

[16] Chen, H., Pottmann, H., Approximation by Ruled Surfaces, Technical Report, Nr.46, 1997.

[17] Chen, H., Pottmann, H., Approximation by Ruled Surfaces, Journal of Computational and Applied Mathematics, 102, pp. 143-156, 1999.

[18] Coons, S.A., Surface Patches and B-spline Curves, Computer Aided Gemetric Design, Academic Press, 1974.

[19] Cox, M., The numerical evaluation of B-splines, DNAC 4, NAtional Physical Laboratory, 1971.

[20] Coxeter, H. S. M., Projektív geometria, Gondolat Könyvkiadó, Budapest, 1986., Eredeti: Porjective Geometry, Second Edition, University of Toronto Press, Toronto, 1974.

[21] Coxeter, H. S. M., A geometriák alapjai, Műszaki Könyvkiadó Budapest, 1987.

[22] Datta, A., Parui, S.K., Chaudhuri, B.B., Skeletonization by topology-adaptive self-organizing neural network, Pattern Recognition 34, Elsevier Science, pp. 617-629, 2001.

[23] Eck, M., Hadenfeld, J., Local energy fairing of B-spline curves, in: Farin, G., Hagen, H., Noltemeier, H. (Eds.), Computing 10. Springer, Berlin, 1995.

[24] Farin, G., Curves and Surfaces for Computer Aided Geometric Design A Practical Guide, Academic Press, 1996.

[25] Floater, M., Parametrization and smooth approximation of surface triangulations, Computer Aided Gemetric Design, 14, pp. 231-250, 1997.

[26] Foley, T., Hagen, H., Advances in Scattered Data Interpolation, Surv. Math. Ind. Vol.4., pp.71-84., 1994.

[27] Freeman, J., Skapura., D., Neural Networks; Algorithms, Applications and Programming Techniques, Addison-Wesley, 1991.

[28] Frey, W., H., Bindschadler, D., Computer Aided Design of a Class of Developable Bézier Surfaces, General Motors R&D Publication 8057, 1993.

[29] Gordon, W., Riesenfeld, R., B-spline curves and surfaces, In R. E. Barnhill and R. F. Riesenfeld editors, Computer Aided Geometric Design, p. 95-126, Academic Press, 1974.

[30] Greiner, G., Hormann, K., Interpolating and approximating scattered 3D data with hierarchical tensor product B-splines, in: Méhauté, A. L., Rabut, C., Schumaker, L. (Eds.), Surface Fitting and Multiresolution Methods. Vanderbilt University Press, Nashville, pp. 163-172., 1997.

[31] Gu, P., Yan, X., Neural Network Approach to the Reconstruction of Freeform Surfaces for Reverse Engineering, Computer Aided Design, Vol. 27, No.1. pp.59-64, 1995.

[32] Hayes, J., Halliday, J., The least-squares fitting of cubic spline surfaces to general data sets, J. Inst. Math. Appl. 14, 89-103, 1974.

[33] Hajós, Gy., Bevezetés a geometriába, Tankönyvkiadó Budapest, 1984.

[34] Herman, I., The use of projective geometry in computer graphics, LectureNotes in Computer Science, 564, Springer-Verlag, 1991.

[35] Hermann, T., Kovács, Z., Várady, T., Special applications in surface fitting, in: Starsser, W., Klein, R., Rau, R. (Eds.), Geometric Modeling: Therory and Practice. Springer, pp. 14-31, 1997.

[36] Hlavaty, V., Differential line geometry, Nordhoff Ltd, Groningen, 1953.

[37] Hoffmann M., Kovács E., Interpolation possibilities using rational B-spline curve, Acta Acad. Paed. Agriensis, Vol. XXV., pp.103-107., 1998.

[38] Hoffmann M., Kovács E., Constructing ruled B-spline surfaces by Kohonen neutral network, Acta Acad. Paed. Agriensis, TOM. XXVII., Sectio Matematicae, pp. 63-68, 2000.

[39] Hoffmann M., Kovács E., Developable surface modelling by neutral network, Computers & Mathematics with Applications (közlésre elfogadva 2002)

[40] Hoffmann M., Interpolation by Beta-spline, Proceedings of the International Conference of Applied Informatics, pp. 36-44, Eger, 1993.

[41] Hoffmann M., Modified Kohonen Neural Network for Surface Reconstruction, Publ. Math. Debrecen, 54 Suppl. 857-864, 1999.

[42] Hoffmann, M., Várady, L., Free-form curve design by neural networks, Acta. Acad. Paed. Agriensis, TOM. XXIV., pp. 99–104, 1997.

[43] Hoffmann, M., Várady, L., Free-form Surfaces for Scattered Data by Neural Networks, *J*ournal for Geometry and Graphics, Vol.2, No.1, pp. 1–6, 1998.

[44] Hormann, K., Greiner, G., An efficient global parametrization method, in: Laurent, P.-J., Sablonniére, P., Schumaker, L. (Eds.), Curve and Surface Design: Saint Malo 1999. Vanderbilt University Press, Nashville, pp. 153-162., 2000.

[45] Horváth, I., Juhász, I., Számítógéppel segített gépészeti tervezés, Műszaki könyvkiadó, 1996.

[46] Hoschek, J., Lasser, D., Fundamentals of Computer Aided Geometric Design, A. K. Peters, Wellesley, MA, 1993.

[47] Hoschek, J., Pottmann, H., Interpolation and approximation with developable B-spline surfaces. In: Mathematical Methods for Curves and Surfaces, (Edited by M. Dæhlen *et al.*), pp.255-264, Vanderbilt University Press, Nashville, 1995.

[48] Hoschek, J., Schneider, M., Interpolation and approximation with developable surfaces. In: Curves and Surfaces with Applications in CAGD, (Edited by A. Le Méhauté *et al.*), pp.185-202, Vanderbilt University Press, Nashville, 1997.

[49] Hoschek, J., Schwanecke, U., Interpolation and approximation with ruled surfaces, in: The Mathematics of Surfaces VII. (ed.:Robert Cripps), Elsevier, pp. 213–231, 1988.

[50] Hoschek, J., Dual Bézier Curves and Surfaces, Surfaces in Computer Aided Geometric. Design, R. E. Barnhill & W Boehm, eds., North Holland, pp.147-156, 1983.

[51] Iglesias, A., Gálvez, A., Applying functional Networks to fit data points from B-spline surfaces, in: Proceedings of the Computer Graphics Internatinal, CGI', Hong-kong pp., 329-332, 2001.

[52] Juhász, I., Számítógépi grafika és geometria, Miskolci Egyetemi Kiadó, 1993.

[53] Klein, F., Vorlesungen Über Höhere Geometrie, Springer, Berlin, 1926.

[54] Kohonen, T., Self-organization and associative memory, Springer Verlag, 1984.

[55] Kovács E., Studying and improving linear mappings by artificial neural networks, Acta Acad. Paed. Agriensis TOM. XXVI., Sectio Matematicae, pp.104-107, 1999.

[56] Kovács E., Curve reconstruction from scaterred data by Kohonen network, Acta Acad. Paed. Agriensis, Vol. XXVII., Sectio Matematicae, pp.69-74., 2000.

[57] Lang, J., Röschel, O., Developable $(1, n)$-Bézier surfaces. Computer Aided Geom. Design. 9 291-298, 1992.

[58] Lippmann, R.P., An Introduction to Computing with Neural Nets, IEEE ASSP Magazine, pp.18-21, April 1987.

[59] Loop, C., DeRose, T., Generalized B-spline Surface of Arbitrary Topology, Computer Graphics, Vol.24., N.4., pp.347-356, 1990.

[60] Ma, W., Kruth, J., Parametrization of randomly measured points for least squares fitting of B-spline curves and surfaces , *C*omputer Aided Design, 27 (9), pp. 663-675, 1995.

[61] Márkus, A., Renner, G., Váncza, J., Genetic algorithms in free form curve design, Daehlen, M., Lyche, T., Schumaker, L. (Eds.), Mathematical Methods for Curves and Surfaces, Vanderbilt University Press, pp. 343-354., 1995

[62] McCullogh, W. és Pitts, W., How We Know Universals: the Perception of Auditory and Visual Forms, *B*ulletin of Math. Biophysics, Vol.9, pp.127-147, 1947.

[63] McCullogh, W. és Pitts, W., A Logical Calculus of the Ideas Immanent in Nervous Activity, Bulletin of Math. Biophysics, Vol.5, pp.115-133, 1943.

[64] Neumann, J., Probabilistic Logic and the Synthesis of Reliable Organisms from Unreliable Components, in: Automata Studies, Princeton University Press, Princeton, pp.43-98, 1956.

[65] Nielson, G.M., Franke, R., Surface Construction Based Upon Triangulations, in: Surfaces in CAGD, North-Holland, Amsterdam, 1983.

[66] Pfeifle, R., Seidel, H.P., Spherical Triangular B-splines with Application to Data Fitting, EUROGRAPHICS '95, Vol.14., N.3., pp.89-96, 1995.

[67] Piegl, L., Tiller, W., The NURBS Book, Springer Verlag, 1997.

[68] Pottmann, H., Farin, G., Developable rational Bézier and B-spline surfaces, Computer Aided Geometric Design, 12, pp. 513–531, 1995.

[69] Pottmann, H., Peternell, M., Ravani, B., An introduction to line geometry with applications, Computer Aided Geom. Design. 31, pp.3-16, 1999

[70] Pottmann, H., Wallner, J., Computational Line Geometry, Springer -Verlag, Berlin Heidelberg, 2001.

[71] Pottmann, H., Wallner, J., Approximation Algorithms for Developable Surfaces, Technical Report, Nr.51, 1998.

[72] Press, W. H., Flannery, B. P., Teukolsky, S. A., Vetterling, W. T., Numerical Recipes in Pascal, The Art of Scientific Computing, Cambridge University Press, 1989.

[73] Rapcsák, A., Tamássy, L., Differenciálgeometria I-III., Tankönyvkiadó Budapest, 1980.

[74] Rényi, A., Wahrscheinlichkeitsrechnung, VEB Deutscher Verlag der Wissenschaften, 1962.

[75] Riesenfield, R.F., Applications of B-spline Approximation to Geometric Problems of Computer Aided Design, PhD disszertáció, Syracuse University, 1973.

[76] Rogers, D. F., Adams, J. A., Mathematical elements for computer graphics, Second Edition, McGarw-Hill pulishing Company, 1990.

[77] Rojas, R., Neural Networks. A Systematic Introduction, Springer-Verlag, 1996.

[78] Rousseeuw, P. J., Leroy, A. M., Robust Regression and Outlier Detection, Wiley, New York, 1987.

[79] Schoenberg, I. Contributions to the problem of approximation of equidistant data by analytic functions, Quart. Appl. Math, 4:45-99, 1946.

[80] Shepard, D., A two Dimensional Interpolation Function for Irregularly Spaced Data,ProceedingsoftheACMNat.Conf., pp.517-524, 1965.

[81] Strommer, Gy., Geometria, Tankönyvkiadó Budapest, 1988.

[82] Struik, D. J., Lectures on Classical Differential geometry, Dover Publications, Inc., Reprint, 1988.

[83] Tiller, W., Rational B-splines for Curve and Surface Representation, IEEE Comp.Graph. and Appl., Vol.3., N.9., pp.36-46., 1983.

[84] Torkkola, K. et al., Status Report of the Finnish Phonetic Typewriter Project, in: Kohonen et al (eds.): Artificial Neural Networks, North-Holland, Amsterdam, pp.771-776, 1991.

[85] Várady, T., Martin, R., Cox,J., Reverse engineering of geometric models –an intruduction, Computer Aided Gemetric Design, 29 (4) pp. 255-268, 1997.

[86] Várady, L., Hoffmann, M., Kovács, E., Improved Free-form Modelling of Scattered Data by Dynamic Neural Networks, Journal for Geometry and Graphics, Vol.3,No.2, pp.177–181, 1999.

[87] Várady, L., Analysis of the Dynamic Kohonen Network Used for Approximating Scattered Data, Proceedings of the 7th ICECGDG, Cracow, pp.433–436, 1996.

[88] Weiss, V.,Andor, L., Renner, G., Várady, T., Advanced surface fitting techniques, Computer Aided Gemetric Design, 19 pp. 19-42, 2002.

[89] Yamaguchi, F., Curves and Surfaces in Computer Aided Geometric Design, Springer-Verlag, Berlin, 1988.

[90] Zigány, F., Ábrázoló geometria, Egyetemi tankönyv, Tankönyvkiadó Budapest, 1951.

8. Kovács Emőd publikációs listája

1. Hoffmann M., **Kovács E.:** Developable surface modelling by neutral network, *Computers & Mathematics with Applications,* CAM5372 (közlésre elfogadva, megjelelnés alatt).

2. Hoffmann M., **Kovács E.:** Constructing ruled B-spline surfaces by Kohonen neutral network, *Acta Acad. Paed. Agriensis*, TOM. XXVII., Sectio Matematicae, pp. 63-68, 2000. (Math. Rev.: 1812710)

3. **Kovács E.:** Curve reconstruction from scattered data by Kohonen network, *Acta Acad. Paed. Agriensis*, TOM. XXVII., Sectio Matematicae, pp. 69-74, 2000. (Math. Rev.: 1812711)

4. **Kovács E.:** Studying and improving linear mappings by artificial neural networks, *Acta Acad. Paed. Agriensis* TOM. XXVI., Sectio Matematicae, pp. 104-107, 1999. (ZB.: 951.68160)

5. Várady L., Hoffmann M., **Kovács E**.: Improved free-form modelling of scattered data by dynamic neural networks, *Journal for Geometry and Graphics*, Vol.3., pp. 177-183, 1999. (ZB.: 940.68146, CompuTec: 19000829851)

6. **Kovács E.:** Az ábrázoló geometria számítógéppel segített oktatásának tapasztalatai, *Informatika a Felsőoktatásban '99* Országos Konferencia Debrecen 1999. augusztus 27-29. Konferencia kötet

7. Hoffmann M., **Kovács E**.: Interpolation possibilities using rational B-spline curve, *Acta Acad. Paed. Agriensis*, TOM. XXV., pp. 103-107, 1998. (ZB.: 952.41008, Math. Rev.: 1728607)

8. **Kovács E.**, Hoffmann Miklós: Monge projekció oktatásának támogatása számítógéppel, *Kandó Kálmán Műszaki Főiskola Centenáriumi Tudományos Ülésszaka,* Budapest 1998. május 6-7. Konferencia kötet

9. **Kovács E.**, Hoffmann M.: Computer Aided Teaching of Descriptive Geometry, *Proceedings of the 3th International Conference on Applied Informatics, Eger*, Vol.I., pp. 179-184, 1998. (CompuTec: 19000407266)

10. **Kovács E.:** A számítógépi grafika oktatásának tapasztalatai az EKTF-n, *Informatika A Felsőoktatásban '96 Országos Konferencia* Debrecen 1996. augusztus 27-30. Konferencia kötet

11. **Kovács E.:** Using some mathematical program in computer graphics teaching, *Proceedings of 7^{th} International Conference on Engineering Computer Graphics and Descriptive Geometry*, Crakow vol II., p.546-549, 1996. (CompuTec: 19990205657)

12. **Kovács E.:** Using Maple in teaching of computer graphics, *Proceedings of the International Conference on Applied Informatics* , Eger, 1995.

Szakanyag:

1. **Kovács E.:** Fejezetek a számítógépi grafikából, *Szakanyag KOMA 1995/1-777 pályázat támogatásával.* (67 oldal jegyzet), Pályázat címe: „Felkészítési lehetőségek a közoktatásban dolgozó tanárok informatikai képzettségének fejlesztésére"

2. **Kovács E.:** A teknőc és a rekurzív görbék, *Inspiráció, Az Informatika-Számítástechnika Tanárok Egyesületének lapja*, 5. évfolyam 2.szám 21.-22. oldal.