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

Doktori (PhD) értekezés tézisei

Kovács Emőd

Debreceni Egyetem
Természettudományi Kar
Debrecen, 2003.

Tartalomjegyzék

1. Eredmények **2**

1.1. A dinamikus Kohonen-háló alkalmazása ponthalmazra illesztett szabad formájú felületek előállítására. 2

1.1.1. A dinamikus Kohonen-háló 2

1.1.2. Új neuronok beszúrása 3

1.1.3. Az új algoritmus hatékonysága 4

1.2. B-spline vonalfelületek konstruálása Kohonen-háló segítségével . . . 4

1.2.1. Vonalfelület konstruálása előre megadott egyenesekből 4

1.2.2. Az algoritmus bemutatása 6

1.2.3. Megjegyzések 7

1.3. Kifejthető felületek konstruálása Kohonen-háló segítségével 7

1.3.1. Távolságfüggvény a síkok terében 8

1.3.2. Approximáció Kohonen-hálóval 9

2. Results **10**

2.1. Improved free-form modelling of scattered data by dynamic neural networks 10

2.1.1. The Dynamic Kohonen Network 10

2.1.2. Insertion of new neurons 11

2.1.3. Efficiency of the new algorithm 12

2.2. Construction ruled B-spline surfaces by Kohonen neural network . . 12

2.2.1. Construction ruled surfaces with given rulings 13

2.2.2. The algorithm 14

2.2.3. Remarks 15

2.3. Developable surface modelling by Kohonen network 15

2.3.1. Distance function in the space of planes 16

2.3.2. The approximation by Kohonen-network 17

3. References **18**

4. Publications of Emőd Kovács **25**

1. Eredmények

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 vonal- és kifejthető felületekre.

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

A Kohonen-féle neurális hálózat kiválóan alkalmazható szabad formájú felületek approximációjában. Az eredeti algoritmus megtalálható a [43]-ben, ezért itt csak a változtatásokat közöljük.

1.1.1. 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.

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.

1.1.2. 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 (lásd [43]), a következő lépéseket kell végrehajtani:

1. Minden aktivitási változót összehasonlítunk a határértékkel. Ha bármilyen 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.

1.1.3. 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). 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.

1.2. B-spline vonalfelületek konstruálása Kohonen-háló segítségével

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

1.2.1. 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.

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 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. Mindemellett a Kohonen hálóra épített 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 a Plücker-koordinátákat.

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:

$$(l_1, ..., l_6) := (l_{41}, l_{42}, l_{43}, l_{23}, l_{31}, l_{12}), \quad l_{ij} = x_i y_j - x_j y_i. \tag{1}$$

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

$$l_1 l_4 + l_2 l_5 + l_3 l_6 = 0. \tag{2}$$

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.

1.2.2. 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. A $t = 1$ 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.

1.2.3. 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 igyekszik az adatok nyújtotta lehetőségen belül legkisebb felületi energiával rendelkező alakot felvenni, azaz megpróbálja minél szűkebben approximálni az adatokat a legkisebb felszíni alak felé tartva. 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.

1.3. Kifejthető felületek konstruálása Kohonen-háló segítségével

A vonalfelületek és speciális részhalmazuk, a kifejthető felületek, jól ismert területek a klasszikus geometriában. 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.

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

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

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

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.

A dolgozat szempontjából a háromdimenziós $\mathbb{P}^3$ projektív térben alkalmazzuk a dualitás elvét.

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

2. 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.

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.

1.3.1. Távolságfüggvény a síkok terében

Hivatkozunk 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.

3. 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*

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

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

$$d_\mu(\mathbf{U}_1, \mathbf{U}_2)^2 = \sum_i ((u_{0,1} - u_{0,2}) + (u_{1,1} - u_{1,2})x_i + (u_{2,1} - u_{2,2})y_i)^2. \qquad (3)$$

1.3.2. Approximáció Kohonen-hálóval

Megadjuk a Kohonen-háló alkalmazásának azon módosításait, 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 abban a 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, amelyet (3)-ban definiáltunk. 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.

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 tartva, amely a legtöbb esetben fölöslegessé teszi egyéb javító technikák alkalmazását. További előnye a módszernek az, hogy egyaránt alkalmazható meglehetősen kevés input adat és eloszlással megadott nagyszámú elvileg végtelen sok input adat esetében is.

Constructing ruled B-spline surfaces by neural networks, computer graphics relations

Theses of PhD dissertation

Emőd Kovács

Debreceni Egyetem
Természettudományi Kar
Debrecen, 2003.

2. Results

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. The next part of the dissertation contains a further application of the previous method for ruled and developable surfaces.

2.1. Improved free-form modelling of scattered data by dynamic neural networks

Since the original method has been described in [43], now we discuss only the development of the network, the modification of the algorithm and the results which prove the increase of the effectiveness of the algorithm.

2.1.1. The Dynamic Kohonen Network

The original Kohonen network consists of two layers of neurons. The input layer has three neurons, since the network will be trained by the coordinates of the 3D input points, while in the output layer the number m of neurons depends on the number n of input points, generally $m = 4n$, or an appropriate number, if n is large. The two layers are totally interconnected, and a weight is associated to every connection. These weights can be considered as the spatial coordinates of vertices of a grid, the predefined topology of which is quadrilateral. During the training process the weights will be changing, hence this grid will move slowly in the three dimensional space toward the input points, meanwhile the topology of the grid will remain the same.

As we can see, the number of output neurons is a crucial point of the network and the choice of m has some uncertainty. To avoid this problem, the dynamic version of the Kohonen network is applied, in which the number of the output neurons, and hence the number of the vertices of the grid is growing dynamically during the training session.

The basic idea of the dynamic change of the net is the following: in every iteration of the training an active neuron is chosen, which is the closest vertex of the grid to the input point. This vertex and its neighbors is moved toward the input point. Suppose, that a certain neuron, or the neurons of a neighborhood is active very frequently. This means that geometrically this part of the grid is not dense enough, there are several input points around this neuron or this neighborhood. So extra neurons will be inserted next to the most frequently chosen neuron.

Since the insertion of one neuron would deform the topology of the grid, we will insert a row or a column of neurons instead. This row or column will be inserted to that side of the neuron, on which the neighbor neuron is the least active. This way the grid will grow dynamically, and will reach the most appropriate size to the end of the training.

2.1.2. Insertion of new neurons

Now we give the formal description of this insertion algorithm. Let the number of output neurons kxl, ($k = 2, l = 2$ at the beginning of the procedure). A resource variable $\lambda_{ij}, (i = 1, ..., k; j = 1, ..., l)$ is associated to every output neuron. The $(i_0 j_0)^{th}$ resource variable will be increased by 1 when the $(i_0 j_0)^{th}$ output neuron will be active. Moreover, a constant Λ is defined as an upper limit of the λ_{ij}. If one of the variables will be equal to this limit (i.e. this neuron has been chosen *Lambda* times), a row or a column will be inserted next to the associated neuron.

After the former training steps the following steps must be executed:

- STEP 1. Compare the resource variables with the limit. If there is an index ij, for which
$$\lambda_{ij} = \Lambda,$$
then
- STEP 2. Choose the least frequent neighbor of this neuron, that is find
$$min(\lambda_{i-1,j}, \lambda_{i,j-1}, \lambda_{i+1,j}, \lambda_{i,j+1})$$
(some of them may not exist, if the neuron is on the border of the grid)
- STEP 3. Suppose, that the neighbor we found is the $(i, j-1)^{th}$ neuron. Then a column will be inserted between the $(j-1)^{th}$ and the j^{th} columns. The weights of the neurons of this new column, as the coordinates of the new vertices of the grid, will be chosen as the average of the weights of their two neighbors, hence geometrically these new vertices will be the midpoints of the sections of their neighbors. k or l has to increase accordingly, in this example $l = l + 1$.
- STEP 4. Reset all the resource variables: $\lambda_{ij} = 0, (i = 1, ..., k; j = 1, ..., l)$.

A further advantage of this insertion, that the ratio of k and l, i.e. the shape of the grid is changing automatically and will be the most advantageous after the training procedure.

The original method can also be modified in terms of stopping criteria. If the number of input points is relatively small, then the original network is said to

be trained if all the input points are on the grid. If we have hundreds of input points, or data given by a distribution, as in some of the scattered data problems, then the network is said to be trained if the changes of the grid is under a certain predefined limit. In this latter case the dynamic network would be also stopped by exceeding of a predefined limit for the number of neurons (hence the number of output neurons can even be smaller than the number of input points).

2.1.3. Efficiency of the new algorithm

The new algorithm improves the method in two ways. First of all, the training iterations, that is the number of presenting input values decreases dramatically. Since the insertion needs only very limited computing time, finally the new algorithm is much faster, than the original one. The table below shows the changes of the number of sufficient iterations (average after several runs). On the other hand, the number of vertices of the final grid is also decreases, which yields a smoother surface.

After the training session a grid will be obtained produced by the Kohonen net. This grid can be considered as the control mesh of the future surface. Either interpolation or approximation can be executed by the standard NURBS surface or Bézier-surface method and at this time of the procedure it is irrelevant, that the starting points were scattered so we can consider them as regular, ordered vertices of a control mesh. We have to notice, this algorithm is modified and by using the dynamic version of the Kohonen algorithm, which allows to change the number and structure of the output neurons during the training session. This possibility yields less computing time and output neurons, while the shape of the grid also fits better to the input points. These advantages help to model the scattered points with a better free-form surface.

2.2. Construction ruled B-spline surfaces by Kohonen neural network

Ruled surfaces and their special subclass, developable surfaces are well-known in classical geometry, but also have extended applications in computer geometry and graphics as well as in computer aided manufacture. In these latter fields, however, the standard way of descripting surfaces is the different types of spline-surfaces, like Bezier, B-spline or NURB surface. The standard definition of any type of spline surfaces uses a quadrilateral control grid as input data. The main purpose of most of the known methods is to construct this control grid. In our method the Kohonen neural network has been applied in a preprocessing step to produce a control grid from the original input data. In our paper original input data structure consists of a set of given lines, called rulings. In this case the description of these lines

and the neural network by homogeneous coordinates will be advantageous, which is a well-known technique in projective geometry. The Plücker coordinates of the projective lines will be also applied.

2.2.1. Construction ruled surfaces with given rulings

Consider the following problem: an arbitrary set of lines are given, find a ruled surface passing these lines, that is a surface for which the given lines are rulings.

More precisely, we would like to interpolate these given lines with one of the standard spline surfaces of CAD, a Bézier, a B-spline or a NURB surface. In this paper the B-spline surface will be used, but all of these surfaces are defined by a control grid of points, called control points. This grid regularly has a quadrilateral topology, and one can chose to approximate or to interpolate these points.

1. Theorem. *If the points of the control grid of a B-spline surface form straight lines in one direction and the type of the grid is* $(2, n)$ *, then the result surface is ruled surface.*

Thus we need to form a quadrilateral grid in which the vertices form straight lines in the direction of the given lines. These lines will be the rulings of the future surface, if it will interpolate the control points, and the theorem mentioned above yields it will be a ruled surface. To find this grid the Kohonen neural network will be applied. This tool will produce the desired grid from the given lines, and then this grid will be the input control grid of the standard interpolation method of the B-spline surface.

Here we have to introduce a very effective tool for handling spatial lines: the Plücker-coordinates. Let us briefly overview this classical part of geometry.

Consider the three dimensional projective space P^3. Here every point has four homogeneous coordinates (x_1, x_2, x_3, x_4), which correlate the Cartesian coordinates (x, y, z) as $x = x_1/x_4, y = x_2/x_4, z = x_3/x_4$ if the point is not at infinity. Now consider a line l of P^3 passing two points (x_1, x_2, x_3, x_4) and (y_1, y_2, y_3, y_4). The six Plücker coordinates of this line will be

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

These coordinates do not depend on the choice of the two points on the line l and fulfill the Plücker identity:

$$l_1 l_4 + l_2 l_5 + l_3 l_6 = 0.$$

This 1-1 correspondence between the homogeneous 6-tupels of real numbers and the lines of $\mathbb{P}^3$ yields another 1-1 correspondence between the lines of $\mathbb{P}^3$ and the points of the five dimensional projective space $\mathbb{P}^5$. With the help of this latter

mapping we can embed the given rulings and the lines of the future grid of the surface. This way the Kohonen network will be trained by points from $\mathbb{P}^5$, that is the input layer will contain 6 neurons, while the output layer will be a polygon in $\mathbb{P}^5$. When the neural network is trained, the result polygon passes through the given points of $\mathbb{P}^5$. Transforming this situation back to $\mathbb{P}^3$ the given lines and a quadrilateral grid will be received, which grid contains the given lines and consists of straight lines at that direction.

2.2.2. The algorithm

Let a set of lines $l^i, (i = 1, ..., n)$ be given. Compute the Plücker coordinates of these lines:

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

Consider a Kohonen neural network with 6 input neuron and $m(= 4n)$ output neuron. Denote the weights of the connection from the j^{th} input neuron to the i^{th} output neuron by w_{ij}. Thus the weights $(w_{i_0 j}), (j = 1, ...6)$ of the connections to an output neuron can be considered as the coordinates of an output point V_i in P^5.

Now we train the Kohonen neural network. Initializing the weights w_{ij} as small random values around the average of the coordinates of the input points and setting the training time $t = 1$, the coordinates $(l^{i_0}_j), (j = 1, ...6)$ of a randomly chosen input point is presented. The Euclidean distance of all output points to this input point is computed:

$$d_m = \sum_{s=1}^{6} (l^{i_0}_s - w_{ms})^2$$

Finding the winning unit, that is the output point closest to the input one, the weights of the connections to this point and to its neighbors are updated by

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

where t denotes the training time. After updating the weights a new randomly selected input is presented until the network is trained. The network is said to be trained, if all the input lines are on the grid. If the input structure consists of hundreds of lines, then this requirement would yield long computing time, hence in this case the network is trained if the changes of the weights fall under a predefined limit.

When the training process finished, the result is a polygon in $\mathbb{P}^5$ with the vertices $V_i(w_{ij}), (i = 1, ..., m), (j = 1, ..., 6)$. Transforming these vertices back to P^3 the projective lines v_i are received, with the Plücker coordinates (w_{ij}). Using the definition of these coordinates backwards, the points of these lines can be

computed. Thus a set of lines is gained among which the all the original input lines can be found.

During the training session one can compute the whole grid at every training time t, using the points of the lines v_i as vertices connected also in the other direction. Hence after the training procedure beside the rulings a quadrilateral grid containing these rulings is also received. The vertices of this grid can be applied as input control points to any of the standard free-form surface algorithm. Hence we can interpolate or approximate these points and the rulings.

2.2.3. Remarks

Obviously there are infinitely many ruled surface passing through a set of given spatial rulings, and probably each of the methods would produce a different one. Comparing our method with other algorithms the main advantage is the following: beside the self-organizing ability of the Kohonen network it also has a minimal energy property, that is the network tries to minimize the strain energy during the training session. Hence in a sense the result grid is the "smoothest" grid containing all the given rulings, while other methods cannot guarantee this property.

2.3. Developable surface modelling by Kohonen network

Ruled surfaces and especially developable surfaces are well-known and widely used in computer aided design and manufacture. Since these fields apply B-spline or NURBS surfaces as de facto standard description methods, it is highly desired to use these methods to construct ruled and developable surfaces from any type of data.

The a ruled surface is covered by a one parameter set of lines, the rulings. These lines has the following property: a ruling line lies in the tangent plane of the surface at every point of the ruling. This property is important in terms of the special ruled surfaces, called developable surfaces. The original definition of these surfaces is the following.

1. Definition. *A surface in $\mathbb{E}^3$ is called developable surface, if it can be isometrically mapped (i.e., developed) onto a plane.*

Considering the property of the ruled surfaces mentioned above and this latter definition, one can easily see the following, well-known result.

2. Theorem. *A ruled surface is developable, if and only if the tangent planes at all points of an arbitrary ruling coincide.*

If the tangent planes vary along the ruling (i.e., they are different), the surface is called general (non developable) ruled surface. This means that considering

the tangent planes of a surface, the developable surface can be described as an envelope of a one parameter family of planes, while the set of tangent planes of a ruled surface has a second parameter along the rulings.

The principle of duality can be applied to describe developable NURBS surfaces as dual spatial NURBS curves, as we can see from the following definition.

2. Definition. *A nonuniform rational B-spline, i.e., NURBS curve* $\mathbf{s}(t)$ *can be written in the following homogeneous form*

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

where $N_i^k(t)$ *are the normalized B-spline functions of order* k *over a knot vector* $t_0 = t_1 = ... = t_k < t_{k+1} < ... < t_n < t_{n+1} = ... = t_{n+k+1}$*, while* $\mathbf{p}_i(w_i, w_i x_i, w_i y_i, w_i z_i)$ *are the control points of the curve.*

The dual form of this curve is

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

with planes $\mathbf{U}_i$ $(w_i, w_i x_i, w_i y_i, w_i z_i)$.

This dual form can be interpreted as an envelope of a one parameter set of planes, i.e., this is a developable NURBS surface.

The above mentioned definition yields, that if a set of planes are given one can find the developable NURBS surface as an envelope of these planes with a curve approximation technique of scattered points in the dual space. In this space, however, first an appropriate measure has to be found, which can produce a distance-like value for planes in an area of interest. This is necessary for us because in the preprocessing step the neural network has to compare the given data with its own data by their distance.

2.3.1. Distance function in the space of planes

The idea described in this section has been recently developed by Pottmann et al ([71]). Euclidean distance and invariants of planes, based on the angle of the two planes are inappropriate for our purpose, because we have to measure the difference between two planes only in an area of interest, and this difference can be arbitrarily large meanwhile the angle between the planes is arbitrarily small.

3. Definition. *If* $\mathbf{U}_1\,(u_{0,1}, u_{1,1}, u_{2,1})$ *and* $\mathbf{U}_2\,(u_{0,2}, u_{1,2}, u_{2,2})$ *are two planes in* A^**, then their distance over* D *is the following*

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

i.e., the $L^2(\mu)$-distance of the linear functions whose graphs are $\mathbf{U}_1$ and $\mathbf{U}_2$, where μ is a positive measure in $\mathbb{R}^2$.(These linear functions are supposed to be always in $L^2(\mu)$).

The measure μ can be defined in different ways (see also [8]). For our purpose the simple form

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

has been chosen, i.e., the sum of some point masses at points (x_i, y_i).

2.3.2. The approximation by Kohonen-network

Now, we give the changes of the general algorithm (described in [43]) in case of the approximation of a set of planes by their envelope, a developable NURBS surface. Here, the input data will be the coordinates of the planes $\mathbf{U}_i(u_{0,i}, u_{1,i}, u_{2,i})$. The main difference, however, appears in step 3. where originally the distance $d_j(\mathbf{p}_{i_0}, \mathbf{q}_j)$ is used. In the general algorithm and applications it is the standard Euclidean distance, but in our case, since we are in the dual space P^*, the special distance d_μ is used which has been defined in the previous section. With this change the neural network algorithm automatically produces the "control polygon" of the dual curve, which can be transformed to the normal tensor product surface. Hence, the scattered set of input planes is modelled by a developable standard NURBS surface.

We have to mention 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 rather few input data as well as infinitely many data given by a distribution

3. References

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 Free-form 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, Lecture-Notes 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.

4. Publications of Emőd Kovács

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.