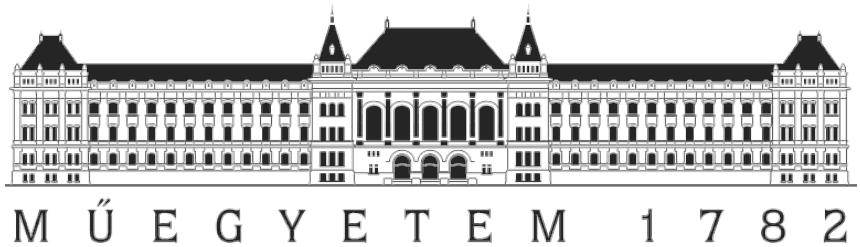
Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar
Irányítástechnika és Informatika Tanszék

# Procedurális ecsetminta generálás Markov mezőkkel, filmipari alkalmazás céljaira

*Készítő:*
Zsolnai Károly,
BME-IIT
zsolnai@iit.bme.hu

*Konzulens:*
Szirmay-Kalos László,
BME-IIT
szirmay@iit.bme.hu




2010. december 7.

# Tartalomjegyzék

**Hallgatói nyilatkozat**

Alulírott Zsolnai Károly szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök, stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. A teljes szöveg közzététele dékáni engedéllyel titkosított diplomatervekre nem vonatkozik.

Budapest, 2010. december 7.

.....

szigorló hallgató

**Összefoglaló**

A dolgozatban egy olyan procedurális eljárás részleteit tárgyaljuk, amellyel rajz- és festőművészek ecsetmintáinak stílusjegyeit elemezve azokból akár végtelen hosszú szekvencia készíthető, amit már meglévő, számítógépes tervezés során elkészült képekre és jelenetekre alkalmazva azt az illúziót kelti, mintha a művész sajátkezűleg rajzolta volna őket. A megközelítés alapjait már meglévő kutatási eredmények adják, amelyek továbbfejlesztésével a módszer éles ipari alkalmazása is lehetővé válik - az állóképek torzításán kívül az új algoritmus egy animációs mozifilm elkészítésében is szerepet kapott.

A módszer nehézsége, hogy a művész által készített ecsetvonások stílusának megőrzésén túlmenően nem engedhető meg, hogy a generált vonalak bármilyen szemmel látható periodicitással rendelkezzenek. Az algoritmus a művészi stílus elemzéséhez valószínűségi megfontolásokat alkalmaz: a Markov mezők tulajdonságai alapján az átmeneti valószínűségek függetlenné tehetők a megelőző állapotoktól. Az ecsetminta ennek következtében tetszőleges pontról, előzetes ismeretek hiányában is akár végtelen hosszúságban folytathatóvá válik.

Az eredményül kapott ecsetminta periodicitásának elkerüléséhez az útvonalakat nem helyben, hanem már előzetes tervezés során rangsoroljuk - azokat a lehetőségeket, amelyek nagy valószínűséggel a görbe végállapotához vezethetnek, az elemzés folyamán plusz költséggel büntetjük. Ehhez a mesterséges intelligencia területén jól ismert megerősítéses, vagy Q-tanulás módszerét alkalmazzuk.

Az eredmények ismertetése mellett a dolgozatban rámutatunk a már meglévő eljárások hiányosságaira, illetve arra, hogy miért nem bizonyulnak elégségesnek a fenti probléma megoldására, továbbá szó esik az ismertetett algoritmus egyéb területeken való felhasználási lehetőségeiről is.

**Abstract**

In this paper we discuss a procedural method that examines the brush strokes of painter artists and then generates similar strokes to render 3D objects on the computer. The result of the method maintains the illusion that the newly created picture was made by the particular artist. The approach is based on previous research results and improves them to be appropriate for film industry applications. The developed method is integrated in a fully featured film rendering pipeline.

The main challenge of the problem is that any kind of visible periodicity is not acceptable while the algorithm's main aspect is the preservation of the artistic style of the painter for arbitrarily long strokes.

We apply a probabilistic approach to analyze the dynamics of motion: according to the attributes of Markov fields we can make state transition probabilities independent of the preceding states. Thus, we are able to compute the next state of the spline from any point. To avoid visibly periodic output, it is a good idea to try to plan ahead and discover the path of transitions that lead to the last state of the sample stroke, and assign higher cost to them in advance to reduce the probability of not matured termination. To accomplish this, we iteratively apply and solve the reinforcement learning (or Q-learning) equation.

The paper also investigates other procedural approaches and discusses the reasons why they cannot be utilized for the same problem. There is also a discussion of other possible fields that may benefit from the algorithm.

# 1. Bevezető

A bevezetőben a későbbiekben felhasznált fogalmak pontos definíciói szerepelnek, illetve ejtünk néhány szót a dolgozatban taglalt módszer alapproblémáiról, és azok megoldásának nehézségeiről. A fejezetben rövid leírást adunk a procedurális grafika és procedurális szintézis területein született kutatási eredményekről is.

## 1.1. PR, NPR

A számítógépes grafikai képszintézis egyik fő ága, a fotorealisztikus képszintézis (photorealistic rendering) olyan matematikai és fizikai modellekkel dolgozik, amelyekkel a való világban észlelt jelenségek lehető leghitelesebb, leghasonlóbb képmását lehet előállítani[1]. A módszer legelső alkalmazói között tartjuk számon napjaink mozgóképiparában standardnek tekintett, a Pixar cég által készített Renderman nevű modellező- és renderelőprogramot[2].

A nem fotorealisztikus képszintézis (non-photorealistic rendering, NPR) területének megszületését elsősorban a rajzolt és festett művek motiválták - a terület célja tehát nem a valósághű, hanem elsősorban esztétikus értékekkel rendelkező, művészi hatást keltő képek, képsorozatok megalkotása.

[1] A csatolt kép forrása: http://www.graphics.cornell.edu/online/box/compare.html
[2] https://renderman.pixar.com/

(a) Fotózott kép (b) Renderelt kép

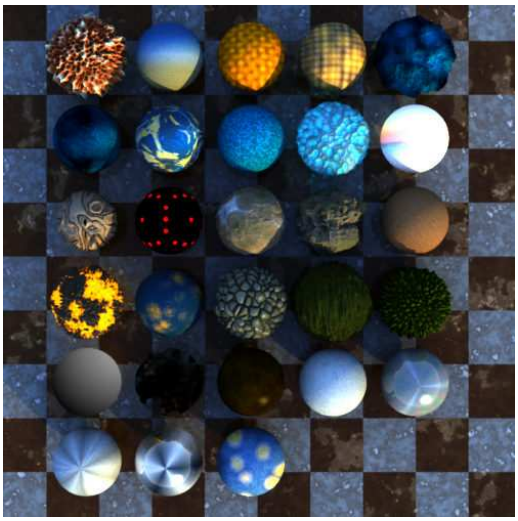
1. ábra. A Cornell egyetemen készült tesztdobozról készült fotó és a számítógépes grafikai PR szintézis összevetése.

2. ábra. A Cel Shader NPR technika alkalmazása.

## 1.2. Procedurális grafika

Procedurálisnak nevezünk minden olyan grafikai megoldást, amely algoritmikusan, matematikai megközelítéssel képes előállítani egy színtér bármely olyan részét, amelyet egyébként csak előre letárolt adatok betöltésével hozhatunk létre. A módszer jellegzetessége a véletlen számok matematikai eloszlás- és sűrűségfüggvények szerinti generálása, illetve struktúrált zajok alkalmazása. Egy jól eltalált összefüggés segítségével a semmiből teljes városokat, tájképeket, vagy

akár textúrákat teremthetünk.



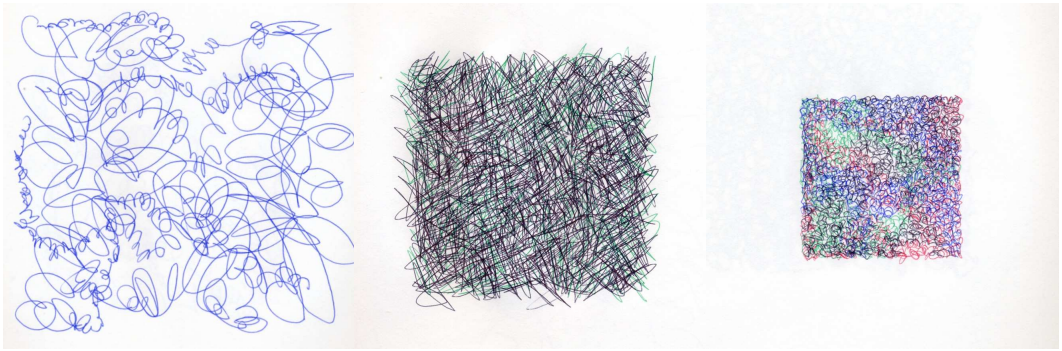
3. ábra. Procedurális eljárásokkal készített textúraminták.

## 1.3. Struktúrált zajok, textúraszintézis

Ken Perlin 1985-ben egy megoldást javasolt, amelyben véletlen számok felhasználásával nyílt lehetőség struktúrált, valódinak látszó textúrák készítésére [7]. Az évek során a módszer több változata is napvilágot látott, univerzalitása révén a későbbiekben számos egyéb grafikai területen is teret nyert - használatával sikeresen generálhatók természeti színterek magasságtérképei, terepek, vízfelületen sodródó hullámok is. A megközelítés igen fontos további eredménye, hogy napjainkban már a procedurális textúraszintézis alapkövének tekinthetjük. A lehetőségek kellően elmés kihasználására bőven akadnak kiváló példák - sikeres szimulációk születtek kézirajzok imitálására is, amiben a kimeneti zaj a grafika világában használt görbetípusok (1.4) kontrollpontjaiként került megadásra[3].

A számítógépes képfeldolgozóiparban évtizedes múltra tekint vissza a digitális fotók retusálásának gyakorlata. Nem egyszerű feladatról van szó, kiváló grafikusi teljesítményről ad számot, ha sikerül úgy módosítani egy adott képen szereplő valamely elemet, hogy azt az illúziót keltse, hogy az a valóságban is a képen látható

[3] A képek forrása: http://emoc.org/hackpact/



4. ábra. Struktúrált zaj alkalmazásával készült kézirajz imitációk.

módon szerepel. A módszer fő nehézsége, hogy az adott elem környezetében szereplő részletek mellett még kellő finomsággal, műgonddal dolgozva is igen csekély azon megoldási lehetőségek száma, amik valódi, természetes benyomást keltenek.
Napjainkban mindehhez már algoritmikus segítséget is kaphatunk. A pixel-alapú textúraszintézist alkalmazó megoldások a kapott minta vizsgálatának során valószínűségi sűrűségfüggvényeket készítenek, amiket felhasználva az új területrészeket pixelenként alkotnak meg. Az éppen soron levő pixel környezetének elemzésével megtippelhető, hogy az eddigi minták alapján mi a legvalószínűbb szín, amire kitölthetjük azt. Az eredeti képhez való hasonlóság a sűrűségfüggvény adott helyén vett várhatóértékére való tippeléssel garantálható [2, 12].
Egy másik megközelítés, a folt-alapú (patch-based) textúraszintézis a kért területeket úgy tölti ki, hogy a meglévő minta különböző méretű részhalmazaival vett illeszkedés mértékét vizsgálja, és megfelelő egyezés esetén ezzel a mintadarabbal folytatja az építkezést [1, 4].

## 1.4. Catmull-Rom spline

A görbéket folytonos vonalként definiáljuk. Legfontosabb tulajdonságuk, hogy egy olyan egyenlettel adhatók meg, amelynek megoldásai a görbe pontjait határozzák

meg. Egy háromdimenziós görbe megadása a következő formában történik:

$$x = x(t), \quad y = y(t), \quad z = z(t), \quad t \in [0,1].$$

Az explicit egyenletbe a $t$ paraméter egy lehetséges értékét behelyettesítve a görbe egy pontját kapjuk meg. A paramétert a teljes megengedett tartományon végigfuttatva a görbe minden pontját meglátogatjuk.

A görbe vezérlőpontjai a görbe által bejárt út irányát jelölik ki - az útvonalat azonban ennyivel nem lehet egyértelműen meghatároznunk, ugyanis azt nem definiáltuk, hogy az egyes pontok közötti átmenet milyen interpolációs módszerrel történjen, így azokhoz különböző, a $t$ paramétertől függő súlyértékeket rendelünk, és a bázisfüggvényben rögzítjük őket. Az $\vec{r_1} = [x_1, y_1, z_1]$ és $\vec{r_2} = [x_2, y_2, z_2]$ pontok közé húzható egyenes egyenlete:

$$x = x_1 \cdot (1-t) + x_2 \cdot t, \quad y = y_1 \cdot (1-t) + y_2 \cdot t, \quad z = z_1 \cdot (1-t) + z_2 \cdot t, \quad t \in [0,1],$$

illetve ugyanez vektoros jelölésrendszerrel:

$$\vec{r}(t) = \vec{r_1} \cdot (1-t) + \vec{r_2} \cdot t.$$

A bázisfüggvényt esetünkben tehát a $B_1(t) = 1 - t$ és $B_2(t) = t$ súlyokkal adtuk meg:

$$\vec{r}(t) = \vec{r_1} \cdot B_1(t) + \vec{r_2} \cdot B_2(t).$$

A $t = 0$ behelyettesítés a $\vec{r_1}$ pont koordinátáit adja meg, a paraméter növelésével a $\vec{r_1}$ pont súlya ugyanannyival csökken, amennyivel az $\vec{r_2}$-é növekszik, végül $t$ = 1-re $\vec{r_2}$ pontot kapjuk. A görbe tehát szépen, egyenletes sebességgel masírozik az egyik vezérlőpontból a másikba: a módszert lineáris interpolációnak nevezzük.

A görbék megrajzolásakor különböző, a mindennapi használat során felmerült

praktikus célokat tartunk szem előtt, az összetettebb spline típusokat pedig ezen elvek alapján osztályozzuk:

A különböző szerkesztőprogramokkal való munka folyamán egy üres területen kontrollpontokat helyezünk el - joggal várhatjuk el, hogy egy új pont hozzáadásakor a görbén csak az érintett részhez közeli tartományok változzanak. Bézier-görbe rajzolása esetén azonban keserű meglepetésre számíthatunk: a nagy műgonddal elkészített alkotásunkhoz új kontrollpont rendelésekor a görbe a teljes tartományon torzul. A kényelmes szerkesztés alapkövetelménye az ún. *lokális vezérelhetőség* lehetősége, ami azt kívánja meg, hogy a görbe bővítésekor az újonnan megadott pontok legfeljebb a szomszédaikra lehessenek hatással. Matematikailag ez annyit jelent, hogy az $i$-edik kontrollpontra $t$ paraméter valamely értékét behelyettesítve annak maximum néhány szomszédja rendelkezhessen pozitív súllyal, a távolabbi pontok súlya pedig zérus vagy negatív legyen.

A görbék *interpolációs* tulajdonsága azt határozza meg, számíthatunk-e egy kontrollpont elhelyezésekor arra, hogy a görbe majd nem csak annak irányában folytatja útját, hanem majd érinteni is fogja-e azt. B-spline esetén lehetőség van a görbe lokális vezérelhetőségére, ám a kényelemért az interpolációs jelleg feladásával kell fizetünk.

Egy épület alaprajzának modellezésekor alapkövetelmény az egyes szobák falainak pontos illeszkedése. Komoly bajba kerülhet az a mérnök, aki olyan tervrajzot ad ki a kezéből, amelyen az egyes termek falai egymást keresztezik, vagy ahol az egyes bútorok kilógnak az őket tartalmazó szobákból. Esetünkben a tervrajzhoz alkalmazott görbetípus feladata annak garantálása, hogy a görbe minden pontja a kontrollpontok összekötésével kapott területen belül helyezkedjen el - ezt *konvex burok* tulajdonságnak nevezzük. A tulajdonság a fentihez hasonló mérnöki gyakorlatokon túlmenően sok teljesen mindennapi alkalmazáshoz is elengedhetetlen.

Az általunk használt bázisfüggvények folytonossága esetén azok lineáris kombinációjaként előállt görbe is folytonos lesz. Ezzel a tulajdonsággal önmagában még nem tudjuk leírni a vonalak simaságát, tekintve, hogy egy meglehetősen szögletes görbe is folytonos tulajdonsággal rendelkezik. A kellően sima görbék esetén nem csak magára a görbére, hanem annak egy- vagy akár többedrendű deriváltjaikra is teljesül a folytonosság. A fenti tulajdonság osztályozására külön szinteket vezetünk be: $C^n$ osztályba soroljuk azokat a görbéket, amelyek legfeljebb $n$ alkalommal folytonosan deriválhatók. A dinamika alaptörvényének ($F = m\ddot{r}$) alkalmazhatóságának minimálisan elégséges feltétele a $C^2$ *folytonosság*: ha egy elhajított kő útvonalát leíró spline az első vagy a második deriválást követően ugrást szenved, a képernyőn lezajló események nem fognak megfelelni a valós fizikai törvényeknek, sőt - a hirtelen, ugrásszerűen változó mozgású test az emberi szem számára bántóan, természetellenesen hat.

A *Catmull-Rom spline* egy harmadfokú polinomokból összeillesztett görbetípus, amelynek szegmenseit 4-4 kontrollpont megadásával írjuk le, az egyes szegmensek találkozásánál a sebességet pedig a két tartomány sebességeinek átlagaként állítjuk elő [11]. Bázisfüggvényét a következőképp kapjuk meg:

$$r(t) = 0.5 \begin{pmatrix} 1 & t & t^2 & t^3 \end{pmatrix} \begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} r_{n-1} \\ r_n \\ r_{n+1} \\ r_{n+2} \end{bmatrix}$$

Értékeljük a bázisfüggvény és annak vizualizációja (5-6. ábrák) alapján a görbetípust!

**Interpoláló**. Az illeszkedési pontokban ($t = 0$ és $t = 1$) egy-egy vezérlőpont 1-es súllyal rendelkezik, így végül a kezdő- és a végponton kívül a görbe bejárása

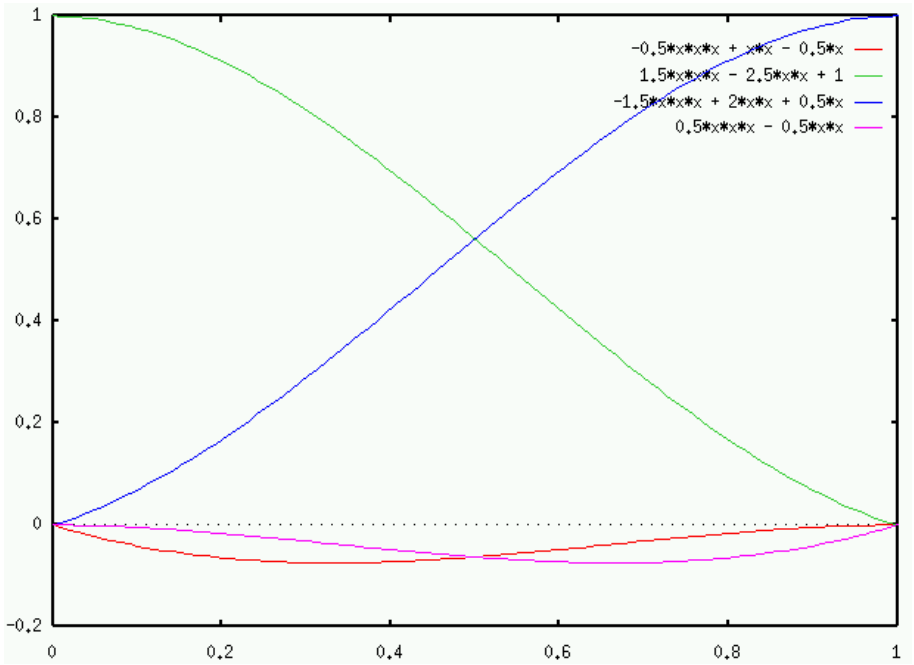
során minden vezérlőpontot meglátogatunk.

**$C^1$**. A görbe $C^1$ folytonos - egy harmadfokú polinomból álló függvényt egy alkalommal folytonosan deriválhatunk, azaz az egyes szegmensek egyszeres folytonossággal illeszkednek.

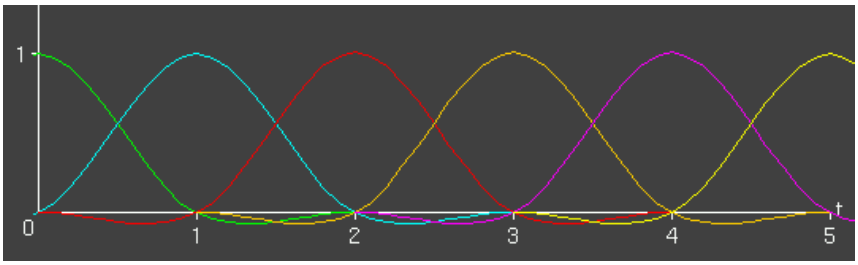
**Lokálisan vezérelhető**. A lokális vezérelhetőség feltétele teljesül: a bázisfüggvény rajzán látható[4], hogy az illeszkedési pontokat leszámítva tetszőleges pontot kiválasztva két pozitív és két negatív súlyértéket kapunk (a súlyok összege minden pontban 1). A két pozitív érték miatt egy kontrollpont megváltoztatása két, hozzá csatlakozó szegmensben okoz változást, illetve a deriváltakon keresztül két további szegmensben is, de több helyen semmiképpen sem.

**Nincs konvex burok**. A bázisfüggvényben negatív súlyértékek is szerepelnek, ezeken a helyeken az adott vezérlőpontban nem vonzó, hanem taszító hatás keletkezik, így a görbe elhagyja a kontrollpontokon felrajzolt konvex burkot.

[4] A csatolt képek forrásai: http://research.cs.queensu.ca/~jstewart/454/images/catmullRom/ és http://www.mvps.org/directx/articles/catmull/




5. ábra. A Catmull-Rom spline bázisfüggvénye (egy periódus).




6. ábra. A bázisfüggvény több periódusa.

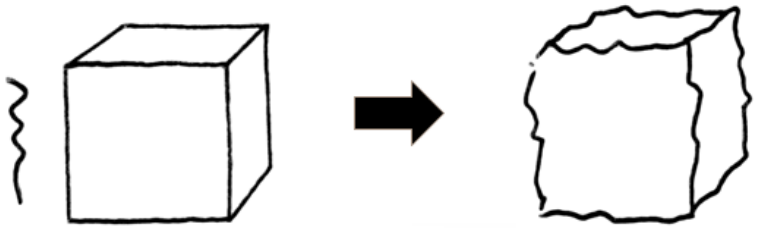
## 1.5. Motiváció

Az animációs filmipar napjaink népszerű modellezőszoftvereinek segítségével virtuális világok szereplőit és környezeteit állítja elő, hatékony képfeldolgozási eljárásaival pedig a színkészletek megfelelő módosításával tetszőleges érzetet és hangulatot képes a filmvászonra csalni - ugyanazon modellek különböző képfeldolgozási eljárásokkal meseszerű, művészek által festett karakterek illúzióját kelthetik, más beállítások mellett pedig lehetnek akár valódinak tűnő szereplők is.

A filmiparban használt modellezőszoftverekkel készített objektumok digitális,

számítógéppel kiszámított mivoltuk miatt teljesen szabályos geometriával rendelkeznek. Amennyiben a filmkockák pergésekor a karakterek sziluettjeire terelődik a tekintetünk, hamar szertefoszlik az illúzió: senki sem fogja elhinni, hogy a feltűnően tökéletes, szabályos vonalak egy képzőművész keze munkáját dicsérnék.

Jelen dolgozatban egy olyan algoritmus részleteit tárgyaljuk, amely képes a fenti problémára megoldást nyújtani: egy festőművész ecsetvonásaiból csekély méretű mintát vesz, annak stílusjegyeit megőrizve pedig meglévő modellek, kézirajzok körvonalait torzítja. A módszer felhasználásával olyan hatás kelthető, mintha azokat ténylegesen az adott művész festette volna.



7. ábra. Egy egyszerű számítógépes modell torzítása a mellette látható ecsetminta alapján.

# 2. A módszer ismertetése

## 2.1. A megoldás fő problémái

A megközelítés fő problémája a művészi vonalvezetés és az ecsetvonások matematikai reprezentációjának miként je. Egyszerű fehérzajjal nem sokra megyünk, valamilyen struktúrált zaj alkalmazása vezethet sikerre, ám ahhoz, hogy ezt matematikailag meg tudjuk fogni, elemezni kell, hogy mitől hordoz magában egy megrajzolt vonal esztétikai értékeket, illetve hogy mely jellegzetességek összessége alkotja a művész stílusjegyeit. Mindezen tényezők pontos matematikai elemzésének

lehetőségére és reprodukciójára van szükség.

A klasszikus textúraszintézis eljárásaival kapcsolatos elvárásunk, hogy ha egy véges méretű, valamilyen periodikus minta feldolgozása után egy hasonló mértékű periodicitást mutató kimenetet kapjunk. Egy festmény esetén ilyen eszközökkel aligha juthatunk sikerre - senki nem látott még olyan festőművészt a világon, akinek a csuklómozdulatain bármilyen, szemmel látható periodicitást lehetne megfigyelni.

## 2.2. Valószínűségi mátrixok

Az algoritmus a Robert D. Kalnins cikkében ismertetett kutatási eredmények elemeit használja fel [3]. A művész ecsetvonásainak tulajdonságait valószínűségi szemléletmód alkalmazásával elemezzük - eloszlásfüggvényt írunk fel minden állapotra, amely tartalmazza a görbe összes többi állapotába való átmeneti valószínűségeket. Definiáljuk egy $k$ pontból álló ecsetmintára $n = k+1$ mellett $D_{ij}$ $[n \times n]$-es mátrixot az $(i, j)$-edik állapotok függvényérték-különbségének abszolútértékével, azzal a kitétellel, hogy a legutolsó oszlopot, amely a trajektória végállapotába vezet, végtelen értékkel inicializáljuk. A főátló elemeit nem szükséges külön kiszámolni, azok mindig 0 értéket vesznek fel:

$$D_{ij} = \begin{cases} 0, & \text{ha } (i = j \vee i = n) \\ \infty, & \text{ha } (i \neq n \wedge j = n) \\ |y_i - y_j|, & \textit{egyébként.} \end{cases} \tag{1}$$

A Markov-mezős megközelítés az állapotátmeneti valószínűségeket csak a függvény jelenlegi állapota alapján, annak korábbi előéletétől függetlenül állapítja meg. Az ecsetvonások dinamikájának megőrzése érdekében az egyes állapotok külön-külön, egyenként való vizsgálata helyett az $i$-edik állapotban az azt megelőző $[i-1, i-$

$-2, \ldots, i-m]$ állapotok tulajdonságait is figyelembe vesszük:

$$D'_{ij} = \sum_{k=0}^{m} w_i \, D_{i-k,j-k}. \tag{2}$$

Az említett ablakméretet ($m$) tetszőlegesen paraméterezhetjük, azonban figyelembe kell vennünk, hogy az adott állapotot annak elhelyezkedésétől függően maximálisan hány megelőző állapot adataival simíthatjuk össze. A szóba jövő megelőző állapotok száma a

$$w_i = \frac{1}{min\{min(i,j), m\} + 1} \tag{3}$$

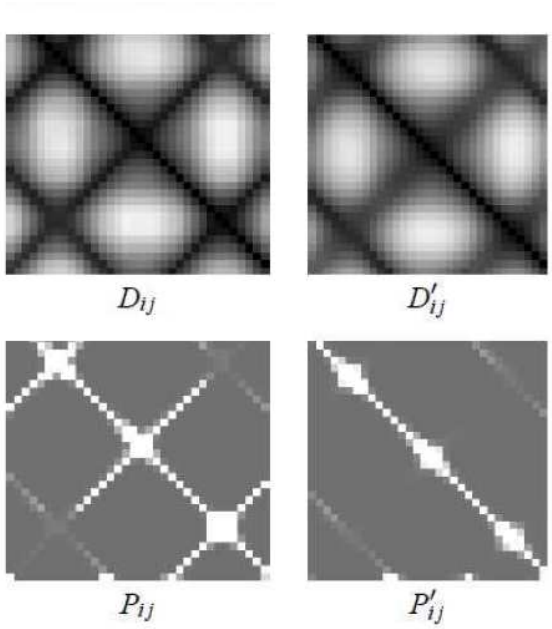
összefüggés alapján számolható. A művelet során a mátrixban történt változások vizualizációja megtekinthető a 8. ábrán[5].

A $D''_{ij}$ mátrix összeállításának módját a következő fejezetrészben részletezzük. Az egyes átmenetek valószínűségét két előzetes megfontolás alapján határozzuk meg: kövessük nagy valószínűséggel a mintaként kapott referenciagörbe trajektóriáját, de emellett adjuk meg a lehetőséget az arról való letérésre is - ennek valószínűsége pedig exponenciális karakterisztika használata mellett legyen arányos az eltérés mértékének reciprokával. A $\sigma$ paraméter használatával finomhangolható a kapcsolat az állapotok közötti függvényérték-távolság és az átmenet bekövetkezésének relatív valószínűsége között. A paraméter értékét úgy kapjuk meg, ha a $D''_{ij}$ mátrixban található 0 és $\infty$ közé eső elemek átlagát egy kellően alacsony, választott értékkel szorozzuk (a referenciaimplementációnkban 0.01-et használtunk). Alacsonyabb $\sigma$ paraméter mellett az algoritmus főleg a referenciagörbe ideális haladási irányát veszi figyelembe, nagyobbra véve pedig a várható átmenettől eltérő állapotátmenetek valószínűségei növelhetők:

$$P_{ij} \propto \exp(-D''_{i+1,j}/\sigma). \tag{4}$$

[5] A csatolt kép forrása: [10]

A görbegeneráláshoz kézenfekvő a sztochasztikus valószínűségi mátrixok alkalmazása, ehhez $P_{ij}$-t soronként normalizáljuk: $\forall i : \sum_{i=0}^{n} P_{ij} = 1$.




8. ábra. A megelőző állapotok tulajdonságainak figyelembevételének előtti és utáni állapotai (3), illetve a belőlük készített valószínűségi mátrixok.

## 2.3. Q-tanulás

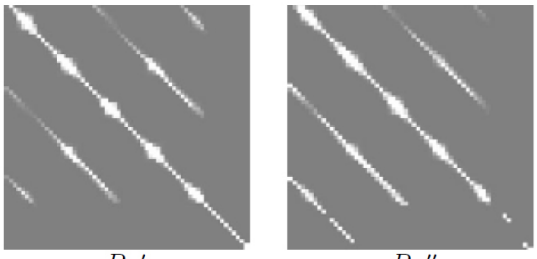
Az előző fejezetben előkészített valószínűségi mátrix önmagában nem képes végtelen hosszú ecsetmintákat generálni - a domináns útvonalak az eredeti ecsetmintában látható vonalvezetést követik, így a szintézis folyamán biztosra vehető, hogy hamar elérjük a görbe végállapotát, ahol zsákutcába kerülünk.

Ennél sokkal jobb eredményeket érhetünk el az útvonal előzetes megtervezésével, ha a várhatóan végállapotba vezető átmeneteket plusz költségekkel büntetjük. A probléma megoldásához a mesterséges intelligencia témakörében jól ismert megerősítéses- vagy Q-tanulás [5] (reinforcement learning, Q-learning) módszerét alkalmazzuk: a tanulás során módosított súlyértékkel ellátott $D''_{ij}$-ből áll elő

végső valószínűségi mátrix. A módszer alkalmazásához szükség lesz a költség fogalom olyan definíciójára, ami erre a problémára vonatkoztatható. Tekintsük a költséget úgy, mint a két tetszőleges állapot közötti függvényérték-különbség abszolútértéke, ami a kevésbé finom, rosszabbnak tekinthető átmeneteket kisebb, a jobb, finomabb átmeneteket pedig nagyobb szerephez juttatja, és a bekövetkezésük relatív valószínűségét is ennek alapján állapítja meg(9. ábra). Figyelembe véve, hogy a módszer $p \geq 1, \alpha < 1$ határfeltételek mellett tetszőleges mátrixra konvergens eredményt ad, $D''_{ij} \leftarrow D'_{ij}$ inicializálást követően a

$$D''_{ij} \leftarrow (D'_{ij})^p + \alpha \min_k D''_{jk} \tag{5}$$

egyenlet iteratív alkalmazásával véges számú lépésben megoldhatjuk a feladatot[6].

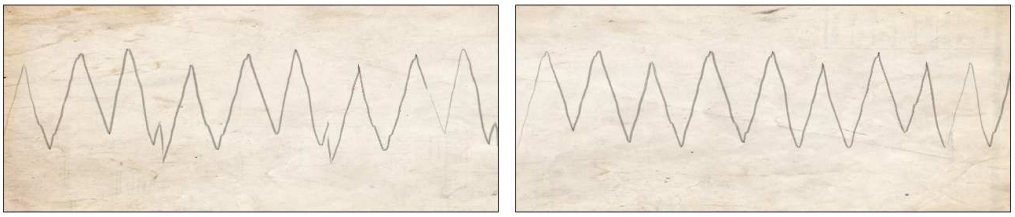



9. ábra. Az eredeti és a tanulás alkalmazása utáni valószínűségi mátrixok vizualizációi. Az új mátrix nem éri el a szekvencia végét, még időben egy korábbi állapotba lép vissza, így nem kerül zsákutcába.

# 3. A görbeszintézis menete

Kezdetben rendelkezésünkre áll a művész által készített ecsetminta, illetve egy görbék sorozatából álló geometriai modell, amelyre alkalmazni szeretnénk a művész ecsetvonásainak jellegzetességeit. A görbeszintézis teljes folyamata három lépésben történik.

[6] A csatolt kép forrása: [10]



10. ábra. Ecsetminta ismétlése. A képen egy olyan mintával dolgoztunk, amelynek az első és utolsó pontja között jelentős magasságérték-különbség van. *Bal* oldalt a naiv megoldást alkalmaztuk, ahol a mintát egymás után vágva ismételgetjük azt, így az illesztési pontokban jól látható a periódusonkénti "ugrás". A *jobb* oldali képen Q-tanulást alkalmaztunk, ami az ecsetminta pályájának előzetes elemzésével lezárja a várhatóan előnytelen állapotokba vezető utakat, így a fenti ugráshoz hasonlatos elégtelenségeket is javítja. Akár végtelen hosszú ecsetmintát generálhatunk anélkül, hogy az eredeti minta végére érnénk, mindezt úgy, hogy soha ne legyen két pontosan ugyanolyan ecsetvonásunk.

## 3.1. Stílusjegyek elemzése

A művészi ecsetvonások dinamikáját egy ecsetmintából dolgozzuk fel, amelyet bittérképes fájlformátumban tárolunk. A minta elemzését megelőzően szükséges volna azt olyan formára hozni, amely megfelel a matematikai függvény definíciójának. Egy szabadformájú ecsetminta esetén egy tetszőleges $x$ értékhez gyakran több $y_i$ függvényérték tartozik - gondoljunk csak bele: minden lefestett vonal valamilyen vastagsággal rendelkezik, így általánosságban semmilyen művészi alkotás vonalai nem tekinthetők függvényszerűnek. Figyelembe véve, hogy a feladatunk a minta haladási irányának meghatározása, megoldhatjuk a problémát úgy, hogy minden egyes $x$ értékhez a hozzá tartozó összes $y_i$ függvényérték helyett ugyanezen értékek tömegközéppontját, vagy súlypontját rendeljük. A fizika a részecskerendszerek tömegközéppontját az egyes részek tömegével súlyozott helyvektorának átlagolásával definiálja:

$$m_{tkp} = \frac{\sum_{i=0}^{n} m_i \, r_i}{\sum_{i=0}^{n} m_i}. \tag{6}$$

A mi esetünkben az $r_i$ helyvektort a $y_i$ függvényértékkel, annak $m_i$ súlyát pedig az adott képpont színintenzitásával ($g_i$) helyettesíthetjük, így a

$$y = \frac{\sum_{i=0}^{n} y_i \, g_i}{\sum_{i=0}^{n} g_i} \tag{7}$$

összefüggést alkalmazva egy függvény formájában kapjuk meg az ecsetminta dinamikáját.

## 3.2. Valószínűségi mátrixok felállítása

A minta elemzésének során a művész által készített ecsetvonások tulajdonságait tartalmazó valószínűségi mátrixot állítunk elő, illetve Q-tanulást alkalmazunk a végállapot elérése, és a látható periodicitás elkerülésére. A módszer részleteit a 2.2 és 2.3 fejezetekben taglaljuk.

## 3.3. Mintagörbék torzítása

A görbeszintézis során eltolás-értékeket (offseteket) hozunk létre, amelyekkel a meglévő számítógépes modellek részeit adó görbéket torzítjuk - az eredeti és az algoritmus által módosított kimeneti görbe is egy-egy Catmull-Rom spline lesz.

A valószínűségi mátrix használatával a görbe egy új pontját állítjuk elő: egy tetszőleges $k$ állapotból meghatározhatjuk, hogy mely $k+1$ állapotba kerüljünk, csupán annyi a feladatunk, hogy (0,1) tartományon egy véletlen számot generálunk, és azt a valószínűségi mátrix $k$-adik sorával vetjük össze. Egy $[P_{k1}, P_{k2}, \ldots, P_{kn}]$ sorvektort kaptunk, ahol a vektor $kn$-edik eleme pontosan a $k$ állapotból $n$ állapotba való jutás valószínűségét jelenti. Mivel a 2.2 fejezetben taglaltak alapján összeállított sztochasztikus valószínűségi mátrixról beszélünk, így abból bármely, tetszőlegesen kiragadott sorvektor elemenkénti összege 1.
Az algoritmus használatával készített valószínűségi mátrixokban szereplő

valószínűség-értékek eloszlása a véletlen számok generálásakor aggodalomra adhat okot: a főátlóban szereplő elemek reprezentálják a legkedvezőbb átmeneteket, így ezen értékek a $\sigma$ paramétertől függően gyakran 0.99 és 0.999 közöttiek is lehetnek. Gondoljunk csak bele, hogy egy [1500 × 1500]-as mátrix esetén egyetlen valószínűség a teljes intervallum 99%-át teszi ki, míg a többi 1499 érték a tartomány maradék 1%-án „nyomorog" (9. ábra). A probléma feloldásának egy lehetséges módja egy megfelelő pontosságú véletlenszám-generátor használata - saját implementációnkban a Mersenne Twister algoritmus használatával 53-bites pontossággal dolgoztunk [6].

Már csak azt kell kitalálnunk, hogy az eredményül kapott offseteket milyen módon tudjuk a már elkészült modellekre alkalmazni. Ahhoz, hogy a torzítandó görbe középvonalához tudjuk az eltolás-értékeket hozzáadni, egy olyan koordinátarendszerre volna szükség, amely képes követni a görbe haladási irányát. Az TNB (tangens-normális-binormális) koordinátarendszer pontosan erre a problémára jelent megoldást: a tangens ($\vec{T}$) tengely a mozgásirányba mutat, az északi tengely a görbe megadott pontjában előállított normálvektor irányába néz ($\vec{N}$), a binormális ($\vec{B}$) pedig az előző két vektorra lesz merőleges. A koordinátarendszer tengelyei a

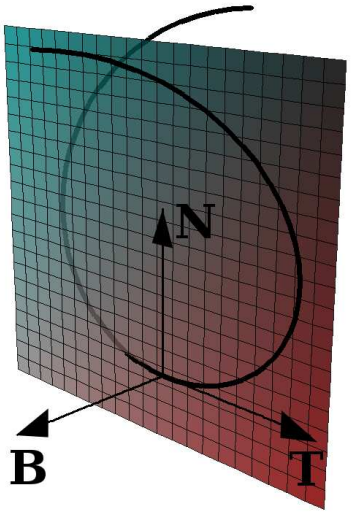
$$\vec{T} = \frac{d\vec{r}}{ds}, \qquad \vec{N} = \frac{\frac{d\vec{T}}{ds}}{\left|\frac{d\vec{T}}{ds}\right|}, \qquad \vec{B} = \vec{T} \times \vec{N} \tag{8}$$

a összefüggések alapján állíthatók elő[7].

## 3.4. Parametrizálás

Ebben a fejezetrészletben az algoritmus működésének módosíthatóságáról, paraméterezésének lehetőségeiről tárgyalunk. A valószínűségi szemlélet alkalmazásának egy további előnyeként fogalmazható meg, hogy az egyes

[7] A kép forrása: http://en.wikipedia.org/wiki/TNB_frame




11. ábra. A TNB koordinátarendszer szemléletes ábrázolása. A tengelyek a görbe tetszőleges pontjában való előállításakor a fejezetben tárgyalt irányokba néznek, így elérhetjük, hogy kövesse a spline útját.

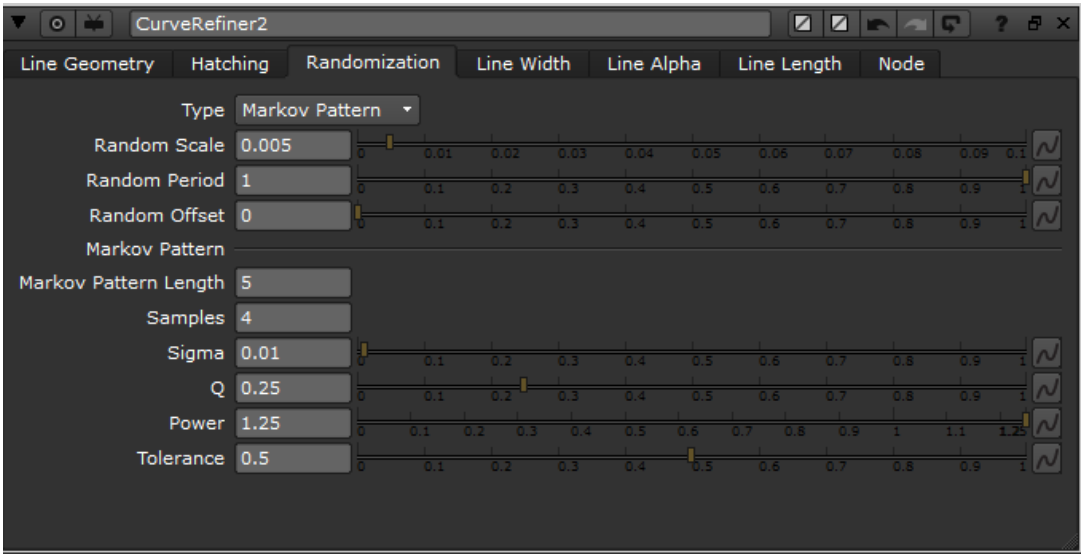
állapotátmeneti valószínűségek megfelelő súlyozásával akár az eredeti görbét is visszakaphatjuk, illetve a torzítás mértékét is tetszőleges nagyságúra vehetjük - a finom, művészi célú alkalmazáson túlmenően akár erőteljes, firkálmány jellegű hatás is kelthető. A korábban ismertetett egyenletek egyes változói tehát tetszőlegesen paraméterezhetők, emellett az eredményesebb, kényelmesebb munkát elősegítő lehetőségeket is bemutatjuk (12. ábra).

A paraméterváltozók nevei, illetve azok szemléletes jelentései következnek.

**Random scale**. a beállítás használatával súlyozható, hogy az eredeti modell kontúrjai és a Markov-mezős torzítás milyen arányban vegyenek részt a végső ecsetminta kialakításában

**Random period**. a megrajzolt ecsetmintából generált görbék sűrűsége, azaz hogy pontosan hány periódust kívánunk felmérni a modellre. Megjegyzendő, hogy a kimeneti görbét nem jellemzi periodicitás, a lehetőség az új kontrollpontok sűrűségének választhatóságára utal.

**Random offset**. megadható, hogy pontosan hányadik állapotból kezdjük a




12. ábra. A program használata a filmstúdióban használt grafikai rendszerben.

görbeszintézist.

**Markov pattern length**. az algoritmus a megadott paraméterrel arányos mennyiségű kontrollpontot készít (annak az ezerszeresét).

**Samples**. A figyelembe vett megelőző állapotok maximális száma. A 3. egyenlet $m$ paramétere állítható be vele.

**Sigma**. A valószínűségi mátrix összeállításakor (4. egyenlet) használt $\sigma$ paraméter értéke. A paraméter szemléletes jelentése az úgynevezett "vállalkozói kedv", azaz az ideális állapotátmenetektől való eltérés valószínűsége. Az érték növelésével a kisebb valószínűséggel bekövetkező állapotátmenetek relatív valószínűsége növelhető. Amennyiben a $\sigma$ értékét infinitezimálisan kicsi értékkel illetjük, az inverz exponenciális karakterisztika miatt a kis valószínűségek ereje zérusnak tekinthető, így a referenciagörbét, azaz az eredetileg megadott ecsetmintát kapjuk vissza. Nagyobb $\sigma$ esetén az algoritmus bátrabban eltér az eredeti ecsetmintától.

**Q**. A Q-tanulás (5. egyenlet) kiértékelésekor felhasznált $\alpha$ paraméter.

Matematikailag a konvergencia sebességét határozza meg, az 1-hez közeli értékekre több Q-iteráció fut, az annál alacsonyabb értékekre pedig kevesebb út kerül súlyozásra. Szemléletes jelentés: "a tanulás mértéke", illetve a paraméter megmondja, hogy az algoritmus "mennyire harciasan küzdjön az előnytelenebbnek tartott állapotok megközelítése ellen".

**Power**. A Q-tanulás képletében (5. egyenlet) a $D'_{ij}$ mátrix kezdőértékeit $p$. hatványkitevőre szintén az algoritmus vállalkozói kedvét módosítja.

**Tolerance**. A Q-tanulás teljesítményelemzésénél (5.1. fejezet) a gyorsabb végrahajtási idők elérése érdekében a konvergencia fogalmát annak matematikai pontosságától ennyivel lazábban definiáljuk újra.

## 3.5. Példakalkuláció

Kövessük végig az algoritmus lefutását egy egyszerű, rövid ecsetminta feldolgozásán keresztül! A szemléletesség kedvéért a példa során a számításokat lényegesen leegyszerűsítettük.
Tekintsünk egy ecsetmintát, amely a függvényszerű leképzés után az

$$\{\, 5,\ 10,\ 15,\ 11,\ 9,\ 20 \,\} \tag{9}$$

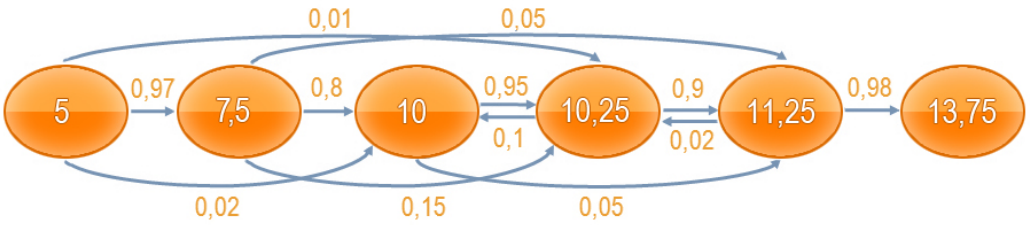
diszkrét értékek sorozatával jellemezhető. Az $m = 4$ paraméter alkalmazása mellett egy 4 nagyságú ablakot alkalmazunk, azaz minden állapot esetén az azt megelőző, pontosan 4 korábbi állapot tulajdonságait vesszük figyelembe. A simítás után a valószínűségi mátrix előállításához a továbbiakban a

$$\{\, 5,\ 7.5,\ 10,\ 10.25,\ 11.25,\ 13.75 \,\} \tag{10}$$

értékekkel folytatjuk a munkát.

A következő lépésben az újonnan számított állapothalmazon egy olyan

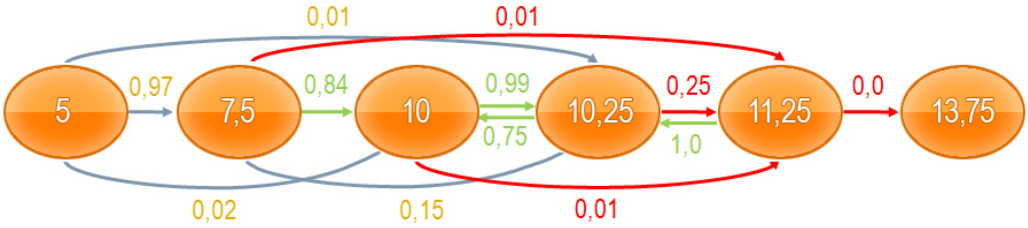
sztochasztikus folyamatot definiálunk, amely nagy valószínűséggel az eredeti minta trajektóriáját követi, de a soron következő állapottól való távolság inverz exponenciális arányában további átmeneteket is megengedünk (4. egyenlet). A 10 állapot után az eredeti minta szerint nagy valószínűséggel a 10.25 állapotba következik, ám az említett karakterisztika szerint egy csekélyebb valószínűséggel megengedjük a 10.25-höz közeli 11.25 állapotba való átlépést is. A valószínűségi mátrix $(i, j)$. eleme az $i$. állapotból a $j$. állatpoba való átmeneti valószínűséget jelenti, azaz minden állomásról mindegyik másik állomásra való átlépési valószínűséget felírunk, a túlságosan kedvezőtlen, távoli lépésekhez az algoritmus 0-hoz közeli, infinitezimális nagyságú valószínűséget fog rendelni (13. ábra).




13. ábra. A Markov-mezős sztochasztikus modell felírása.

A Q-tanulás alkalmazása megelőzi akár több lépésen keresztül is képes észrevenni a zsákutcába való jutás esélyét, így az ehhez vezető állapotátmeneteket plusz költségekkel bünteti - ezzel párhuzamosan pedig természetesen az előnyösebb helyzeteket előidéző állapotátmenetekhez nagyobb valószínűséget rendelünk. A 11.25 állapotból rögtön, egyetlen lépésben a végállapotba kerülnénk, így ezt az átmenetet teljesen megtiltjuk - a kedvezőbb, visszafele lépés relatív valószínűségét pedig ennek megfelelően növeljük. A 10. állapotból egyetlen lépésben eljuthatunk a végállomást megelőző 11.25 állapotba, így ez az átmenet is büntetésben részesül.

A példalevezetés során a pontos numerikus kiértékelés helyett közelítő értékekkel dolgoztunk, amelyek feladata az egzakt értékek közlése helyett algoritmus szemléletmódjának bemutatása (14. ábra). Az egyszerűség kedvéért továbbá az ecsetmintát egy 6 állapotú Markov-folyamatra képeztük le, ellenben minden




14. ábra. Ugyanezen modell a Q-tanulás egy iterációját követően.

gyakorlati, valós ipari alkalmazás esetén egy viszonylag rövid ecsetmintát is legalább 4-500 állapottal jellemzünk.

# 4. Újdonságok

A filmstúdióban dolgozó művészek eredményes munkájához olyan grafikai programokra van szükség, amelyek úgy képesek megközelíteni a végső termék képminőségét, hogy közben valós időben láthatóvá teszik az adott paraméterváltoztatás szintérre gyakorolt hatását. A művészek órabére lényegesen magasabb az erőforrások bérlésének költségénél, így nem megengedhető, hogy a dolgozó hosszú perceket várakozzon a legutolsó változtatásainak eredményeire.

Az algoritmus lefuttatásának legszámításigényesebb részéhez, a Q-tanulás kiértékelésének folyamatához egy olyan új lehetőséget vezettünk be, amellyel lehetővé válik annak valós időben való alkalmazása. Az egyes Q-iterációk során vizsgáljuk, hogy az adott iterációs lépés a valószínűségi mátrixban mekkora eltérést jelent az azt megelőzőhöz képest. Amennyiben ez az érték egy előre meghatározott, paraméterezhető toleranciaszint alá csökken, nem folytatjuk tovább a tanulást a következő iterációval, hanem a kapott eredményt úgy tekintjük, mintha a végső konvergencia állapotot értük volna el. A tanulás toleranciaszintjét magasabbra állítva jelentősen rövidebb számítási idő alatt kaphatjuk meg a végeredmény közelítését (5.1), így kényelmesen, akár azonnal szemrevételezhetjük, hogy a

legutolsó változtatásaink milyen hatással voltak a görbére. A végső, produkciós render pipeline meghívásakor kellően alacsony toleranciaszint beállításával nemcsak pontosabb eredményekhez jutunk, de a költségvetés függvényében megtervezhetjük, hogy mennyi időt és erőforrást kívánunk rászánni a Markov mezős algoritmus kiértékelésére.

Arno Schödl videotextúrákkal kapcsolatos kutatási eredményeit [8–10] felhasználva Robert D. Kalnins cikkében egy módszert ismertet a valószínűségi mátrix használatával történő splineok generálására [3], ami állóképek torzítására alkalmazható. A megközelítéshez animációs megfontolásokat tartalmazó kiegészítéseket javaslunk. A szintézis során generált pontok valamilyen adatszerkezetben való tárolásán túlmenően tekintettel kell lenni a sziluettek időbeli megváltozására is. Gondoljunk csak egy tetszőleges alakzatra, amit éppen a térben forgatunk: a körvonalak bizonyos része eltűnik, a test felénk forduló oldalán pedig új, korábban takarásban levő részletek válnak megfigyelhetővé. Ezt az időbeli változást két dimenzióban tekinthetjük a sziluettek görbéinek rövidülésének és hosszabbodásának. A görbepontok tárolásával és egy kisebb részhalmazuk újrafelhasználásával megvalósítható a rövidülés kiszámítása a következő képkockára. A görbe meghosszabbításához azonban tudnunk kell folytatni is azt, ezt azonban az adott állapotokhoz tartozó függvényértékek tárolásával nem tehetjük meg - a megoldásunkban ehelyett az általunk generált görbe állapotait tároljuk, megjelenítéskor pedig az eredeti ecsetmintában hozzájuk tartozó offset-értékeket keressük ki. Az említett adatszerkezet alkalmazásával lehetővé válik az algoritmus mozgóképekre, animációkra való alkalmazása is.

Az ecsetminta meghosszabbításakor újra a valószínűségi mátrixhoz kell nyúlunk, aminek beolvasási ideje az eredeti minta hosszától függően akár néhány másodpercig is terjedhet. A beolvasás után az újabb pontok előállítása akár százezres nagyságrendben is elhanyagolható idő alatt történik. Az algoritmus "burn in" idejét

megspórolhatjuk, ha minden görbéhez szándékosan több pontot állítunk elő, mint amennyire terveink szerint szükségünk lehet. Alig mérhető teljesítménycsökkenés mellett elkészíthetjük rögtön a maximális becsült mintaméretet, amire csak szükségünk lehet a görbe animálása során, így az adott görbével való munka során már biztosan nem kell többet a valószínűségi mátrixhoz nyúlnunk. Az ecsetminta hosszabbításának igénye esetén azonnal felhasználhatjuk a már előzetesen legenerált "tartalékot".

Egy tetszőleges állapot matematikai reprezentációjában a módszer egy valamely $m$ paraméter függvényében (átlagoló ablakméret) a megelőző állapotok tulajdonságait is figyelembe veszi, a megelőző állapotok száma azonban nem feltétlenül éri el ezt az értéket (ez különösen nagy $m$-ek esetén igaz). Az eredeti módszert tartalmazó cikkek egyike sem veszi figyelembe ezt a megszorítást: az $w_i = 1/m$ súlyozás alkalmazása helyett a (3) egyenletben szereplő megoldást javasoltuk, amely a lehetséges megelőző állapotok számát is figyelembe veszi az átlagolás során, így tetszőlegesen nagy $m$-ekre lehetővé válik az eredeti görbe stílusjegyeinek hiteles modellezése. Ezzel nem csak hogy a valósághoz közelibb, de egy általánosabb megoldást is kaptunk.

# 5. Eredmények, minták

## 5.1. Q-tanulás teljesítmény

A fejezetben a toleranciaszintek bevezetésével elérhető teljesítménynövekedést mérésekkel igazoljuk. Tekintsük az alábbi kódrészletet, amely az algoritmus Markov-mezős fázisának alkalmazásakor annak sikeres lefutásának körülményeit írja le:

```
The output matrix size will be 837x837.
1st pass - computing the Dij matrix...
2nd pass - computing the Dij' matrix...
```

```
3rd pass - computing the Dij'' matrix with Q-learning...
Convergence reached. Difference: 0.488262, tolerance: 0.5.
Q iteration count: 264
4th pass - computing the Pij matrix...

Maximum magnitude: 111.411
Magnitude sum: 1.6268e+007
Magnitude sigma_entries: 694059
Sigma naturally: 23.4389
Sigma multiplied by 0.03: 0.703168

4th pass cont'd - normalization of Pij...
Markov field phase processing time was 17.131s.
```

A lefutás eredménye alapján tehát egy [837 × 837]-es valószínűségi mátrixban elemezzük a művész ecsetvonásait. A toletanciaszintet 0.01-re vettük: a Q-tanulás alkalmazásakor ha $k$. és $k + 1$. iterációk között a mátrixban történt összes értékmódosítás abszolútértéke 0.01 alá csökken, akkor nem iterálunk tovább, úgy vesszük, hogy elértük a konvergencia állapotát.

A mért adatok alapján a Q-tanulás szempontjából szóba jöhető módosítandó értékek 0 és $\infty$ közé esnek - ezeknek a száma a teljes mátrixban: 694000, átlagértékük pedig 23.43. Az összes szóba jöhető elem értékösszegének átlagos értéke ez alapján körülbelül 16.200.000 - a jelenlegi toleranciaszinttel ezt a konvergencia elérésének állapotában 0.5-es pontossággal közelítjük meg. A mért végrehajtási idő 264 Q-iteráció után: **17.131s**.

Tekintve, hogy egy ipari animációs mozifilm teljes renderelési csővezetékében rengeteg egyéb számításigényes algoritmus is helyet kap, ilyen várakozási idők mellett teljesen reménytelen érdemi munkát végezni.

Miközben a filmstúdióban a jelenetek megtervezésén és finomhangolásán dolgozunk, nincs időnk minden ötletünk megjelenésére fél perceket várakozni: szeretnénk *valós időben* látni a változtatásaink eredményét, így tegyük fel, hogy nagyvonalúan kiegyezünk a tökéletesen pontos megoldástól egy maximálisan 0.01%-ban eltérő, közelítő megoldással is. Nézzük meg, hogy mekkora sebességnövekedésre számíthatunk, ha a toleranciaértéket a mátrixban szereplő elemek összegének(*Magnitude sum*) 0.01%-ával állapítjuk meg!

```
The output matrix size will be 837x837.
1st pass - computing the Dij matrix...
2nd pass - computing the Dij' matrix...
3rd pass - computing the Dij'' matrix with Q-learning...
Convergence reached. Difference: 1570.75, tolerance: 1620.
Q iteration count: 55
4th pass - computing the Pij matrix...

Maximum magnitude: 111.411
Magnitude sum: 1.62893e+007
Magnitude sigma_entries: 693795
Sigma naturally: 23.4786
Sigma multiplied by 0.03: 0.704358

4th pass cont'd - normalization of Pij...
Markov field phase processing time was 3.619s.
```

A mérési eredmények alapján drasztikus teljesítménycsökkenést tapasztaltunk: elegendő volt 2 Q-iteráció ahhoz, hogy az általunk lazábban újradefiniált konvergencia fogalmát elérjük. A mért végrehajtási idő 55 iteráció után: **3.619s**.

A mátrix tanuláskor szóba jövő elemeinek számát (*Magnitude sigma_entries*) az ezen elemek átlagértékével (*Sigma naturally*) súlyozva összevetjük a két mérés kimeneti eredményeit: valóban 0.01%-os hibával tértünk el a tökéletes megoldástól - cserébe közel *4.73-szor* olyan gyorsan elkészültünk a számításokkal! Mindez papíron jól hangzik, azonban szükséges volna a gyakorlatban is megtekinteni a két módszer kimenete közötti különbségeket. A Mersenne-Twister véletlenszám-generáló algoritmussal álvéletlen sorozatok előállításával garantálható az algoritmus determinisztikus lefutása - szabad szemmel semmilyen különbség nem volt tapasztalható.

Mindezt az egyszerűsítést azért tehetjük meg, mert a $k$. Q-iteráció során elemenként $\alpha^k$ nagyságrenddel arányos változásokat várhatunk a valószínűségi mátrixban (a 0 és $\infty$ költségű állapotátmenetek kivételével), így abban kellően sok iteráció után igen csekély változások mennek végbe. Rengeteg számítást megspórolhatunk, ha a köznapi munkálataink során a konvergencia lazább definícióját alkalmazzuk.

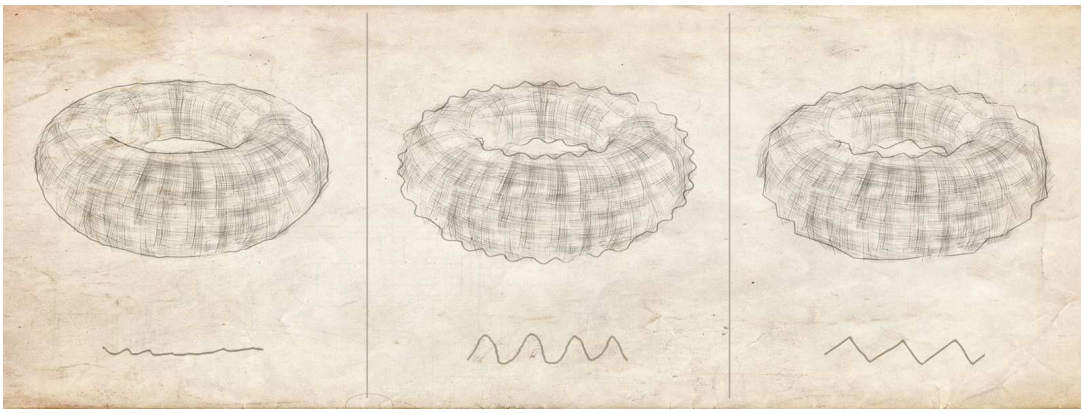
*Megjegyzés: jelen mérési eredmények egy olyan mérési elrendezésre vonatkoznak, melynek számításigénye messzemenően meghaladja még a legszélsőségesebb ipari alkalmazás során előfordult helyzetekét is. Az új módszer gyakorlati, filmstúdióban való felhasználásai során század- és ezredmásodperces nagyságrendű végrehajtási időket mértünk.*

## 5.2. Gyakorlati és ipari alkalmazások

Az előző fejezetekben az algoritmus matematikai tulajdonságairól értekeztünk, a továbbiakban annak gyakorlati használhatóságát a legkülönbözőbb környezetben való alkalmazási példákkal illusztráljuk.

Kezdetben szeretnénk megbizonyosodni arról, hogy egy kellően jellegzetes ecsetminta bevitelét követően a megoldásunk ténylegesen ahhoz hasonlót generál. A

torzított kép sziluettjeit az eredeti ecsetminta jellegzetességei dominálják, emellett rendelkezik azzal a kellemes tulajdonsággal, hogy semmilyen szemmel látható periodicitás nem figyelhető meg rajta.



15. ábra. Tórusz kép különböző deformációi a lent látható ecsetminták alapján. A torzított képen Markov-mezővel elemeztük és variáltuk az ecsetmintában megfigyelt stílusjegyeket, a Q-tanulás alkalmazása pedig a szemmel látható periodicitás kiiktatásával természetes, valódi kézi rajz illúzióját tárja elénk.

A következő rajzon a módszert egy összetettebb geometriájú alakzatra alkalmazzuk. A gyakorlott művészek friss, kezdetleges ötleteiket jórészt piszkozatok és vázlatrajzok segítségével kezdik el kidolgozni. A megfelelő tónusok és formák megtalálásáig rengeteg elhibázott próbálkozáson át vezet az út. A következő kép készítésekor ezt a tényezőt is figyelembe véve a fő körvonalak mellett vékonyabb vastagságú, kevésbé hangsúlyos kísérővonalakat is alkalmaztunk (16. ábra).

16. ábra. Torzítás alkalmazása egy összetettebb geometriájú rajzra. Helyenként egy helyett többszörös kontúrvonalakat használtunk, különböző vonalvastagsággal, a rajz így egy művészi skicc impresszióját kelti.

A már meglévő piaci megoldások és kutatási eredmények továbbfejlesztését elsősorban az ipar hívó szava motiválta: a dolgozatban bemutatott algoritmus a *Lichthof Productions Ltd.*[8] filmstúdió megrendelésére készült, és fontos szerepet kapott a cég legújabb animációs mozifilmje, az *Egill: The Last Pagan*[9] poszt-produkciós munkálatai során. Alapvető fontosságú szempont, hogy a megoldás az egyszerű kézirajzos példákon kívül éles ipari környezetben is alkalmazható legyen, ez a feltétel pedig igen szigorú minőségi és teljesítménybeli elvárásokat jelent.

Az algoritmus már a filmstúdió teljes grafikai rendszerének többi elemével összehangoltan működik, így lehetőség nyílt a csapat művészeivel való közös munka során a film jeleneteire alkalmazni azt. Az eredményeket példákkal illusztráljuk.

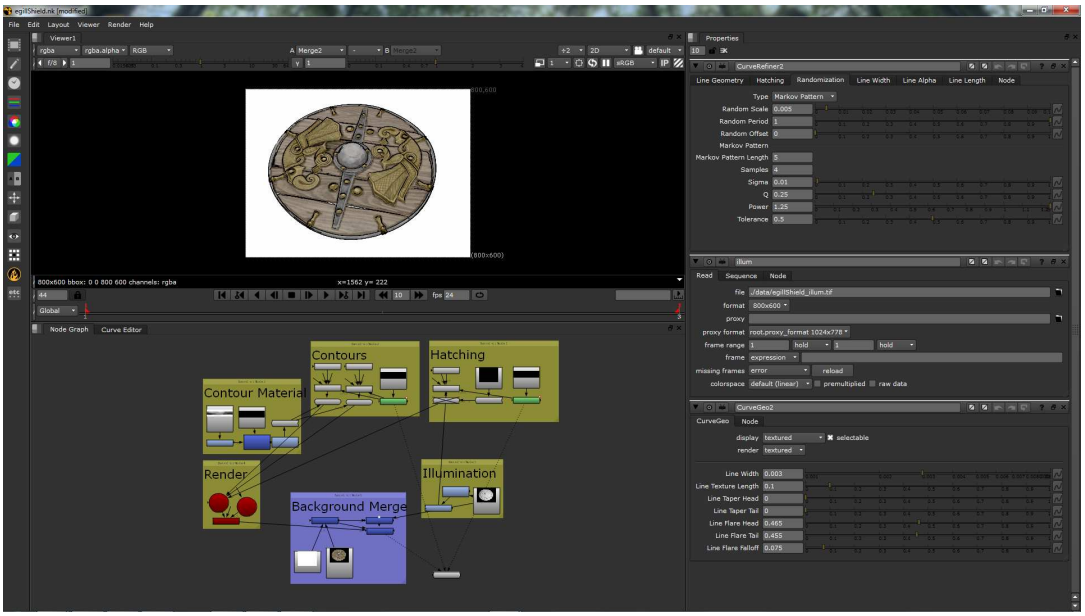
17. ábra. Egill pajzsának modellje az *Egill: The Last Pagan* c. filmből.

[8] http://www.imdb.com/company/co0164894/
[9] http://www.imdb.com/title/tt1492806/

18. ábra. Egill Skallagrímsson, az *Egill: The Last Pagan* c. film főszereplője. Egill karaktere egyszerre kíméletlen barbár és érzékeny költő, aki furcsa betegsége miatt nem állt meg a növésben, egyre hatalmasabbá vált, s időközben igencsak meg is csúnyult. Sajnos ezen a problémán még a Markov-mezőkkel sem tudtunk segíteni.

# 6. Összegzés, további munka

Egy olyan módszert mutattunk be, amely festőművészek által készített ecsetminták stílusjegyeinek elemzését követően annak jellegzetességeit meglévő képekre, jelenetekre alkalmazza. Az eredményül kapott képek mentesek a számítógépes modellezőprogramok által készített modellek geometriáit jellemző "sterilitástól", és azt az illúziót keltik, hogy a jelenet valóban egy festőművész munkájának eredményeként jött létre. A témakörben már meglévő kutatási eredmények vívmányait használtuk fel, azok gyengeségeit kijavítva továbbgondoltuk azokat: olyan adatszerkezetet és tárolási módszert ajánlottunk, amellyel lehetséges




19. ábra. A program felhasználói felülete munka közben.

az algoritmus mozgóképekre való alkalmazása. A művész ecsetmintáinak elemzése során a megoldás pontossága szabályozható, így a torzított képek akár valós időben is előállíthatók - a filmstúdióban dolgozó művész folyamatosan figyelemmel kísérheti, hogy a legutolsó módosítása milyen hatást gyakorol az éppen szerkesztendő jelenetre. Ezen kívül változtatásokat javaslunk a megelőző állapotok hatékonyabb figyelembevételére is. Az új algoritmus az eddig született kutatási eredményekhez képest olyan fejlődéseket mutat fel, amely lehetővé teszi a módszer ipari környezetben való felhasználását, és egy animációs mozifilm elkészítésének folyamatában is fontos szerepet kapott. A módszer ilyen irányú továbbfejlesztése és éles ipari alkalmazása még nemzetközi szinten is példátlan újdonságnak számít.

A Markov mezős szintézis jelenleg is olyan eredményeket tudhat magáénak, amelyet a számítógépes grafika egyéb területein alkalmazott módszerekkel csak nehezen, komoly megszorításokkal lehet reprodukálni. A jövőben a módszer további lehetséges alkalmazási lehetőségeit látjuk annak többdimenziós függvényekre és magasságmezőkre való kiterjesztésében, ami izgalmas új alternatívát jelentene mintából való pixel-alapú textúraszintézis, illetve tetszőlegesen bonyolult,

többdimenziós struktúrákba szervezett tulajdonságok alapján való mintagenerálás világában.

# Hivatkozások

[1] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. Proceedings of SIGGRAPH 2001, pages 341–346, August 2001.

[2] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In IEEE International Conference on Computer Vision, pages 1033–1038, Corfu, Greece, September 1999.

[3] Robert D. Kalnins, Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Joseph C. Lee, Philip L. Davidson, Matthew Webb, John F. Hughes, and Adam Finkelstein. WYSIWYG NPR: Drawing strokes directly on 3D models. ACM Transactions on Graphics (Proc. SIGGRAPH), 21(3):755–762, July 2002.

[4] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. ACM Transactions on Graphics, SIGGRAPH 2003, 22(3):277–286, July 2003.

[5] Andrew Moore Leslie Pack Kaelbling, Michael Littman. Reinforcement learning: A survey. Journal of Artificial Intelligence Research, 4:237–285, 1996.

[6] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul., 8(1):3–30, 1998.

[7] Ken Perlin. An image synthesizer. SIGGRAPH Comput. Graph., 19(3):287–296, 1985.

[8] Arno Schödl and Irfan Essa. Machine learning for video-based rendering. In In Advances in Neural Information Processing Systems, pages 1002–1008. MIT Press, 2000.

[9] Arno Schödl and Irfan A. Essa. Controlled animation of video sprites. In SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 121–127, New York, NY, USA, 2002. ACM.

[10] Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa. Video textures. In SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pages 489–498, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[11] L. Szirmay-Kalos (editor). Theory of Three Dimensional Computer Graphics. Akadémia Kiadó, Budapest, 1995. http://www.iit.bme.hu/~szirmay.

[12] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pages 479–488, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.