Hatékony megoldások lencsefényfoltok és szimulációs adatsorok vizualizációjára

Egyetemi doktori (PhD) értekezés

Csoba-Bodonyi Andrea Beatrix
Témavezető: Dr. Kunkli Roland Imre

DEBRECENI EGYETEM
Természettudományi és Műszaki Tudományi Doktori Tanács
Informatikai Tudományok Doktori Iskola
Debrecen, 2026.

Ezen értekezést a Debreceni Egyetem Természettudományi és Műszaki Tudományi Doktori Tanács Informatikai Tudományok Doktori Iskola Adattudomány és vizualizáció programja keretében készítettem a Debreceni Egyetem műszaki doktori (PhD) fokozatának elnyerése céljából.
Nyilatkozom arról, hogy a tézisekben leírt eredmények nem képezik más PhD disszertáció részét.

Debrecen, 2026. . . . . .

. . . . . .
Csoba-Bodonyi Andrea Beatrix
jelölt

Tanúsítom, hogy Csoba-Bodonyi Andrea Beatrix doktorjelölt 2019–2026 között a fent megnevezett Doktori Iskola Adattudomány és vizualizáció programjának keretében irányításommal végezte munkáját. Az értekezésben foglalt eredményekhez a jelölt önálló alkotó tevékenységével meghatározóan hozzájárult. Nyilatkozom továbbá arról, hogy a tézisekben leírt eredmények nem képezik más PhD disszertáció részét.
Az értekezés elfogadását javasolom.

Debrecen, 2026. . . . . .

. . . . . .
Dr. Kunkli Roland Imre
témavezető

Hatékony megoldások lencsefényfoltok és szimulációs adatsorok vizualizációjára

Értekezés a doktori (PhD) fokozat megszerzése érdekében
az informatika tudományágban

Írta: Csoba-Bodonyi Andrea Beatrix okleveles programtervező informatikus

Készült a Debreceni Egyetem
Informatikai Tudományok doktori iskolája
(*Adattudomány és vizualizáció* programja)
keretében

Témavezető: Dr. Kunkli Roland Imre

Az értekezés bírálói:

Dr. . . . . .
Dr. . . . . .
Dr. . . . . .

A bírálóbizottság:

elnök: Dr. . . . . .
tagok: Dr. . . . . .
Dr. . . . . .
Dr. . . . . .
Dr. . . . . .

Az értekezés védésének időpontja: 20. . . . . .

Tartalomjegyzék

1. Bevezetés

A számítógéppel létrehozott vizuális tartalmak előállítása már hosszú ideje a tudományos munkák fókuszába helyezte a minél valósághűbb megjelenést és a hatékonyságot. Mivel a cél a lehető legmagasabb hitelesség elérése, mindez közvetlenül vonja maga után a képalkotási folyamatok sajátosságainak elengedhetetlen figyelembevételét.

Akár emberi szem, akár kamerák által létrehozott képekről beszélünk, a képalkotási folyamat számos, a képalkotó rendszer jellemzőiből fakadó sajátossággal rendelkezik. Ezáltal a létrejövő kép is merőben eltér a különböző képalkotó rendszerek esetén. Sokszor a várt képen kívül egyéb kívánt vagy nem kívánt jelenségek is szerepet kapnak. Ahhoz, hogy egy valósághű kimenetet hozzunk létre, a képalkotási folyamat során létrejött kép ezen szereplőit is reprodukálni kell. Ezeknek a jelenségeknek a hű szimulációjához azonban elengedhetetlen az őket eredményező eszközök és folyamatok fizikai jellemzőinek ismerete és figyelembevétele a szimuláció során.

Munkánk nagy részében a képalkotási folyamat során keletkező lencsefényfoltokkal foglalkoztunk. Ezt a jelenséget a kameraobjektívekben, a fénysugarakat érintő folyamatok eredményezik. A létrejövő lencsefényfoltok sokszor nem kívánt kimenetek, ennek ellenére mégis hangsúlyos művészi eszközként tartják ezeket számon a fotográfia és filmográfia területén [1, 2], így a gyakorlatban is sokszor használatosak. Nem kívánt jellegük miatt azonban a lencsefényfoltok eltávolítása is sokszor kerül a tudományos munkák középpontjába [3, 4]. A szakirodalomban elérhető friss munkákban elsősorban gépi tanulási algoritmusokkal igyekeznek eltávolítani a jelenség következtében létrejövő fényfoltokat a valós kamerákkal készített képekről, ehhez azonban kritikus a tanulóadatok minősége és mennyisége. A tanító adathalmaz létrehozása történhet számítógéppel, így a szimulált lencsefényfoltok ezen a területen is szerepet kaphatnak.

Művészi területen történő felhasználásai miatt a jelenség valószerűsége iránt támasztott elvárás jelentős, azonban a tanulóadatok létrehozásához, majd a magas pontosságú tanuláshoz a szimulációs algoritmus fizikai megalapozott-

sága kritikus. A lencsefényfoltok valósidejű alkalmazásokban történő felhasználása miatt ugyanakkor elengedhetetlen a futási idő minimalizálása. A cél tehát a fizikai szempontból bonyolult képalkotó folyamat reprodukciója minél rövidebb idő alatt.

A már elérhető megközelítések nem teljesítik mindkét feltételt egyszerre, így munkánk fő célja egy olyan módszer létrehozása volt, ami képes a valósidejű, fizikailag hű lencsefényfolt-szimulációt megvalósítani. A releváns szakirodalomban korábbról elérhető módszerek közül kiemelt szerepe van Hullin és mtsai. [5, 6] két módszerének. Ők egyrészt egy ritka sugárrácsot követnek végig az optikai rendszeren a költséges sugárkövetés hatékonyabbá tételének érdekében [5], másrészt egy polinomiális módszert dolgoztak ki olyan általános célokra, amikor sugárkövetés végrehajtására lenne szükség [6]. Munkánk során kiindulási pontokként használtuk fel a fenti két módszert. A valósidejű lencsefényfolt-szimulációs eljárás létrehozását a folyamat két fő fázisán keresztül közelítettük meg. Egyrészt kritikus a sugárkövetés költségének minimalizációja, másrészt a kimeneti kép előállításához szükséges raszterizáció hatékonyabbá tétele. A sugárkövetéses fázisban tehát a polinomiális megközelítést alkalmaztuk a ritka sugárrácsokkal kombinálva. A raszterizációs fázis idejének csökkentése érdekében pedig egy csempézett megközelítést alkalmaztunk, ami más számítógépes grafikai területeken már sikeresen felhasználásra került [7, 8]. A lencsefényfoltokhoz kapcsolódó munkánkat, valamint a területen elért eredményeinket a 3. fejezetben prezentálom.

A képalkotó rendszerek sajátosságain felül munkánk másik részét egy kutatási együttműködés jelentette. A Debreceni Egyetem Pszichológiai Intézetének kutatói fogalmazták meg az igényt egy háromdimenziós megjelenítő keretrendszer iránt, ami a folyamatban lévő kutatásuk adatainak vizuális megjelenítésére képes, ezzel járulva hozzá az elemzés sikerességéhez.

A probléma forrását az ő kiindulási szimulációs algoritmusuk által generált adatok jellege képezte. Mivel kutatásukban egy mikroorganizmus viselkedése indirekt módon, az organizmus szemszögéből érzékelt környezetváltozással volt reprezentálva, így a generált adatok megjelenítéséhez azok transzformációjára volt szükség. Emiatt a keretrendszer megalkotásához egy olyan metódust hoztunk létre, amely képes ennek a problémának a hatékony feloldására. Er-

re a célra egy baricentrikus koordinátákon alapuló algoritmust alkottunk meg, amely képes a mikroorganizmus lokális koordináta-rendszerében a hozzá relatív módon definiált objektumok globális rendszerbe történő konverziójára. A területen végzett munkánkat, valamint elért eredményeinket a 4. fejezetben mutatom be.

2. Elméleti háttér

Az értekezésben bemutatott módszerek két, egymástól eltérő problématerület köré szerveződnek: a kamerarendszerekben létrejövő lencsefényfoltok fizikailag megalapozott hatékony megjelenítése, valamint olyan szimulációs adatsorok vizualizációja, ahol a vizsgált jelenet indirekt leírása miatt szükséges a koordináta-rendszerek közötti konverzió. Ennek a fejezetnek a célja, hogy összefoglalja mindazon fogalmi és módszertani alapokat, amelyekre az értekezés későbbi fejezeteiben bemutatott eljárások és eredmények épülnek. Részletesen bemutatom az optikai jelenség leírásához szükséges alapfogalmakat, valamint ismertetem azokat a szimulációs megközelítéseket, amelyek a megvalósítás szempontjából meghatározóak. Prezentálom ugyanakkor a mindkét területet érintő munkákban kulcsfontosságú baricentrikus koordináták tulajdonságait és számítási módjait.

2.1. Képalkotó rendszerek lencsefényfoltjai

A becsillanási lencsefényfoltok képalkotó rendszerek által létrehozott jelenségek, legyen szó az emberi szemről vagy pedig kamerákról, és az észlelt kép részeivé válnak. Bár a fényfoltokat mindkét esetben a megfigyelt jelenetben részt vevő erős intenzitású fényforrások okozzák, megjelenésük erősen függ attól, hogy emberi szem vagy kamera hozta őket létre. Általánosan két alkotóelemre bontható a jelenség. Az egyik ezek közül közös az emberi szem és a kamera által generált fényfoltokban, ez pedig a csillagszerű becsillanási mintázat megjelenése a fényforrás közelében. Az emberi látásban észlelt fényfolt másik alkotóeleme egy szivárvány árnyalatait tartalmazó gyűrű szintén a fényforrás körül. A kamerák képalkotó folyamata során létrejövő jelenség ehelyett viszont színes, áttetsző, szellemeknek nevezett foltokat tartalmaz a fényforrástól változó távolságokban.

Egy kamera esetén a lencsefényfoltoknak nevezett jelenséget a kamerába belépő, majd azon végighaladó fénysugarak nemvárt diffrakciója és belső visszaverődése okozza [9]. A fent is felsorolt alkotóelemek közül a becsillanást

a diffrakció okozza, a szellemeket pedig a belső visszaverődések. A fényfoltszimulációs munkánk során a szellemek hű reprodukálására fókuszáltunk.

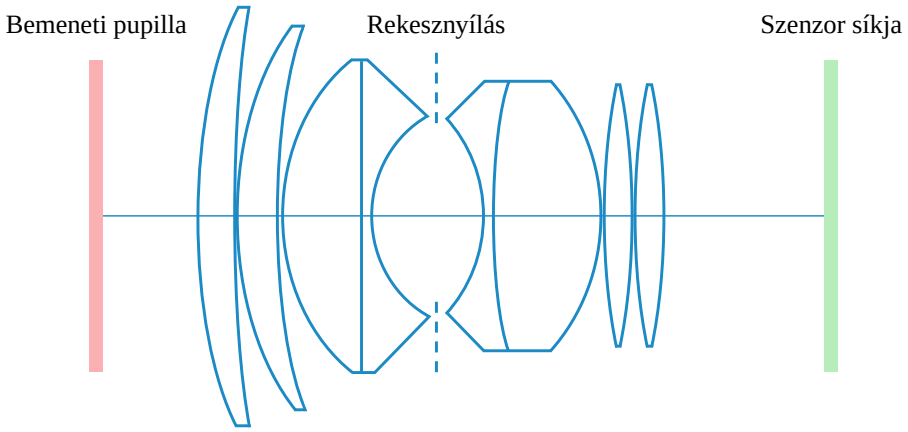
2.2. Kamerarendszerek leírása

Egy valós optikai rendszer esetén az analitikus sugárkövetés végrehajtásához elengedhetetlen a rendszer fizikai tulajdonságainak ismerete. A munkánk során felhasznált optikai rendszer lencseelemek sorozatából és egy rekeszből áll. Ezek a lencseelemek a modell egyszerűsítése és a sugárkövetés komplexitásának csökkentése érdekében síkokként és gömbi felületekként vannak reprezentálva analitikus formulákkal. A valós optikai rendszerek között található olyan, ami komplexebb elemeket is tartalmaz (például aszférikus felületeket), az ezekkel történő metszésvizsgálat magas költsége miatt viszont ezeket figyelmen kívül hagytuk. Az ilyen típusú felületek reprezentációja és a szükséges metszési tesztek nehezebben meghatározhatók és implementálhatók, ami miatt ezek nem részei a munkánknak. Ez azonban nem jelenti korlátját a megközelítéseinknek, mivel az említett analitikus metszési formulák birtokában az általunk javasolt eljárások kiterjesztése az ilyen felületek támogatásával triviális.

Az optikai rendszer minden lencsekomponense egy vagy több fénytörő felületből állhat. Ezek a lencseelemek különálló felületek sorozatával vannak leírva, akkor is, ha egy adott lencsét több felület is alkot. Az egyes felületeket leíró tulajdonságokat a görbületi sugara, a vastagsága, az anyaga, a törésmutatója és a magassága jelentik. A rekesznyílás alakját egy maszktextúra felhasználásával modellezzük, aminek a magassága a sugárkövetéshez előre definiált. A 2.1. ábra egy sematikus reprezentációját mutatja be egy optikai rendszernek, aminek pontos leíró tulajdonságait a 2.1. táblázatban foglaltam össze.

Amint egy fénysugár elér egy felületet, az addig meglévő energiájának egy részét a lencse elnyeli, egy másik részét visszaveri, a fennmaradó részét pedig továbbítja a következő felület felé. A lencsefényfoltok létrejöttéhez legalább egy visszaverődésre szükség van a fénysugár útja során, viszont azon fénysugaraknak van lehetősége elérni a szenzort, amelyek páros számú visszaverődéssel rendelkeznek.

Ahogyan azt Hullin és mtsai. [5] javasolták, első lépésként felsoroljuk a



2.1. ábra. A 2.1. táblázatban leírt Angenieux Double-Gauss objektív vizualizációja.

potenciális sugárútvonalakat, amelyek lencsefényfoltot okozhatnak. A 2.2. ábrán látható néhány ilyen útvonal (az útvonalakat az OpenLensFlare nevű eszközzel generáltuk [10]). Egy fénysugár energiája minden fényvisszaverődés után csökken. A teljes fizikai hitelesség érdekében a lehetséges sugárútvonalak mindegyikét figyelembe kellene venni, ám az esetek nagy többségében csak azok a sugarak rendelkeznek elegendő energiával a látható szellemek létrehozásához, amelyek csak két fényvisszaverődést szenvedtek. Emiatt munkánk során ezekre a fénysugarakra szorítkoztunk. Szintén Hullin és mtsai. javaslatára [5] figyelmen kívül hagytuk azokat a fénysugarakat, amik a rekesznyíláson több, mint egyszer haladnak át (két fényvisszaverődés megy végbe a rekesznyílás két oldalán). Az ilyen sugarak nagy része beleütközik a rekeszbe, ezért az ő hozzájárulásuk a látható szellemekhez minimális. A 2.2. ábrán látható sugárútvonalak ezzel a megközelítéssel lettek generálva.

2.3. Szellemek meghatározása analitikus sugárkövetéssel

Ahogyan azt a 2.2. alfejezetben is ismertettem, a szellemfoltok olyan sugarak által jönnek létre, amik páros számú visszaverődésen mennek keresztül, mivel ezek a sugarak azok, amik elérhetik a rendszer szenzorát. Ahhoz, hogy a szellemek szenzoron létrejött képét meghatározzuk, sugárkövetést alkalmaztunk minden olyan sugár esetén, amely a fenti kritériumoknak megfelel. Hullin

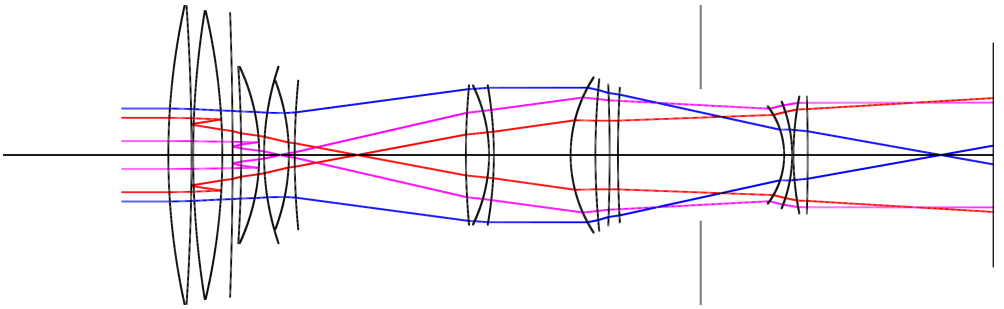
Görbületi sugár	Vastagság	Anyag	Törésmutató	Magasság
164,13	10,99	SF5	1,6751	27,00
559,20	0,23	levegő		27,00
100,12	11,45	BAF10	1,6689	25,50
213,54	0,23	levegő		25,50
58,04	22,95	LAK9	1,6913	20,50
2551,10	2,58	SF5	1,6751	20,50
32,39	15,66	levegő		13,50
-	15,00	rekesznyílás		12,75
-40,42	2,74	SF15	1,6992	12,50
192,98	27,92	SK16	1,6204	18,00
-55,53	0,23	levegő		18,00
192,98	7,98	LAK9	1,6913	17,50
-225,30	0,23	levegő		17,50
175,08	8,48	LAK9	1,6913	17,50
-203,55	55,74	levegő		17,50

2.1. táblázat. Egy Angenieux Double-Gauss objektív (USP 2701982A, f/11, 100 mm fókusztávolság, 14 fénytörő felület) fizikai struktúrája.

és mtsai. [5] javaslatát követve párhuzamos sugarak egy ritka rácsát követjük végig az optikai rendszeren. Egy sugárútvonal esetén a sugár metszéspontjait a rendszer felületeivel sorra megvizsgáljuk, ezt követően pedig meghatározzuk a visszaverődéseket és fénytöréseket a felületekkel való találkozáskor egészen addig, amíg a sugár el nem éri a szenzort.

A sugár által hordozott energia minden visszaverődéssel csökken, ezért a létrejövő szellem láthatósága is jelentősen alacsonyabbá válik a visszaverődések számának növekedésével. Ebből az okból kifolyólag Hullin és mtsai. [5] csak azokat a fénysugarakat vették számításba a szimuláció során, amik pontosan két visszaverődésen mennek keresztül, mivel az ettől több visszaverődés közel láthatatlanná teszi a szellemet. Ugyanakkor azokat a sugarakat, amik kettőnél több visszaverődésen mennek keresztül, nagyobb eséllyel akadályozza a rekesz. Ezek tehát figyelmen kívül hagyhatók, mivel elhanyagolható a hatásuk a szimuláció valószerűségére.

Mivel a lencsefelületek síkokként és gömbi felületekként vannak reprezentálva, analitikus felületleírások egy sorozatával rendelkezünk. Emiatt a sugár viselkedését egyszerűen meg tudjuk határozni a sugár-gömb és a sugár-sík metszéspontok hatékony vizsgálatával. A sugárkövetés alatt a sugaraknak a lencse-



2.2. ábra. Egy Nikon objektív (S.53-131852, f/16, 140 mm fókusztávolság) által generált sugárútvonalak közül néhány. A kék vonalak olyan sugárútvonalakat reprezentálnak, amelyeknek nem része fényvisszaverődés, ők alkotják a kívánt képet. A piros és lila vonalak olyan sugárútvonalakhoz tartoznak, amelyek során két fényvisszaverődés ment végbe, ezáltal lencsefényfolt-szellemet generáltak.

felületekkel való interakciója során a visszavert energia mennyiségét, valamint a visszavert vagy tört sugárirányokat analitikus formulákkal határozzuk meg. Ennek a folyamatnak az eredménye egy szenzorra vetített sugárrács, amely a létrejövő szellemet reprezentálja. A rácspontok közti réseket raszterizációval és hardveresen gyorsított interpolációval töltjük ki annak érdekében, hogy egy folytonos szellemtextúrát kapjunk.

A sugárkövetés során eltárolunk néhány sugarankénti tulajdonságot is, amik az interpolációhoz szükségesek. Ilyen például a sugárnak rekesznyílás síkján kapott koordinátái, amire a rekeszt reprezentáló maszktextúra mintavételezéséhez, valamint az optikai tengelytől a sugárkövetés alatt kapott legnagyobb távolság meghatározásához van szükség. Meghatározzuk ugyanakkor a sugár továbbított és a visszavert energiamennyiségét a szenzor elérésekor észlelt intenzitás miatt. Erre a célra a Fresnel-féle egyenleteket használjuk [11]:

$$R = \frac{1}{2}\left(\frac{n_1\cos\theta_1 - n_2\cos\theta_2}{n_1\cos\theta_1 + n_2\cos\theta_2}\right)^2 + \frac{1}{2}\left(\frac{n_1\cos\theta_2 - n_2\cos\theta_1}{n_1\cos\theta_2 + n_2\cos\theta_1}\right)^2, \tag{2.1}$$

$$T = 1 - R, \tag{2.2}$$

ahol R és T a visszavert és a továbbított energia mennyiségét jelöli. Továbbá az optikai elemek magasságát használjuk fel az érvénytelen sugarak felismerésére (mint például azok a sugarak, amik az objektív borításába vagy a rekeszbe

ütköznek). Ezek az érvénytelen elemek hozzájárulnak az interpoláció analitikailag folytonos kezeléséhez, ezáltal fontos szerepük van a ritka sugárkövetéses megközelítésben. Végezetül mivel minden sugár garantáltan csak egyszer halad keresztül a rekesznyíláson, ezért megjelöljük ezeknek az áthaladási pontoknak a relatív pozícióját a rekesznyílás síkján. Ezeket textúrakoordinátákként felhasználva a raszterizáció során a maszktextúrával együttesen vagyunk képesek a pontos íriszalakzat reprezentációjára.

2.4. Polinomiális optika

A fényfolt-szimulációs folyamat egyik legköltségesebb fázisa az analitikus sugárkövetés. Ennek egy ígéretes alternatív megoldása a más renderelési feladatokra már jól alkalmazott polinomiális optika. A hatékonyság növelése érdekében a sugárkövetés magas költségét egy előszámítási lépésbe helyezik át, amit elegendő egyszer végrehajtani, a valós renderelési folyamaton kívül, így a renderelés idejét nem befolyásolja. A létrejövő sugárkövetett adatokra aztán polinomok rendszerét illesztik, aminek a renderelés alatt történő kiértékelési költsége már szignifikánsan alacsonyabb, mint a teljes sugárkövetéses folyamat végrehajtásának.

Az itt említett polinomok a következő formában adottak:

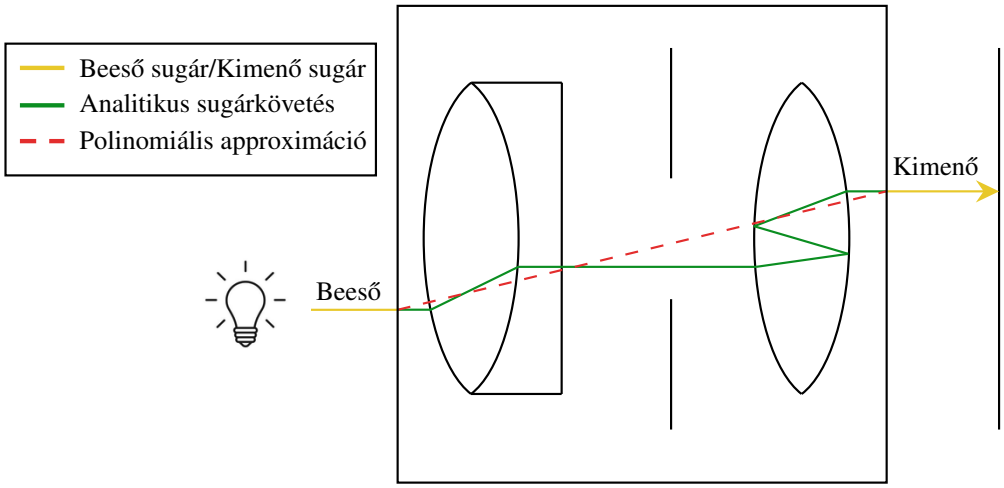
$$p(x) = \sum_{k=0}^{t} m_k, \tag{2.3}$$

ahol $x = (x_1, x_2, ..., x_d)$ egy d dimenziós bemeneti vektora a polinomnak, t a polinomtagok száma, m_k pedig a k-adik polinomtag. Az m_i polinomtag egy együtthatónak és a bemeneti változók egy adott hatványának a szorzata. Formálisan az alábbi módon írható fel:

$$m_i = c_i \cdot \prod_{j=1}^{d} x_j^{l_{i,j}}. \tag{2.4}$$

Egy polinomot akkor nevezünk *sűrűnek*, ha az összes polinomtag együtthatója nemnulla ($c_i \neq 0$ minden i esetén), egyébként *ritkának* nevezzük.

A polinomoknak számos előnye van, mint például a simaságuk, a képességük a tetszőleges bemeneti adatra való egyszerű illesztésre, a hozzávetőlegesen alacsony kiértékelési költségük vagy a deriváltjuk egyszerű használata. Gyakorlatban a polinomokkal megvalósított sugárátvitelhez tipikusan egy saját polinomot hoznak létre az összes kimeneti sugárparaméterhez, ami a bemeneti sugarat az optikai rendszer egy kiválasztott síkjáról (mint például a bemeneti pupilla síkja) közvetlenül egy szintén tetszőlegesen választott síkra (mint például rekesz síkja vagy az optikai rendszer szenzora) transzformálja. Ez a megközelítés szimuláció közben az optikai rendszer mögöttes struktúráját nem használja fel, hanem azt egy fekete dobozként kezeli azután, hogy a polinomiális rendszer fel lett építve. Ennek a megközelítésnek a sematikus rajzát láthatjuk a 2.3. ábrán.

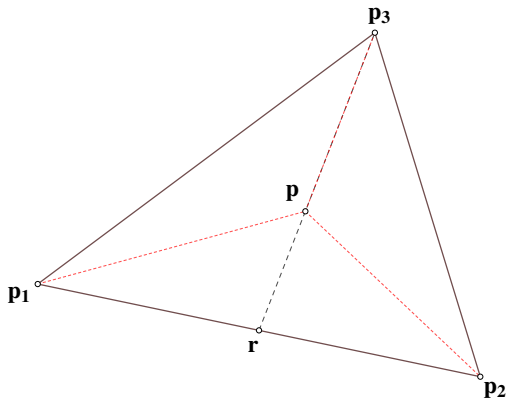


2.3. ábra. Az analitikus sugárkövetés és a polinomiális rendszerrel történő közelítése.

2.5. Baricentrikus koordináták

A későbbi fejezetekben bemutatott kutatási munka során több különböző esetben is hangsúlyos szerepet kaptak a baricentrikus koordináták. Mivel fontos volt ezeknek a hatékony meghatározása, így ebben az alfejezetben szeretném ismertetni azok működési elvét és különböző számítási módjait.

A baricentrikus koordináták képesek egy tetszőleges pont pozícióját meghatározni néhány referenciapont megfelelően súlyozott összegeként úgy két, mint három dimenzióban. Hagyományosan a kétdimenziós baricentrikus koordináták kiszámításához háromszögeket használnak, mivel erre a célra csupán három referenciapont szükséges [12]. Ezzel egy időben azonban felmerülhet olyan szituáció, amikor a baricentrikus koordinátákat több, mint három referenciapont figyelembevételével szükséges meghatározni, például négyszögek esetén. Három dimenzióban már térbeli pontok egy halmazára támaszkodva szükséges meghatároznunk a baricentrikus koordinátákat, amihez legalább négy referenciapont szükséges, tehát egy tetraéder csúcsai. Az alábbiakban a Hughes és mtsai. könyvében [13] leírtakra alapozva mutatom be a baricentrikus koordináták működését kétdimenziós esetben.



2.4. ábra. A $\mathbf{p_1p_2p_3}$ háromszög belsejében lévő $\mathbf{p}$ pont baricentrikus koordinátái. Az ábrán $\mathbf{r} = (1-0{,}5)\,\mathbf{p_1} + 0{,}5\mathbf{p_2}$ és $\mathbf{p} = (1-0{,}4)\,\mathbf{r} + 0{,}4\mathbf{p_3}$. A $\mathbf{p}$ pont t, u és v baricentrikus koordinátáinak meghatározására a bemutatott számítási eljárást alkalmazva a következőt kapjuk: $\mathbf{p} = 0{,}3\mathbf{p_1} + 0{,}3\mathbf{p_2} + 0{,}4\mathbf{p_3}$.

Tekintsünk egy háromszöget $\mathbf{p_1}$, $\mathbf{p_2}$ és $\mathbf{p_3}$ csúcsokkal. Erre látható egy példa a 2.4. ábrán. A $\mathbf{p_1}$ és $\mathbf{p_2}$ csúcsok közötti élen lévő bármelyik pont meghatározható a következő formában:

$$(1-k)\,\mathbf{p_1} + k\mathbf{p_2}, \tag{2.5}$$

ahol $0 \leq k \leq 1$. Tegyük fel, hogy az $\mathbf{r}$ pont a fenti formában definiált. Végezetül az $\mathbf{rp_3}$ szakaszon lévő bármelyik $\mathbf{p}$ pont meghatározható a fentihez hasonlóan

a következő formában: $(1-l)\,\mathbf{r}+l\mathbf{p_3}$, ahol $0 \leq l \leq 1$.

Így tehát a $\mathbf{p}$ pontot kifejezhetjük a k, l és a háromszög pontjainak felhasználásával. A fenti számításokból a következőt kapjuk:

$$\begin{aligned}
\mathbf{p} &= (1-l)\mathbf{r}+l\mathbf{p_3} = \\
&= (1-l)((1-k)\mathbf{p_1}+k\mathbf{p_2})+l\mathbf{p_3} = \\
&= (1-l)(\mathbf{p_1}-k\mathbf{p_1}+k\mathbf{p_2})+l\mathbf{p_3} = \\
&= \mathbf{p_1}-k\mathbf{p_1}+k\mathbf{p_2}-l\mathbf{p_1}+kl\mathbf{p_1}-kl\mathbf{p_2}+l\mathbf{p_3} = \\
&= (1-k-l+kl)\mathbf{p_1}+(k-kl)\mathbf{p_2}+l\mathbf{p_3}
\end{aligned} \tag{2.6}$$

Vezessük be a következő jelöléseket:

$$t \coloneqq 1-k-l+kl, \tag{2.7}$$

$$u \coloneqq k-kl, \tag{2.8}$$

$$v \coloneqq l. \tag{2.9}$$

Ezeket használva $\mathbf{p}$ felírható a

$$\mathbf{p} = t\mathbf{p_1}+u\mathbf{p_2}+v\mathbf{p_3} \tag{2.10}$$

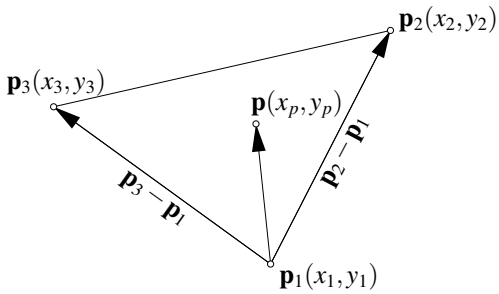
formában. Könnyen látható, hogy $t+u+v=1$. A t, u és v számok a $\mathbf{p}$ pont baricentrikus koordinátái, és használatukkal bármely pont meghatározható a $\mathbf{p_1}$, $\mathbf{p_2}$ és $\mathbf{p_3}$ pontok súlyozott összegeként. A javasolt számítási módszerrel 0 és 1 közötti baricentrikus koordinátákkal a háromszög belső pontjait tudjuk leírni, de a koncepció általánosítható a sík bármely pontjára.

A baricentrikus koordináták meghatározását egy háromszög csúcsai alapján Marschner és Shirley könyvükben [12] az alábbi módon formalizálták:

$$A = \begin{pmatrix} x_2-x_1 & x_3-x_1 \\ y_2-y_1 & y_3-y_1 \end{pmatrix}, \quad \boldsymbol{\lambda} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} x_P-x_1 \\ y_P-y_1 \end{pmatrix}, \tag{2.11}$$

$$\boldsymbol{\lambda} = A^{-1}\cdot\mathbf{b}, \tag{2.12}$$

aminek jelölései a 2.5. ábrán vannak prezentálva.

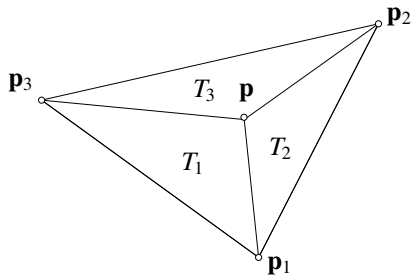


2.5. ábra. A **p** pont baricentrikus koordinátáinak háromszögre támaszkodó kiszámításához használt jelölések.

A baricentrikus koordinátákat háromszögek területeinek arányaiként is meghatározhatjuk szintén a Marschner és Shirley könyvében bemutatottak alapján. Jelöljük λ_1, λ_2 és λ_3-mal a keresett baricentrikus koordinátákat. Ezek a következő formában állnak elő:

$$\lambda_1 = \frac{T_1}{T_{\text{teljes}}}, \quad \lambda_2 = \frac{T_2}{T_{\text{teljes}}}, \quad \lambda_3 = \frac{T_3}{T_{\text{teljes}}}, \tag{2.13}$$

ahol T_1 a $\mathbf{p_1pp_3}$ háromszög, a T_2 a $\mathbf{p_1pp_2}$ háromszög, a T_3 a $\mathbf{p_2pp_3}$ háromszög, a T_{teljes} pedig a $\mathbf{p_1p_2p_3}$ háromszög területét jelölik. Ezeket a jelöléseket láthatjuk a 2.6 ábrán.



2.6. ábra. A **p** pont baricentrikus koordinátáinak háromszögterületekre támaszkodó kiszámításának szemléltetése.

A 2.12. egyenlet tulajdonképpen egy lineáris egyenletrendszer megoldását szolgáltatja, amiben az inverzszámításhoz a mátrix determinánsára van szükség. A 2.13. egyenletben alkalmazott területek meghatározásához szintén determinánsok használata szükséges. Belátható, hogy a két számítási mód ugyan-

azokhoz a determinánshányadosokhoz vezet.

Skala [14] homogén koordináták vektoriális szorzattal kombinált alkalmazását javasolta a baricentrikus koordináták kiszámítására, ami egy, a grafikus meghajtókon hatékonyabb számítást eredményezett. A számítás a következő módon formalizálható:

$$\mathbf{b} = \mathbf{x} \times \mathbf{y} \times \mathbf{w}, \tag{2.14}$$

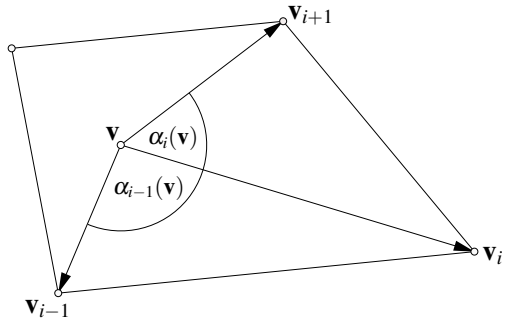
ahol $\mathbf{x} = (x_1,\ x_2,\ x_3,\ x_P)^T$, $\mathbf{y} = (y_1,\ y_2,\ y_3,\ y_P)^T$, $\mathbf{w} = (1,\ 1,\ 1,\ 1)^T$, és $\mathbf{b} = = (b_1, b_2, b_3, b_4)^T$ jelöli a baricentrikus koordináták homogén alakját. A végső baricentrikus koordináták aztán a következő módon vannak meghatározva:

$$a_1 = -\frac{b_1}{b_4}, \quad a_2 = -\frac{b_2}{b_4}, \quad a_3 = -\frac{b_3}{b_4}\ . \tag{2.15}$$

Hormann és Tarini [15] munkájukban egy olyan eljárást javasoltak, amellyel a baricentrikus koordináták kiszámítását egy négyszög csúcsaira támaszkodva tudjuk elvégezni. Ennek előnye az a képesség, hogy tetszőleges formájú négyszögeket képes kezelni, például konkáv és önmagát metsző négyszögeket is:

$$\lambda_i(\mathbf{v}) = \frac{\tan(\alpha_{i-1}(\mathbf{v})/2) + \tan(\alpha_i(\mathbf{v})/2)}{\|\mathbf{v} - \mathbf{v_i}\|}, \tag{2.16}$$

aminek a jelöléseit a 2.7. ábra prezentálja, $\alpha_i(\mathbf{v})$ pedig a $(\mathbf{v}_i - \mathbf{v})$ és a $(\mathbf{v}_{i+1} - \mathbf{v})$ által bezárt előjeles szöget jelöli.



2.7. ábra. A **v** pont baricentrikus koordinátáinak négyszögre vonatkozó kiszámításához használt jelölések.

Konvex és nem önmagát metsző poligonok esetén a baricentrikus koordináták egy gyorsabb számítását lehet végrehajtani a Wachspress-koordináták felhasználásával [16]. A Wachspress által leírt eredeti formalizációnak a GPU műveletek hatékony kihasználásának tekintetében előnyösebb módosított felírása a következő [17]:

$$\lambda_i(\mathbf{v}) = \frac{w_i(\mathbf{v})}{\sum_{j=1}^{n} w_j(\mathbf{v})}, \tag{2.17}$$

$$w_i(\mathbf{v}) = A(\mathbf{v}_{i-1},\ \mathbf{v}_i,\ \mathbf{v}_{i+1}) \prod_{j \neq i-1,\ i} A(\mathbf{v},\ \mathbf{v}_j,\ \mathbf{v}_{j+1}), \tag{2.18}$$

ahol $A(\mathbf{v},\ \mathbf{v}_j,\ \mathbf{v}_{j+1})$ jelöli a $[\mathbf{v},\ \mathbf{v}_j,\ \mathbf{v}_{j+1}]$ háromszög előjeles területét. Egy alternatív módszerként Loop és DeRose [18] a következő definíciót javasolta a w_i meghatározására:

$$w_i(\mathbf{v}) = \prod_{j \neq i-1,\ i} A(\mathbf{v},\ \mathbf{v}_j,\ \mathbf{v}_{j+1}). \tag{2.19}$$

3. Lencsefényfoltok szimulációja

Annak ellenére, hogy a lencsefényfoltok gyakran nemkívánatos velejárói a képalkotási folyamatnak, mégis egy erőteljes művészi eszközként szolgálnak a fotográfia és filmográfia területén [1, 2], videojátékokban pedig a valószerűség növelésében van hangsúlyos szerepük. Számos jelentős tudományos alkalmazással is rendelkeznek, mint például a becsillanásoknak a hétköznapi feladatokra (például vezetés [19]) gyakorolt hatása, valamint vizuális jelzések szolgáltatása olyan szintetikus képanyagok esetén, ahol magas intenzitású fényforrások jelennek meg [20]. Ebből kifolyólag a jelenség számítógéppel történő reprodukciója hosszú ideje fókuszban van.

A hatékony és fizikailag korrekt fényfolt-szimulációk mellett sok új munka a fényfoltok eltávolítására összpontosít [3, 4, 21, 22] és nem azoknak a számítógéppel történő szintézisére. Ezen munkák nagy része olyan tanítóhalmazok használatára építkezik, amik becsillanási fényfoltot tartalmazó képekből állnak [4, 21–23]. Ezek az adathalmazok tipikusan digitális kamerákkal készített fényképeket tartalmaznak, ami a folyamatot bonyolulttá, hibára hajlamossá, idő- és erőforrás-igényessé teszi [21]. A tanításhoz szükséges becsillanási fényfoltokat tartalmazó képek számítógéppel is létrehozhatók, emiatt pedig a hatékony fényfolt-szimulációs eljárások nagy mértékben hozzájárulhatnak a munka hatékonyságához.

3.1. Szakmai előzmények, motiváció

Tekintve a számítógép által létrehozott képek minél valószerűbbé tétele iránti egyre növekvő igényt, a kapcsolódó kutatási területeken hangsúlyos szerepet kapnak a képalkotási folyamatok tulajdonságai, valamint az azok által eredményezett jelenségek reprodukálása. Ilyen például a kromatikus aberráció [24], valamint a lencsefényfoltok szimulációja is [25–27]. A továbbiakban röviden ismertetem a lencsefényfoltok szimulációjával kapcsolatos, a szakirodalomban korábban elérhető releváns munkák főbb eredményeit.

3.1.1. Textúraalapú módszerek szellemek szimulációjára

A lencsefényfoltok szellemkomponensének szimulációjára a legkorábbi megközelítések statikus textúraelemeket alkalmaztak. A jelenség fizikai tulajdonságai ebben az esetben nem voltak számításba véve, a textúrákat empirikus módon helyezték el a képen. Kilgard [28] algoritmusában a fényfolttextúrákat a kép középpontján átmenő egyenes mentén helyezték el. Ezt követően King [29] kiterjesztette az eljárást a fényfoltok méretének és intenzitásának módosításával azoknak a kép közepétől számított távolsága alapján. Maughan [30] a fényfoltok intenzitásának finomhangolását a fényforrások részleges láthatóságának figyelembevételével oldotta meg. Később Sekulic [31] a takarási viszonyok meghatározási idejének csökkentésére hardveresen gyorsított eszközöket javasolt. Oat [32] a textúraelemek használatát helyettesítette utófeldolgozó szűrőkkel, amiket a kép magas intenzitású pontjaiban alkalmazott. Végezetül Alspach [33] egy vektoralapú megközelítést implementált a szellemek reprezentációjára, amelyet a felhasználó által paraméterezhetővé tett.

3.1.2. Fizikailag megalapozott módszerek szellemek szimulációjára

Mivel az igény a szimulációk minél magasabb fokú valószerűségére egyre inkább növekszik, ezért a lencsefényfoltok fizikailag megalapozott módon történő reprodukciója szintén egy fontos témává vált. Tekintve, hogy ez a jelenség erősen kameraspecifikus, a kamera optikai paramétereit fontos volt a szimulációs folyamat részévé tenni. A sugárkövetés egy gyakran alkalmazott eleme ennek a megközelítésnek az általa támasztott magas számítási igény ellenére is, a módszerrel elérhető hitelesség miatt. Először Chaumond [34] tett javaslatot a szellemek fizikailag megalapozott szimulációjára, amiben fénykövetést alkalmazott egy valós optikai rendszer fénytörő felületeivel. Keshmirian [35] fotontérképeket használt fel a fény terjedésének meghatározására. Mivel ezek a megközelítések nem voltak alkalmasak speciális kameraeffektusok (pl. spektrális fényfoltok) szimulációjára, ezért Steinert [27] az eljárás részeként a lencsék borításainak paramétereit is figyelembe vette. Mindamellett azonban, hogy a fényfoltok megjelenése javult, az eljárás teljesítménye miatt az nem volt alkalmas valósidejű szimulációra. A Lendermann és mtsai. [36] által

javasolt eljárásban a precíz diffrakciószimuláció kap szerepet, illetve a szerzők olyan optikai rendszereket alkalmaztak, amik tetszőleges fókusztávolsággal és rekesznyílás-mérettel rendelkeznek. Hanika és Dachsbacher [37] Monte Carlo renderelőt alkalmazott, hogy tovább növelje a pontosságot. Az optikai rendszer precíz leírásának hiányában Walch és mtsai. [38] olyan digitális képeket dolgoztak fel, amelyeken látható lencsefényfoltok jelentek meg. Ezek alakjára Bézier-görbék egy sorát illesztették, amelyek segítségével a fényfoltok szintézisét tetszőleges fényforrás-pozíció jelenlétében valósították meg.

A fent részletezett algoritmusok jó minőségű kimenetek létrehozására alkalmasak, viszont számításigényesek. A probléma megoldására Hullin és mtsai. [5] egy sugárkötegelést alkalmazó eljárást javasoltak ritka sugárrácsok felhasználásával. Az általuk javasolt megközelítésben egy durva sugárrácsot követtek végig az optikai rendszeren az összes szellemre. A szellemek így egy háromszögelt alakzat formájában jöttek létre, a ritka sugárrács réseit pedig interpolációval töltötték be. Pekkarinen és Balzer [2] bebizonyította a módszer gyakorlati alkalmazhatóságát. A szellemek egyesével történő kirajzolása azonban magas számú memóriatranzakciót igényelt, ezáltal csökkentve az algoritmus áteresztőképességét. Az eljárás előfeldolgozó lépésének ideje szintén lényegesen növekedett a kamerarendszer komplexitásának emelésével. A renderelési teljesítmény további javításának céljából Lee és Eisemann [39] paraxiális sugárkövetést és transzfermátrixokat alkalmazott a sugárkövetés eredményének approximációjára, amit a textúraalapú módszerek előnyeivel együtt használt fel. Hennessy [40] a szellemek intenzitását a rekesznyílás átmérőjéből határozta meg, a színük megállapítására pedig egyetlen sugarat követett végig a rendszeren egy véletlenszerűen megválasztott beesési szöggel, minden csatorna esetén. Ezeknek az eljárásoknak a hátránya abban rejlik, hogy nem képesek komplex szellemalakzatokat kezelni (pl. nemlineáris deformációk), mivel minden szellem egy textúrázott négyszögként van megjelenítve.

3.1.3. Algoritmusok a becsillanás szimulációjára

A lencsefényfoltok becsillanási komponense az emberi szemben megjelenő becsillanáshoz hasonló, mivel mindkettő a diffrakcióból és a fényszórásból ered,

amelyeket a rendszeren áthaladó fénysugár útjában lévő akadályok okoznak. Kakimoto és mtsai. [41] a Fraunhofer-diffrakció számítási módját alkalmazták az emberi szemben létrejövő becsillanás szimulációjára. Van den Berg és mtsai. [42] ezt követően a Rayleigh–Debye–Gans approximációt használták a szemben lévő molekulák által okozott fényszórás modellezésére. Ritschel és mtsai. [20] ezt a jelenséget Fresnel diffrakciós integráljával és a gyors Fourier-transzformálttal modellezték. Ezekhez az eljárásokhoz hasonlóan a kamerákban létrejövő fényfoltok becsillanási komponensét Hullin és mtsai. [5] szimulálták először, amihez a rekesznyílás képének Fourier-transzformáltját alkalmazták. Joo és mtsai. [43] továbbfejlesztették ezt az eljárást az olyan lencsetökéletlenségek szimulációjával, amik a gyártási folyamat során jönnek létre, majd kiterjesztették az algoritmust az aszférikus lencsék kezelésével. Végül Scandolo és mtsai. [44] javasoltak egy hierarchikus algoritmust a Fraunhofer-diffrakció gyors kiszámítására.

3.1.4. Polinomiális optika a szellemek szimulációjára

Az analitikus sugárkövetés valósághű képeket eredményez ugyan, viszont nagyon magas számítási költségekkel jár. Emiatt a szakirodalomban megjelentek a polinomiális optika használatát javasló munkák, amik a sugárkövetés eredményét közelítik sokkal rövidebb idő alatt.

Hullin és mtsai. [6] egy olyan eszközt hoztak létre, amivel a sugártranszferfüggvények gyorsan és pontosan közelíthetők azok Taylor-sorfejtésének néhány elemével. A szerzők polinomok egy rendszerét illesztették az optikai rendszerben lévő összes lencsefelületre, amiből a végső transzferfüggvény a polinomok összefűzésével jött létre. Schrade és mtsai. [45] kiegészítették ezt a polinomiális eljárást széles látószögű objektívekkel és létrehoztak egy továbbfejlesztett polinomillesztési eljárást, amely a polinom legrelevánsabb tagjainak kiválasztásával képes a sűrű polinomokat pontosan közelítő ritka polinomokat előállítani. Ezt követően Zheng és Zheng [46] adaptív polinomiális regressziót alkalmazott a ritka polinomiális lencsemodell felépítésére, ami a polinomillesztési folyamat további javulását eredményezte mind a pontosság, mind a futási idő tekintetében. Goossens és mtsai. [47] egy olyan módszert javasoltak,

ami már meglévő sugárkövetési adatokra illeszt polinomokat.

Hullin és mtsai. [6] bebizonyították, hogy az általuk létrehozott eszköz képes lencsefényfoltokat szimulálni offline környezetekben. Megközelítésük azonban nem vette figyelembe a felületi visszaverődést és a sugarak akadályozását az objektív borítása által, amik jelentősen megváltoztatták a kapott fényfolt megjelenítését. Pekkarinen [2] szintén Taylor-polinomokat alkalmazott a fényfoltok renderelésére, viszont a pontos takarási viszonyok meghatározására a renderelt jelenet geometriáját és a kamerarendszer elemeit is figyelembe vette. Ezenkívül módszerük a Schlick-formulát alkalmazta az első lencsefelületen átvitt energia durva becslésére. Később Dilorio [48] további optimalizációkat javasolt a polinomtagok számának csökkentésére, viszont ennek eredményét csak egy nagyon kicsi, szintetikus objektív-összeállításra értékelte ki. Nemrégiben Sabatschus [49] ritka polinomiális rendszereket alkalmazott a lencsefényfoltok modellezésére, viszont a bemutatott eredmények nem támasztották alá a valós alkalmazhatóságot, mivel a létrejövő polinomok nem voltak alkalmasak valósághű és precíz fényfolt-szimulációk létrehozására.

A fent bemutatott korábbi, polinomiális optikára építkező szimulációs megközelítések több szignifikáns korlátozással is rendelkeznek. Egyrészt sem az objektív borítása által akadályozott elemek egyenkénti helyes ellenőrzése, sem a Fresnel-egyenletek kiértékelése (a továbbított és visszavert energiamennyiség kiszámításához) nincs figyelembe véve ezekben a munkákban. Másrészt az eredményeket nem ellenőrizték referenciakimenetekkel, aminek a hiányában viszont a bemutatott eredmények érvényessége nem meghatározható. Harmadrészt egyik módszer fókusza sem a GPU-alapú végrehajtás által elérhető magas teljesítményen van. Ez pedig általában nagy méretű polinomiális rendszereket eredményez, jelentősen növelve a polinomiális rendszer kiértékelésének költségét. Végezetül az előző problémához kapcsolódóan fontos megjegyezni, hogy a renderelési teljesítményt sem elemezték részletesen az említett munkákban. Emiatt viszont a megközelítések alkalmazhatósága valósidejű környezetekben nem megállapítható.

3.1.5. Összegzés és motiváció

A terület relevanciájának köszönhetően a szakirodalomban számos megoldás fellelhető a becsillanási fényfoltok számítógépes szimulációjára. Ezekről a megoldásokról azonban elmondható, hogy mindegyikük kompromisszumot köt, hogy elérje a kívánt célját, legyen szó a hatékonyságáról (fizikai helyesség nélkülözésével) vagy a szimuláció minőségéről (költséges sugárkövetésre támaszkodva). Ennek következményeként a már meglévő megoldások nem képesek egy időben pontosan és gyorsan szimulálni tetszőleges alakzatú, fizikailag megalapozott becsillanási fényfoltokat úgy, hogy a valósidejű alkalmazások elvárásait is kielégítsék. A területen végzett munkánk során tehát a lencsefényfolt szellemkomponenseinek szimulációját szerettük volna megvalósítani oly módon, hogy az valós időben alkalmazható legyen, valamint a kimenet minőségében és fizikai alapjaiban se kelljen kompromisszumot kötni.

A szakirodalomban elérhető közvetlen analitikus lencsefényfolt-szimulációs megközelítések többnyire csak akkor alkalmasak valósidejű alkalmazásokhoz, ha az optikai rendszer mérete (ezáltal a szimulált szellemek darabszáma) alacsony, valamint ha csak egyszerű (sík és szférikus) felületi elemeket tartalmaz az optikai rendszer. A polinomiális regresszió képes lehet a folyamat számos részét érintő korlátozások feloldására. Azonban a fényfoltokat létrehozó sugárútvonalak és a létrejövő szellemalakzatok komplexitásából kifolyólag ezeknek a módszereknek a valósidejű alkalmazása magas minőségű lencsefényfolt-renderelésre bonyolult, ahogyan azt a releváns korábbi munkák is demonstrálták ezen a területen [2, 6, 48, 49]. Transzfermátrixok [39] szintén alkalmazhatók a fényfoltok szimulációjához szükséges sugárkövetés hatékony végrehajtására, viszont ez a megközelítés nem képes a komplex szellemalakzatok kezelésére, ezáltal csökkentve a szimuláció valószerűségét. Munkánk fő motivációja ezen korlátok feloldása volt.

3.2. Szellemek hatékony raszterizációja

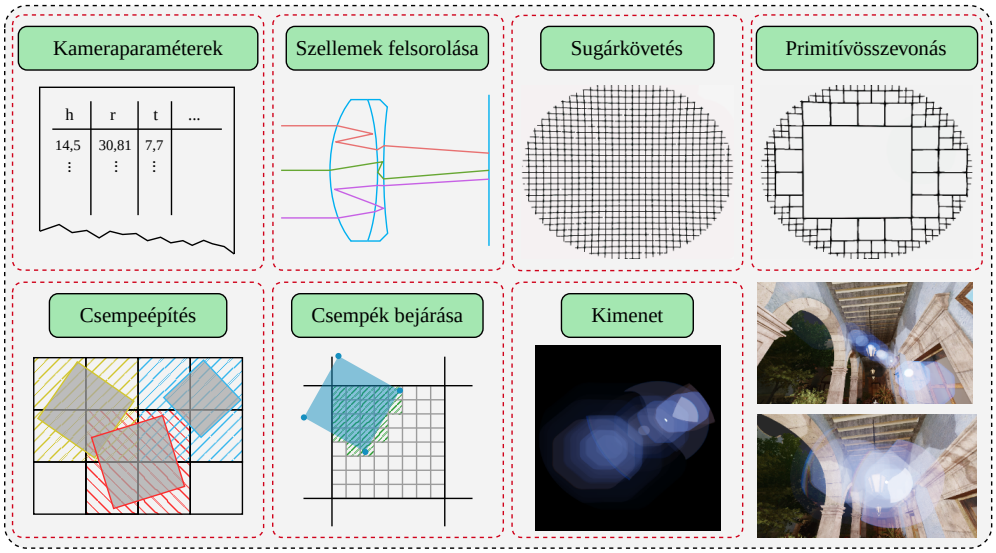
Amint azt a 3.1. fejezetben is kiemeltem, azok az algoritmusok, amik statikus textúrákkal dolgoznak, semmilyen fizikai megalapozottsággal nem rendelkez-

nek, viszont azok a megközelítések, amik fizikailag korrekten hivatottak szimulálni a lencsefényfoltokat, számítási igényükből fakadóan nem használhatók valósidejű alkalmazásokban. Fő célunk tehát egy olyan eljárás kidolgozása volt, amelynek a felhasználásával lehetővé válik a lencsefényfolt szellemkomponenseinek reprodukciója interaktív környezetekben sugárkövetés alkalmazásával. Ehhez első lépésében a fényfolt-szimuláció raszterizációs fázisát vizsgáltuk meg. Ebben az alfejezetben a 2023-ban megjelent publikációnk [50] alapján mutatom be munkánk részleteit és a területen elért eredményeinket.

Ahogyan azt a 2.3. alfejezetben bemutattam, a lencsefényfoltok szellemkomponensének szimulációjára egy ritka sugárrács optikai rendszeren történő végigkövetését alkalmazzuk Hullin és mtsai. [5] javaslata alapján. Ennek a megközelítésnek egy kritikus hátránya azonban a magas memóriasávszélességre vonatkozó igény, ami a szellemek egyesével történő raszterizációjából fakad. A szellemek gyakran nagy részben fedik egymást, és annak ellenére, hogy közülük sok alig látható, a végső renderelés költségéhez mégis hozzájárulnak, hiszen az intenzitás és a láthatóság mértékétől függetlenül egyesével megjelenítésre kerülnek. Az átfedő szellemek kirajzolása a közös pixelek nagy mennyiségű újraírását eredményezi, ami viszont magas sávszélességi igényekhez vezet. Ugyanez a probléma a számítógépes grafika más feladatainál is felmerült már. Ilyen és ehhez hasonló feladatok megoldására többször használtak már csempealapú megközelítést, amellyel a kimenet előállításához szükséges adatok hatékonyabban pufferelhetők, így pedig a renderelési fázis költsége csökkenthető [7, 8]. Ebből kiindulva egy olyan algoritmust hoztunk létre, ami jelentősen hatékonyabbá teszi a fényfoltok raszterizációját és növeli a fényfolt-szimuláció általános áteresztőképességét.

Algoritmusunk fő lépéseit az 1. algoritmus foglalja össze és a 3.1. ábra vizualizálja. Az első fázisban a sugárkövetést hajtjuk végre ritka sugárrácsokkal [5]. Következő lépésként a sugárkövetés után létrejövő rácsokból négyzeteket építünk. Ezt követően a létrejött primitíveket a kimeneti kép oldalaival párhuzamos oldalú, nagy méretű csempékbe gyűjtjük, majd ezt a felosztást tovább finomítjuk, ezáltal egy kisebb méretű csempéket tartalmazó rácsra bontva fel a képet. Végezetül kimeneti pixelenként bejárjuk az egyes képpontokhoz tartozó csempéket, és létrehozzuk a végső szellemszimulációt. A fejezet to-

vábbi részeiben részletes leírást nyújtok új módszerünk egyes lépéseiről.



3.1. ábra. Az általunk javasolt szellemrenderelő algoritmus lépései. Eljárásunk bemenetként a szimulált kamerarendszer fizikai paramétereit kapja. Ezt követően felsoroljuk a legláthatóbb, legmagasabb intenzitással rendelkező szellemeket képező sugárútvonalakat. Következő lépésben egy ritka sugárrácsot követünk végig az optikai rendszeren a hullámhossz és szellemútvonal minden egyedi kombinációjára. Ennek eredményét felhasználva négyszögeket hozunk létre a kimeneti sugárrácsokból, összevonjuk a hasonló szomszédos négyszögeket, majd az eredményeket egy globális pufferben eltároljuk. A következő fázisban egy kétszintű (durva és finom felosztású) csempehierarchiát hozunk létre azon négyszögek indexeit felhasználva, amik metszik az egyes csempéket. Végezetül a kimenetet az egyes kis méretű csempék képpontonkénti bejárásával hozzuk létre, amihez a releváns négyszögek hozzájárulását gyűjtjük össze.

3.2.1. Primitívek létrehozása

Az általunk javasolt raszterizációs fázis első fő lépése a szenzoron létrejövő szellemet alkotó primitívek létrehozása, amit a kimeneti képpé alakítunk majd. A felsorolt szellemútvonalak birtokában elvégezzük a sugárkövetést, aminek eredményeként a szenzorra vetített sugárrácsok egy halmazát kapjuk. Egy ritka sugárrácsot követünk végig az optikai rendszeren, amit minden csatorna és minden felsorolt szellem esetén végrehajtunk. A sugárrácsok réseit a sugarankénti információk interpolációjával töltjük be.

1. algoritmus. A lencsefényfolt szellemkomponenseinek renderelésére javasolt saját algoritmusunk.

Bemenet: Optikai rendszer paraméterei ($\mathscr{O}$),
hullámhosszak listája (Λ),
fényforrások listája ($\mathscr{L}$),
aktuális képkocka kameraállása ($\mathscr{C}$),
primitívösszevonási lépések száma (n_m)

Kimenet: az összes kimeneti pixel listája, ami a bemenetek alapján létrejövő összes szellemet szimulálja ($\mathscr{P}_o$)

$\mathscr{G} \leftarrow \text{SzellemekFelsorolasa}(\mathscr{O})$

foreach $(g, \lambda, l) \in \mathscr{G} \times \Lambda \times \mathscr{L}$ **do**
  $\sigma \leftarrow \text{BejovoFenyiranyMeghatarozasa}(\mathscr{C}, l)$
  $n_g \leftarrow \text{SugarracsMeretMeghatarozasa}(g, \lambda, \sigma)$
  $\mathscr{R}_i \leftarrow \text{SugarracsEpites}(n_g, \sigma)$
  $\mathscr{R}_s \leftarrow \text{SugarracsKovetes}(\mathscr{R}_i, \mathscr{O}, \lambda)$
  $\mathscr{P} \leftarrow \text{PrimitivEpites}(\mathscr{R}_s)$

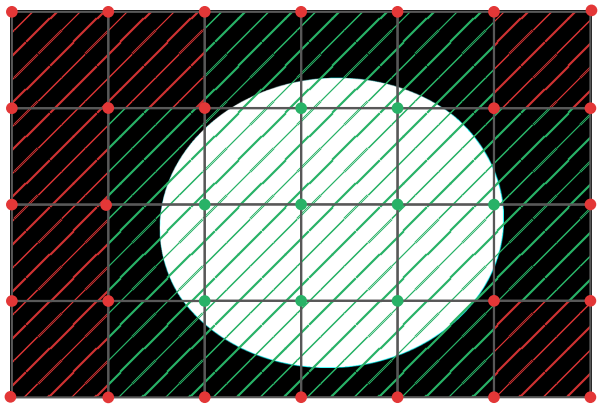
for $i \leftarrow 1$ **to** n_m **do**
  $\mathscr{P} \leftarrow \text{SzomszedosPrimitivekOsszevonasa}(\mathscr{P}, i)$

$\mathscr{T} \leftarrow \text{CsempekEpitese}(\mathscr{P})$

foreach $(t, p_o) \in \mathscr{T} \times \mathscr{P}_o$ **do**
  $p_o \leftarrow \text{PrimitivHozzajarulasokOsszesitese}(t, p_o)$

Mivel a szellem egyes részeit csoportosítani szeretnénk, ezért a kimeneti sugarakból primitíveket hozunk létre. Ehhez négyszögeket építünk a sugárrács minden 2×2 szomszédságából. Ezzel egy időben szeretnénk eldobni az érvénytelen primitíveket. Egy négyszög akkor lesz érvényes, ha legalább egy érvényes sugarat tartalmaz, mivel ezek a négyszögek fognak az interpolációhoz hozzájárulni és elengedhetetlenek az analitikai folytonossághoz. Emiatt eldobunk minden olyan négyszöget, amit csak érvénytelen sugarak alkotnak. Egy sugarat akkor tekintünk érvényesnek, ha a sugár nem ütközik sem a rekeszbe, sem az objektív borításába, rendelkezik nem elhanyagolható mennyiségű energiával, és nem szenved el teljes fényvisszaverődést. Ezt a folyamatot a 3.2. ábra illusztrálja.

A fent leírt szűrési eljárásunk a primitívek eltárolandó listájának, ezáltal



3.2. ábra. A sugárrács primitívjeinek szűrése az adatmennyiség csökkentése érdekében. A zöld pontok az érvényes, a piros pontok pedig az érvénytelen sugarakat jelölik. Egy primitívet akkor tárolunk el, ha az legalább egy érvényes sugarat tartalmaz (zölddel kiemelve), egyébként eldobjuk (pirossal kiemelve).

pedig módszerünk későbbi lépéseiben vizsgálandó adat mennyiségének drasztikus csökkenését eredményezi. Azokat a primitíveket, amik átmentek a szűrési folyamaton, egyetlen primitívpufferben tároljuk el, ami a szimuláció összes további lépésében lesz majd felhasználva.

3.2.2. Primitívek összevonása

Ahhoz, hogy komplex szellemalakzatokat pontosan, műtermékek nélkül szimuláljunk, gyakran nagy elemszámmal rendelkező sugárrácsokat szükséges használnunk. Ilyen esetekben azonban a komplexitást eredményező speciális görbületek általában a sugárrács széleihez közel jönnek létre, míg minden egyéb terület reprezentálható lenne kevesebb sugarat tartalmazó ráccsal is. Emiatt a sugarak sűrűségére vonatkozó igény leszűkül a periférikus területekre.

Egy ideális megoldást jelentene az adaptív sugárrács használata, ahol csak azok a területek tartalmaznának magasabb sűrűségben sugarakat, amik a magas komplexitást okozzák. Ilyen adaptív rácsokat létrehozni és kezelni azonban erőforrás-igényes, ami a feladatot problémássá teszi és megnehezíti a gyakorlati alkalmazhatóságot. Ebből kifolyólag eljárásunkban egyenlő felosztású sugárrácsokat alkalmaztunk [5], viszont azt egy olyan primitív-összevonó lépéssel egészítettük ki, aminek eredményeként az alacsony komplexitású területekhez

tartozó szomszédos primitívek összeolvadnak, mivel ezeket indokolatlan lenne magas részletességgel reprezentálni.

Ennek a folyamatnak a során olyan 2×2-es méretű primitívblokkokat próbálunk összevonni, amelyeknek részét képező primitívek teljesítenek egy feltételhalmazt. Először ellenőrizzük az összes primitív érvényességét, ezzel garantálva azt, hogy a rács határain lévő érvénytelen primitívek figyelmen kívül lesznek hagyva ebben a folyamatban, tehát a magas komplexitású régiók változatlanok maradnak. Ezt követően páronként megvizsgáljuk az összevonandó primitívek éleit. Ezek irányai is kellően közel kell legyenek egymáshoz (tehát közel párhuzamosak kell legyenek), egyébként valamilyen komplex görbület jött létre a vizsgált régióban, így azok a négyszögek nem összevonhatók. Ennek a feltételnek a kiértékelésére a tesztelendő primitívek élpárjainak koordinátánkénti eltérését összegezzük. Két élről akkor állítjuk, hogy egy irányba néznek, ha ennek a számításnak az eredménye egy felhasználó által paraméterezhető (γ) határérték alá esik. Ez a következő módon formalizálható:

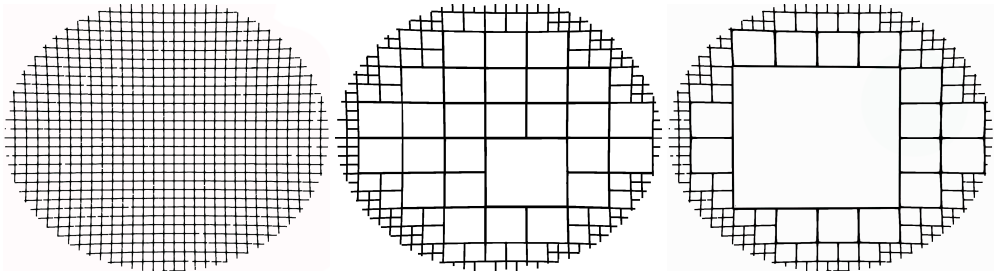
$$\mathbf{e}_A = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}, \quad \mathbf{e}_B = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}, \tag{3.1}$$

$$(x_1 - x_2) + (y_1 - y_2) < \gamma, \tag{3.2}$$

ahol $\mathbf{e}_A$ és $\mathbf{e}_B$ a tesztelendő élpárt jelöli. Egy 2×2-es primitívblokk akkor teljesíti azt az összevonási feltételt, ami az élekre vonatkozik, ha a blokkot alkotó összes szomszédos primitív esetében páronként teljesül a fent leírt párhuzamossági feltétel a vizsgált primitívek négy élpárjának mindegyikére.

Végezetül egy iteratív primitívösszevonást hajtunk végre a primitívépítési fázis során. Az egyes összevonási lépések eredményei tovább egyesíthetők egymást követő lépésekben, aminek eredményeként egyre nagyobb primitívblokkokat hozhatunk létre. Ha az összevonási lépéseket egy hierarchia elemeiként képzeljük el, akkor csak az azonos szinten lévő primitívek vonhatók össze, mivel az összevonás nem hajtható végre egy nagyobb és egy kisebb primitív esetén. Az összevonás után létrejövő alakzatok tehát nem lehetnek négyszögtől eltérő alakúak. A 3.3. ábra egy Nikon objektív által generált szellem sugárkövetett rácsképének az iteratív primitívösszevonási folyamatunk végrehajtá-

sa során létrejövő néhány fázisát mutatja be összevonás nélkül, egy mérsékelt összevonással ($\gamma = 0{,}001$) és egy nagyobb mértékű összevonással ($\gamma = 0{,}1$).



3.3. ábra. Eltérő mértékű primitívösszevonások vizualizációja. Összevonás nélkül (baloldali), mérsékelt összevonás (középső), nagy mértékű összevonás (jobboldali).

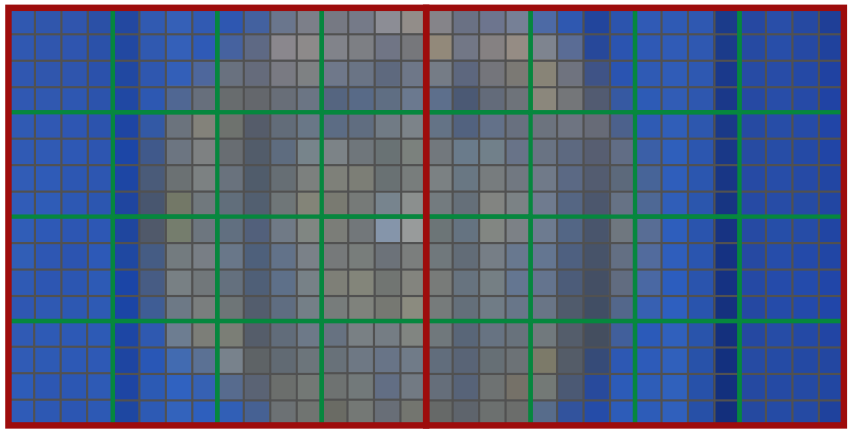
3.2.3. Csempepuffer létrehozása

A sugárkövetés kimenete a primitívekből felépített szellemképeknek egy halmaza. Célunk a szellemeket alkotó primitívek csoportosítása az átfedő részeik alapján. Ebből kifolyólag egy egységesen felosztott rácsot hozunk létre, aminek celláit csempéknek nevezzük, mérete pedig a kimeneti kép méretével egyezik meg. A csempéket csoportosítás céljára használjuk, így a kimeneti kép pixeleit a megfelelő csempékhez rendeljük pozíciójuk alapján. A csempékhez olyan primitíveknek egy-egy listáját rendeljük, amelyek a csempe legalább egy kimeneti pixelét tartalmazzák (tehát legalább egy pixeléhez hozzájárulnak). A folyamat végén az egyes csempék csak azon primitívek halmazát fogják tartalmazni, amelyek relevánsak a csempében található pixelek tekintetében.

A csempénkénti primitívpufferek hatékony létrehozásához egy kétszintű hierarchikus csoportosítást implementáltunk, ami növeli a csempeépítés hatékonyságát. Az első szinten a primitíveket egy durva rácsfelosztás szerint csoportosítjuk (nagy méretű csempékkel), ami még a primitívépítési fázisban van megvalósítva, miután a primitíveket sikeresen összevontuk. A durva felosztás egy előszűrési lépést jelent, ami szignifikánsan csökkenti a végső felosztáshoz vizsgálandó primitívek számát.

A hierarchikus csoportosítás második lépésében bejárjuk a durva felosztás során létrejött nagy méretű csempéket a csoportosítás finomítása érdekében.

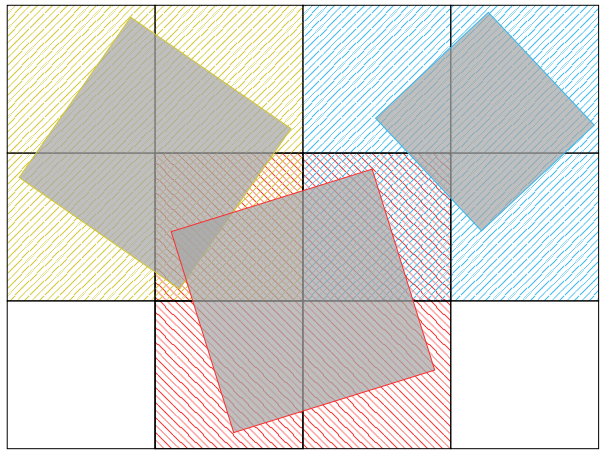
Ehhez az aktuális nagy méretű csempe által lefedett, a finomított felosztás során létrejövő kisebb méretű csempékhez rendeljük a számukra releváns primitíveket. Minden esetben biztosítjuk, hogy az összes kis méretű csempét teljesen lefedjen egy nagy méretű csempe, emiatt a durva felosztás méretét a finom felosztás méretének többszörösére választjuk. Egy ilyen csempehierarchia látható a 3.4. ábrán.



3.4. ábra. Csempehierarchia. A pirossal, zölddel és szürkével körvonalazott cellák jelölik a durva felosztást, finom felosztást, és a kimeneti képpontokat.

A hierarchia mindkét szintjén a csempék és a primitívek metszéseinek vizsgálatával csoportosítjuk a primitíveket. Az összes primitív esetén bejárjuk a csempék listáját, és meghatározzuk a potenciális metszéseket. A metszésellenőrzések egyszerűsítéséhez a primitívek képtérbeli koordinátáit és a tengelyekkel párhuzamos oldalú határoló négyzeteit használjuk. Amikor a vizsgálat azt mutatja, hogy egy primitívnek és egy csempének van átfedő része, a primitívet hozzárendeljük a csempéhez. A folyamat sematikus vizualizációját mutatja be a 3.5. ábra, az algoritmikus leírását pedig a 2. algoritmus foglalja össze.

A fenti folyamat eredményeként az összes csempéhez olyan primitívek egy halmazát kapjuk, amik potenciálisan hozzájárulnak a kimeneti kép azon pixeleihez, amiket lefed az adott csempe. A memóriafelhasználás csökkentése miatt a halmazok reprezentációja a globális primitívpufferbe mutató indexekkel történik. A globális primitívpuffer azokat az összevont primitíveket tartalmazza, amik az előző primitívlétrehozási és -összevonási fázisban jönnek létre.



3.5. ábra. Csempepuffer építése. A vizualizált rács minden cellája egy csempét reprezentál. A sárgával, kékkel és pirossal kiemelt cellák mutatják azokat a csempéket, amik az azonos színű primitívek legalább egy pixelét tartalmazzák.

3.2.4. Csempepuffer pixelenkénti bejárása

Az általunk javasolt eljárás fő célja, hogy elkerülje a szellemek egyesével történő, pixelenkénti raszterizációját. Emiatt egy lépésben összegyűjtjük az összes szellem potenciális hozzájárulását az egyes pixelekhez. A cél elérése érdekében csempénként egy puffert alkalmaztunk, amely puffereknek a képpontonkénti bejárásával összegyűjtöttük az egyes primitívek hozzájárulását a képpontokkal való átfedő részeik alapján. Ez a folyamat látható a 3.6. ábrán.

A végső megjelenítés a kimeneti kép képpontonkénti bejárásával történik, amely során meghatározzuk és összesítjük a releváns primitívek hozzájárulásait. A bejárás alkalmával sorra vesszük a kimeneti kép pixeleit, majd minden pixel esetén megvizsgáljuk az őt tartalmazó csempe primitívlistáját. A kimeneti pixel előállításához a bejárás során összesítjük a lista összes olyan primitívjének hozzájárulását, amelynek része a vizsgált pixel (a hozzájárulás 0, ha nem része a pixel a primitívnek). Az eljárás összefoglalását a 3. algoritmus tartalmazza. A pixelenkénti hozzájárulás meghatározásához baricentrikus koordinátákat alkalmazunk, ahogyan a 3.2.5. fejezetben részletezésre kerül.

A pixelek bejárása párhuzamosan történik, csoportosításuk pedig a kis méretű csempékhez való hozzárendeltségük alapján valósul meg. A folyamat so-

2. algoritmus. Kétszintű hierarchikus csempeépítési eljárásunk.

Bemenet: Összevont primitívek listája ($\mathscr{P}$),
nagy és kis méretű csempék listája ($\mathscr{T}_c$, $\mathscr{T}_d$).
Kimenet: Az átfedő primitívekkel feltöltött csempénkénti pufferek.

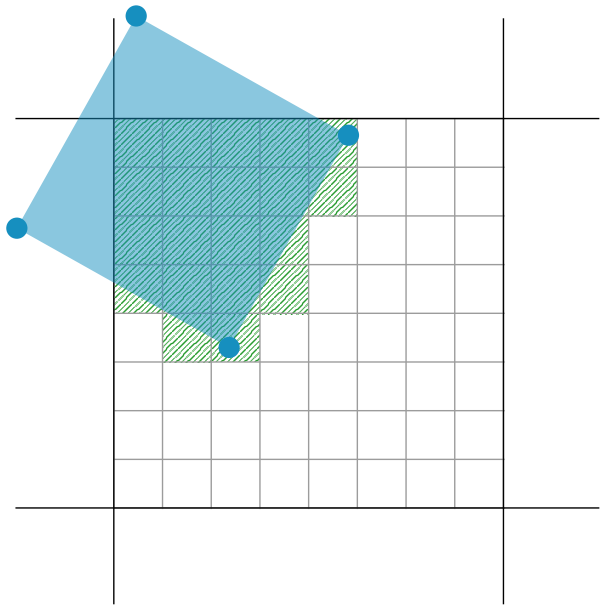
```
foreach (p,t) ∈ 𝒫 × 𝒯_c do
    if p ∩ t ≠ ∅ then
        PrimitivIndexNagyCsempehezRendeles(p,t)
foreach t ∈ 𝒯_d do
    𝒫_t ← TartalmazoNagyCsempePrimitivjei(t)
    foreach p ∈ 𝒫_t do
        if p ∩ t ≠ ∅ then
            PrimitivIndexKisCsempehezRendeles(p,t)
```

3. algoritmus. A primitívek pixelenkénti hozzájárulásának meghatározása és összesítése.

Bemenet: Kis méretű csempék puffereit tartalmazó lista ($\mathscr{T}_d$)
Kimenet: Kimeneti kép az összes primitív raszterizációjával

```
foreach t ∈ 𝒯_d do
    𝒫_t ← CsempePrimitivjeinekBeolvasasa(t)
𝒫_i ← CsempePixelei(t)
foreach (p_t, p_i) ∈ 𝒫_t × 𝒫_i do
    b ← BaricentrikusKoordinatakKiszamitasa(p_t, p_i)
    if b ∈ [0,1] then
        p_i ← PrimitivHozzajarulasOsszesitese(p_t, p_i, b)
```

rán hardveralapú csoportosítást alkalmaztunk (úgynevezett compute shaderek felhasználásával), mivel ez a megközelítés képes a primitívadatok olvasását szétosztani a compute shader csoportok tagjai között, valamint hatékonyan elérhetővé tenni a beolvasott adatot a megosztott memória segítségével. Ennek következményeként módszerünk sávszélességigénye szignifikánsan csökkent, amiből kifolyólag a csempebejárás hatékonyabbá vált.



3.6. ábra. Pixel és primitív közti átfedés ellenőrzése. A szürkével jelölt cellák egy adott csempe pixeleit jelölik. A zölddel jelölt cellák reprezentálják azokat a pixeleket, amiket valóban lefed a kék primitív (tehát metszik egymást). Ezekben az esetekben a kék primitív hozzájárul a kimeneti pixel színéhez.

3.2.5. A primitívek képpontonkénti hozzájárulásának meghatározása

Az egyes primitívek által lefedett képpontoknak a meghatározására és a rács kitöltéséhez szükséges adatinterpolációra baricentrikus koordinátákat használtunk, amit már sikeresen alkalmaztak kétdimenziós primitívekhez kapcsolódó interpolációs feladatokra [51, 52]. Egy adott primitív négy csúcsát referenciákként felhasználva minden pixelhez meg tudjuk határozni annak baricentrikus koordinátáit. A baricentrikus koordináták természetéből fakadóan egy pixel pozíciója relatív a primitívhez képest, amit fel tudunk használni egy potenciális metszés meghatározására.

Ahogyan azt a 3.2.1. alfejezetben bemutattam, a sugárkövetés végrehajtása után eredményül kapott sugárrácsot felhasználva négyszögelt szellemeket hozunk létre. A primitívek baricentrikus koordinátákkal történő feldolgozásának egy módja lehetne az, ha a négyszögeket két háromszögre osztanánk, és mindkét háromszöget külön dolgoznánk fel a (2.12) vagy a (2.15) egyen-

let felhasználásával. Gyakorlati kísérleteink során viszont azt figyeltük meg, hogy hatékonyabb számítást kapunk, ha közvetlenül a négyszögekkel dolgozunk. Ebben az esetben a (2.16) vagy a (2.17) egyenletekben leírt számítási módokat használhatjuk fel.

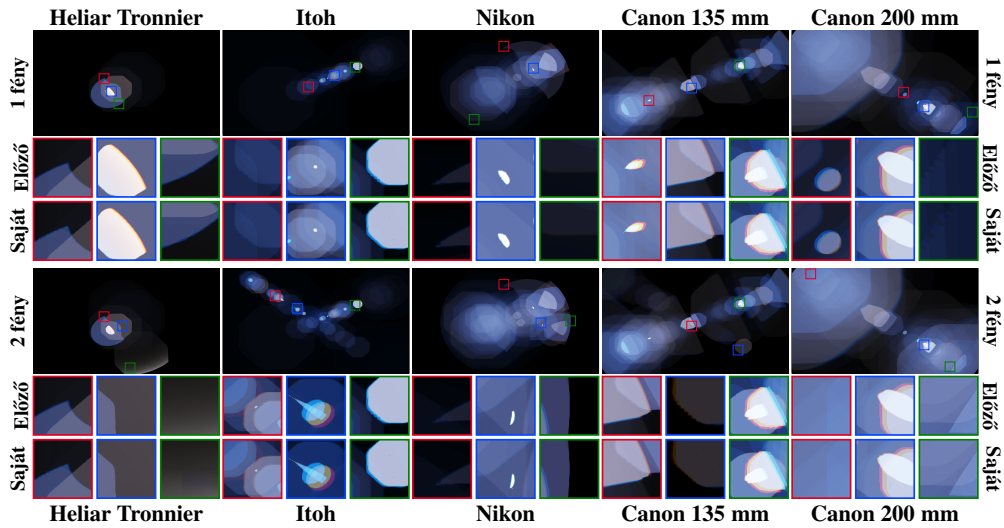
Mindkét megközelítés esetén igaz az, hogy ha akárcsak egyike a baricentrikus koordinátáknak a $[0,1]$ tartományon kívül esik, akkor a pixel nem része a négyszögnek, ezért az ő hozzájárulása nulla. Egyébként kiértékeljük az adott négyszögnek a kimeneti pixel színéhez való hozzájárulását, amit szintén baricentrikus koordinátákkal valósítunk meg. A pixelt tartalmazó primitív négy csúcsát referenciapontokként felhasználva a hozzájárulást a primitív (háromszög vagy négyszög) csúcsaihoz tartozó tulajdonságok (intenzitás, rekesznyíláson való metszéspont koordinátái stb.) súlyozott összegeként kapjuk meg, ahol a súlyokat a pixelhez tartozó baricentrikus koordináták adják. A csempebejárás során ezt az értéket összesítjük.

Mivel a sugárkövetés bemenete egy szabályos rács, az optikai rendszer elemei pedig gömbi felületek, ezért a rács képe a szenzoron konvex négyszögeket tartalmaz. Emiatt a Loop és DeRose [18] által javasolt számítási módot alkalmaztuk a (2.17) és a (2.19) egyenletekben leírtakkal. Ugyanakkor az általánosított számítási módszer szintén jól működik a baricentrikus koordináták négyszögekre vonatkozó kiszámítására. Az egyes baricentrikus megközelítések hatásai a számítási időkre a 3.2.6.6. alfejezetben vannak kiértékelve.

3.2.6. Eredmények

A referenciaimplementáció létrehozásához a C++ programnyelvet és az OpenGL grafikus könyvtárat használtuk fel. A magas számítási igénnyel rendelkező fázisokat GLSL compute shaderekkel valósítottuk meg. Amint azt a 3.2.4. alfejezetben kiemeltem, az implementáció során a megosztott memória tulajdonságait kihasználva szignifikánsan növeltük az adatolvasás hatékonyságát. Hardverek tekintetében egy AMD Ryzen 5 1600 CPU-t és egy NVIDIA GeForce RTX 3060 GPU-t használtunk az összes méréshez.

Eredményeink kiértékelésére öt különböző tesztesetet alkalmaztunk, amelyekben szereplő optikai rendszerek eltérő komplexitással rendelkeznek:



3.7. ábra. Az általunk javasolt módszerrel generált lencsefényfoltok egy és két fényforrással. A kiemelt régiók a saját módszerünk és a Hullin és mtsai. által javasolt [5] referenciamódszer kimeneteinek összehasonlítását szolgáltatják.

– Egy Heliar Tronnier objektív alacsony számú lencsefelülettel (USP 2645156, f/16, 100 mm fókusztávolság, 8 felület, 11 szellem).
– Egy Itoh széles látószögű objektív, amely a közepes komplexitást reprezentálja (USP 4196968, f/32, 123 mm fókusztávolság, 18 felület, 64 szellem).
– Egy Nikon teleobjektív egy további közepes komplexitású kamera tesztelésére (S.53-131852, f/16, 140 mm fókusztávolság, 21 felület, 139 szellem).
– Egy Canon teleobjektív két különböző fókuszállásban a magas komplexitású optikai rendszerek reprezentációjára (USP 5537259, f/32, 135 mm és 200 mm fókusztávolság, 33 felület, 304 szellem).

Két különböző fényviszonyt teszteltünk az összes fenti objektív esetén, az első esetben egy, a második esetben pedig két fényforrást felhasználva a jelenetben. A 3.7. ábrán láthatók a kimeneti képek, amelyek felbontása 1920×1080.

A módszerünk összehasonlítására elkészítettük egy referenciaimplementációját a Hullin és mtsai. [5] által javasolt algoritmusnak, amihez az összehasonlíthatóság érdekében ugyanazokat az eszközöket használtuk fel, amiket a saját

módszerünk implementációjához is. A sugárkövetéses fázisban az összes eltérő fényforrás, szellem és RGB csatorna kombinációra egy külön sugárrácsot követtünk végig az optikai rendszeren.

Az előző eljárás és a saját módszerünk összehasonlítása során mindkét algoritmusban 256×256 méretű rácsokat alkalmaztunk, amit a szellem szenzorra vetített méretének megfelelően dinamikusan csökkentettünk. A csökkentés mennyiségének meghatározására a következő formulát alkalmaztuk:

$$\min\left(\max\left(\left(\frac{G_x \cdot G_y}{S_x \cdot S_y} \cdot \sigma\right)^{\varphi},\ 0{,}05\right),1\right), \tag{3.3}$$

ahol S és G jelöli a szenzorhoz és a szellem szenzorra vetített képéhez tartozó négyszög méretét, σ és φ pedig felhasználó által szabadon konfigurálható paraméterek. Az összes tesztesetben $\sigma = 1$-et és $\varphi = 0{,}5$-öt használtunk, amely értékeket empirikusan határoztuk meg, mivel ez a beállítás elegendő csökkenést eredményezett kis méretű szellemek esetén. Ez a megközelítés eltér a Hullin és mtsai. [5] által javasolt eljárástól, mivel ők a sugárrácsok méretét szellemenként határozták meg. Az általunk választott alap rácsméret hozzávetőlegesen nagy, viszont a célunk ezzel a választással a kimenet minőségének maximalizálása, valamint a rácsméretek meghatározási komplexitásának minimalizálása volt. A Hullin és mtsai. által javasolt megközelítés szintén egy működő alternatíva lenne mindkét összehasonlított eljáráshoz, ezért a mi megközelítésünk nincs hatással a teljesítményösszehasonlításra.

3.2.6.1. Teljesítménykiértékelés

Ebben az alfejezetben az általunk javasolt renderelő eljárás teljesítményprofilját értékelem ki, valamint összehasonlítom azt a Hullin és mtsai. által javasolt algoritmussal az előző alfejezetben leírt összes tesztesetben. A teljesítményelemzés elvégzéséhez megmértük az algoritmusunk egyes fázisainak futási idejét, valamint a teljes renderelési időt mindkét módszer esetén. Az eredményeket a 3.1. táblázatban foglaltam össze.

A teljesítménymméréshez 8×8-as kis csempeméretet, illetve 128×128-as nagy csempeméretet alkalmaztunk. Ezeket empirikusan választottuk, mivel kí-

sérleteink során azt figyeltük meg, hogy ezek a paraméterek eredményeztek egyensúlyt a futási idők és a kimenet minősége között. Az egyes paraméterek hatását szintén kiértékelem ennek a fejezetnek egy későbbi részében.

Az egyes fázisok teljesítményét tekintve elmondható, hogy a teljes számítási idő legnagyobb részét a sugárkövetéses folyamat adja. A következő részben mutatom be a sugárkövetés hatékonyságának fejlesztésén végzett munkánk részleteit és eredményeit. Második a sorban a raszterizáció, ami viszont az eredményeink által is alátámasztva hatékonyabbá tehető a primitívek összesítésével, masszív adatmennyiségek esetén is. Végezetül a csempeépítés költsége elhanyagolható, ami nagy részben a kétszintű hierarchikus megközelítésünk eredménye.

Az ebben a fejezetben bemutatott teljesítményjavításaink legfőképpen a csempealapú összesítésből fakadó memória-sávszélességi igény jelentős csökkenéséből, a kétszintű hierarchikus megközelítésnek köszönhető gyors primitívcsoportosításból és a primitívegyesítés miatti primitívmennyiségcsökkenésből adódnak. A primitívegyesítési megközelítés a raszterizációra és a csempeépítési lépésre egyaránt hatással van.

Általánosságában elmondható, hogy a javasolt módszerünk teljesítménye

		Saját módszerünk				**Előző módszer**		
		Sugárkövetés és nagy csempék	**Kis csempék**	**Csempe-bejárás**	**Teljes**	**Sugár-követés**	**Raszte-rizáció**	**Teljes**
1 fényforrás	Heliar Tronnier	0,26	0,10	0,36	**0,72**	0,14	0,96	**1,1**
	Itoh	1,17	0,13	0,57	**2,01**	0,78	1,27	**2,05**
	Nikon	3,47	0,28	1,65	**5,4**	2,36	5,46	**7,82**
	Canon 135 mm	8,09	0.31	1,83	**10,23**	6,42	13,09	**19,51**
	Canon 200 mm	17,27	0,49	4,54	**22,30**	13,49	19,44	**32,93**
2 fényforrás	Heliar Tronnier	0,44	0,11	0,54	**1,09**	0,23	1,37	**1,6**
	Itoh	1,96	0,17	0,87	**3,00**	1,30	2,12	**3,42**
	Nikon	5,87	0,50	2,87	**9,24**	4,29	13,36	**17,65**
	Canon 135 mm	9,38	0,36	2,73	**12,47**	6,77	14,3	**21,07**
	Canon 200 mm	24,95	0,66	5,49	**31,10**	18,70	27,07	**45,77**

3.1. táblázat. Saját algoritmusunk lépéseinek számítási ideje (ms), valamint a saját algoritmusunk és a referenciaalgoritmus teljes renderelési ideje (ms) öt különböző tesztesetre egy és két fényforrás esetén. A mérésekhez 8×8-as kis, valamint 128×128-as nagy csempeméretet alkalmaztunk.

felülmúlja az előző algoritmusét, ez látható a 3.1. táblázatban összefoglalt eredményekből is. Megközelítésünk nagyjából megduplázza a raszterizációs teljesítményt még a legegyszerűbb tesztesetben is, a legmagasabb komplexitással rendelkező szituációban pedig ötszörös teljesítménynövekedést eredményez. Méréseink szintén alátámasztják a raszterizációs fázis szignifikánsan jobb skálázhatóságát a fényforrások számának növekedésével. Emiatt algoritmusunk kiválóan alkalmazható olyan gyakorlati alkalmazásokban, ahol nagyszámú fényforrást alkalmaznak a jelenetekben.

3.2.6.2. Minőségkiértékelés

Hullin és mtsai. [5] elvégeztek egy összehasonlítást a sugárkövetés-alapú módszerük kimenete és valós kamerák által készített fényképek között. Ezzel alátámasztották eljárásuk képességét a valószerű kimenetek létrehozására, amely kimenetek megfelelően közelítik a rögzített lencsefényfoltokat. Mivel a mi eljárásunk pontosan ugyanazokra a fizikai alapokra épít, Hullin és mtsai. módszerét alkalmasnak ítéltük az összehasonlításhoz szükséges referenciaképek előállítására. Emiatt fő célunk az előző eljárás kimeneteinek reprodukálása volt a raszterizációs idő drasztikus csökkentésével.

Az általunk javasolt algoritmus kimeneti minőségének validációjához összehasonlítottuk a saját kimeneti képeinket a referenciaképekkel, aminek eredménye a 3.7. ábrán tekinthető meg. Az összehasonlításhoz meghatároztuk a 3.7. ábrán prezentált kimenetek és a referenciametódus által szolgáltatott kimenetek maximális jel-zaj viszonyát. Az említett metrika a szakirodalomban gyakran alkalmazott létrejövő képek pontosságának mérésére [53], valamint fényfoltok szimulációjával kapcsolatos munkák esetén is gyakran alkalmazzák [39, 44]. Az eredményeinket a 3.2. táblázatban foglaltam össze, amiből egyértelműen látható, hogy saját megközelítésünk hűen reprodukálja a célként kitűzött referenciaképeket, nem eredményez semmilyen szignifikáns eltérést a referenciametódushoz viszonyítva.

A 3.7. ábrán kiemeltem azokat a régiókat, amik a legmagasabb eltéréseket produkálták. Ezekből a régiókból láthatóvá válik, hogy a leghangsúlyosabb eltérések a szellemek alulmintavételezett széleinél vehetők észre. Ez a ritka

	Heliar Tronnier	Itoh	Nikon	Canon 135 mm	Canon 200 mm
1 fényforrás	50,27	49,50	49,91	46,33	46,23
2 fényforrás	46,86	46,81	48,27	45,91	47,59

3.2. táblázat. Az általunk javasolt eljárás a 3.7. ábrán prezentált kimeneteinek és a referenciametódus kimeneteinek maximális jel-zaj viszonya (PSNR, decibelben).

sugárrácsok interpolációjából fakad, ami mindkét algoritmus esetén javításra szorul. A sugárrács méretének növelése kiküszöbölné ezt a nemkívánatos hatást, mivel ebben az esetben az általunk javasolt, baricentrikus koordinátákon alapuló interpoláló algoritmus nem okozna ehhez hasonló hibákat.

Végezetül a primitívösszevonási lépés szintén okoz néhány látható eltérést. Ugyanakkor a raszterizációs megközelítések kimeneteinek precíz összehasonlítása érdekében nem végeztük el a Hullin és mtsai. által javasolt spektrális szűrést. Gyakorlati tapasztalataink azt mutatják, hogy egy ilyen spektrális elmosás teljesen eltüntetné ezeket a nemkívánatos jelenségeket, valamint más egyszerű képtérbeli szűrő (pl. mediánszűrő) használatával ez szintén jól javítható. Továbbá a primitívösszevonási lépések számának csökkentése vagy egy precízebb összevonási heurisztika szintén csökkentené a látható eltéréseket. Mindezek tekintetében elmondható, hogy gyakorlati alkalmazásokban a primitívösszevonási lépés semmilyen látható műterméket nem eredményezne.

3.2.6.3. Csempehierarchia hatása

Ebben az alfejezetben értékelem ki a kétszintű csempeépítési stratégiánk és az egyes csempeméretek hatásait a teljes renderelési időre. Ehhez 8×8-as és 16×16-os csempeméreteket teszteltünk, és mindkét esetben megvizsgáltuk a teljes futási időt a hierarchikus csempestratégia használatával és anélkül, egy fényforrással a jelenetben.

Méréseinket a 3.3. táblázatban foglaltam össze, amik alátámasztják, hogy a durva csempék használata szignifikánsan csökkenti a renderelési költséget minden esetben, mivel az egy hatékony és gyors előszűrést szolgáltat a csempeépítés során. Emiatt, amint azt az előző alfejezetben is említettem, ez a megközelítés biztosítja azt, hogy a csempeépítés költsége minimális maradjon.

Továbbá a kiértékelés azt mutatta, hogy a 8×8-as méretű csempék jobban teljesítettek az összes jelenetben. Ez a viselkedés azzal magyarázható, hogy a jelentősen kisebb csempénkénti primitívlista kellően alacsonyan tartja a raszterizáció során feldolgozandó primitívek mennyiségét. Emiatt úgy ítéltük, hogy a 8×8-as méret ideális lehet a gyakorlati alkalmazásokhoz, így tehát ezt a méretet választottuk a 3.2.6.2. alfejezetben leírt méréseinkhez is.

Fontos megjegyezni, hogy a kétszintű csempehierarchia és a csempeméretek megválasztása nincs hatással a kimeneti minőségre. Ennek oka, hogy a csempékben egy adott csempeméret esetén mindig pontosan ugyanazokat a primitíveket kapjuk a csempehierarchia alkalmazásával és anélkül. A hierarchikus megközelítés pusztán egy előszűrést biztosít a párhuzamos feldolgozás hatékonyságának érdekében. Ezenfelül a végeredményhez mindig ugyanazoknak a primitíveknek a hozzájárulásait gyűjtjük össze, függetlenül a csempe méretétől. A különbség csupán abban rejlik, hogy a képpontonkénti csempebejárás során pontosan hány darab nem releváns primitívet vizsgálunk meg, ám ezek hozzájárulása a baricentrikus koordinátákra épülő interpoláció miatt nulla.

3.2.6.4. Primitívösszevonás kiértékelése

Az általunk javasolt primitívösszevonási stratégia futási időre és kimeneti minőségre gyakorolt hatásának kiértékeléséhez három tesztesetet használtunk fel: összevonás nélkül, mérsékelt összevonással (4 összevonási lépés, $\gamma = 0{,}001$), valamint egy nagymértékű összevonással (4 összevonási lépés, $\gamma = 0{,}1$). Egyet-

	8×8		**16×16**	
	Durva felosztással	**Durva felosztás nélkül**	**Durva felosztással**	**Durva felosztás nélkül**
Heliar Tronnier	0,73	3,86	0,99	1,77
Itoh	1,89	6,84	2,26	3,38
Nikon	5,38	19,78	6,69	8,04
Canon 135 mm	10,76	25,04	12,24	13,56
Canon 200 mm	21,64	51,65	25,33	31,68

3.3. táblázat. Futási idő mérések (ms) eltérő csempeméretekkel a durva felosztással, valamint nélküle. A nagy csempék mérete 128×128.

len fényforrást alkalmaztunk az összes tesztesetben. Minden esetben megmértük a teljes renderelési időt, valamint a kimenetünk és a referenciaeljárás által generált kimenet maximális jel-zaj viszonyát. Az eredményeket a 3.4. táblázatban foglaltam össze.

	Nincs összevonás		**Mérsékelt összevonás**		**Nagymértékű összevonás**	
	Futási idő	**PSNR**	**Futási idő**	**PSNR**	**Futási idő**	**PSNR**
Heliar Tronnier	1,12	50,38	0,74	50,27	0,72	46,87
Itoh	2,45	49,54	2,03	49,50	1,91	49,50
Nikon	7,47	50,18	5,42	49,91	5,07	46,88
Canon 135mm	11,88	46,48	10,25	46,33	10,04	45,20
Canon 200mm	26,96	46,23	21,84	46,23	20,75	45,12

3.4. táblázat. Az általunk javasolt eljárás futási ideje (ms) és a kimeneteinket a referenciakimenetekkel összehasonlítva kapott PSNR értékek primitívösszevonás nélkül, mérsékelt összevonással és nagymértékű összevonással.

Amint azt az eredményeink is mutatják, a primitívösszevonási megközelítésünk még a legkisebb komplexitású kamerák esetén is javított a számítási teljesítményen, mivel a számítási komplexitás kellően alacsony ahhoz, hogy a kiszűrt primitívek nagyobb hatással legyenek a teljes szimulációs időre, mint az összevonási eljárás költsége. Ugyanakkor megfigyelhető, hogy amint a kamera komplexitása növekszik, a primitívösszevonás egyre nagyobb hatékonyságnövekedést eredményez.

A 3.4. táblázatból jól látható, hogy a mérsékelt összevonásnak elhanyagolható hatása van a kimeneti minőségre, viszont jelentősen javítja a futási időt már egy közepes komplexitású kamera esetén is. A folyamat kisebb hatással van ugyan a kis kamerarendszerekre, viszont hasznos lehet, ha több fényforrást alkalmazunk, mivel ilyenkor a létrejövő szellemprimitívek száma növekszik.

A nagymértékű összevonás csökkentené ugyan a PSNR-t és látható hibákat eredményezne, viszont ezek a hibák is jól elrejthetők a [5] cikkben javasolt spektrális szűrőkkel. Így ez a stratégia is egy jó választás lehetne, ami további teljesítményjavulást eredményezhetne. Az alkalmazás- és kameraspecifikus paraméterek megfelelő megválasztása esetén ugyanakkor található egy olyan megoldás, ami megfelelően egyensúlyozza a primitívösszevonás hatását a futási időkre és a kimeneti pontosságra.

3.2.6.5. Memóriaigény

Ebben az alfejezetben kiértékelem az általunk javasolt renderelő algoritmus memóriahasználatát. Első lépésként megvizsgáltuk a primitívadatokhoz szükséges tárhelyigényt. Erre a célra egy 5 000 000 primitív tárolására alkalmas, statikusan allokált puffert használtunk, ami 610,35 MB-ot foglalt el. Gyakorlati tapasztalataink alapján ez a mennyiség bőven belefér a gyakorlati alkalmazások által támasztott memóriaigénybe, emiatt egy biztonságos, viszont enyhén pazarló választás. Ám egyértelműen látható, hogy még a pesszimista tárhelyigénybecslés is megfelelő a gyakorlati alkalmazások számára. Kiértékeltük ugyanakkor a primitívek pontos számát mind az öt, a 3.2.6. alfejezetben leírt kamera esetén egy és két fényforrással, valamint eltérő primitívösszevonási stratégiák felhasználásával. Ennek eredményeit a 3.5. táblázatban foglaltuk össze, amik szintén a megközelítésünk használhatóságát támasztják alá valós alkalmazásokban, valamint megmutatják a valós memóriaigényeket realisztikus helyzetekben.

		Nincs összevonás		**Mérsékelt összevonás**		**Nagymértékű összevonás**	
		Primitívek darabszáma	**Memória-igény**	**Primitívek darabszáma**	**Memória-igény**	**Primitívek darabszáma**	**Memória-igény**
1 fényforrás	Heliar Tronnier	122 129	16,77	19 577	2,69	13 808	1,90
	Itoh	210 309	28,88	31 347	4,30	31 011	4,26
	Nikon	665 942	91,45	99 767	13,70	80 021	10,99
	Canon 135mm	562 492	77,25	143 413	19,69	109 120	14,99
	Canon 200mm	1 953 436	268,26	244 855	33,63	204 673	28,11
2 fényforrás	Heliar Tronnier	210 809	28,95	28 877	3,97	23 453	3,22
	Itoh	315 335	43,30	50 657	6,96	50 543	6,94
	Nikon	1 280 183	175,81	188 204	25,85	153 584	21,09
	Canon 135mm	643 200	88,33	148 146	20,34	131 781	18,10
	Canon 200mm	2 721 915	373,80	343 908	47,23	293 430	40,30

3.5. táblázat. Primitívek darabszáma és a szükséges memória (MB) az egyes optikai rendszerek, fényforrásdarabszámok és primitívösszevonási stratégiák esetén.

Algoritmusunk memóriahasználatának másik nagy részét a durva és finom csempepufferek jelentik. Mivel eljárásunk a csempéket a primitívpufferekbe mutató indexek felhasználásával építi fel, ezért a csempepufferek memóriaigénye elhanyagolható és teljességgel alkalmas valós alkalmazásokhoz. Az egyes

durva csempékhez épített 50 000 négyszög tárolására alkalmas puffer létrehozása 257,492 MB GPU memóriát igényel összesen, ami a gyakorlati alkalmazások memóriaigényéhez képest viszonylag alacsony. A 8×8-as méretű sűrű csempék esetén csempénként egy 2 000 primitív tárolására alkalmas puffer allokálása összesen 249,192 MB-ot igényel. Gyakorlati tapasztalataink alapján ezek a memóriaigények messze meghaladják a valósan elfoglalt memóriát mind a durva, mind a sűrű csempékhez rendelt pufferek esetén. A fenti allokációkat egy teljes mértékig biztonságos, viszont túlságosan pesszimista becslésként alkalmaztuk a nem megfelelően megválasztott pufferméretekből fakadó hibák elkerülése érdekében. A sűrű csempék valós memóriaigényét is megvizsgáltuk a 3.5. táblázatban felsorolt esetek mindegyikére, aminek eredményét a 3.6. táblázatban foglaltam össze.

		Nincs összevonás		**Mérsékelt összevonás**		**Nagymértékű összevonás**	
		Primitívek darabszáma	**Memória-igény**	**Primitívek darabszáma**	**Memória-igény**	**Primitívek darabszáma**	**Memória-igény**
1 fényforrás	Heliar Tronnier	332 133	1,27	142 643	0,54	123 146	0,47
	Itoh	540 705	2,06	218 598	0,83	217 006	0,83
	Nikon	2 102 238	8,02	940 987	3,59	846 657	3,23
	Canon 135mm	2 084 646	7,95	1 157 224	4,41	1 031 562	3,94
	Canon 200mm	6 139 602	23,42	2 737 477	10,44	2 520 730	9,62
2 fényforrás	Heliar Tronnier	578 862	2,21	233 763	0,89	211 570	0,81
	Itoh	763 951	2,91	325 343	1,24	324 900	1,24
	Nikon	3 986 146	15,21	1 741 900	6,64	1 575 888	6,01
	Canon 135mm	2 428 032	9,26	1 309 292	4,99	1 234 047	4,71
	Canon 200mm	8 345 476	31,84	3 715 012	14,17	3 462 741	13,21

3.6. táblázat. A kis méretű csempékben tárolt primitívindexek darabszáma és a teljes szükséges memória mérete (MB) az egyes optikai rendszerek, fényforrásdarabszámok és primitívösszevonási stratégiák esetén.

Végezetül fontos megjegyezni, hogy a pufferméretek felső korlátait empirikusan határoztuk meg, amivel biztosítottuk azt, hogy az algoritmusunk képes legyen az összes szellem megjelenítésére nagy sugárrácsméretek és a legösszetettebb optikai rendszerek esetén is. A pufferméretek drasztikusan csökkenthetőek, ha alacsonyabb komplexitású kamerákat választunk (ami sokkal kevesebb szellemet generál), vagy kisebb sugárrácsokat alkalmazunk. Ezenkívül ezek a pesszimista becslések is tökéletesen mutatják, hogy a 8×8-as méretű csempe-

rács alkalmazása is tökéletesen alkalmas az algoritmusnak felhasználói szintű hardvereken történő futtatására, a veszteséges beállítások ellenére is.

3.2.6.6. Baricentrikus koordináták

Tekintve, hogy a baricentrikus koordináták alkalmazása egy hangsúlyos részét képezi a renderelő algoritmusunk számítási költségének, kiértékeltük a különböző (a 2.5. alfejezetben részletezett) baricentrikus számítási módok teljesítményre gyakorolt hatását. Megmértük a csempebejárási fázis hosszát a háromszögekre támaszkodó (alap számítási mód [12] és a homogén koordinátákat alkalmazó [14]), valamint a négyszögekre támaszkodó (általánosított megközelítés [15], Wachspress által javasolt [16], illetve Loop és DeRose által javasolt módosítás [18]) számítási módok alkalmazásával. Egyetlen fényforrást alkalmaztunk bemenetként az összes tesztesetben. Méréseink eredményét a 3.7. táblázatban foglaltam össze.

	Háromszög [12]	**Négyszög [15]**	**Háromszög [14]**	**Négyszög [16]**	**Négyszög [18]**
Heliar Tronnier	0,49	0,44	0,41	0,39	0,36
Itoh	0,76	0,69	0,65	0,61	0,57
Nikon	2,54	2,21	1,99	1,86	1,62
Canon 135mm	2,86	2,49	2,24	2,08	1,82
Canon 200mm	6,40	5,62	4,95	4,63	4,01

3.7. táblázat. Az általunk javasolt eljárás csempebejárási fázisának futási ideje (ms) a különböző baricentrikus számítási megközelítések alkalmazásával.

Méréseinkből egyértelműen látható, hogy a legjobb eredményeket Loop és DeRose, valamint Wachspress megközelítése eredményezte. Látható továbbá, hogy a Skala által javasolt, homogén koordinátákon alapuló számítás szintén jó eredményeket produkál a négyszögeknek háromszögekre történő felosztása ellenére. Ez azzal magyarázható, hogy a 4D vektoriális szorzat GPU-n hatékonyan kiszámítható alacsony számú művelet elvégzésével. Ennek ellenére azonban megfigyelhető, hogy a négyszögalapú megközelítés konzisztensen jobban teljesített minden vizsgált tesztesetben, és további előnyökkel (pl. alacsonyabb számítási- és memóriaigény) is rendelkezik.

Végezetül fontos kiemelni, hogy az általánosított négyszögalapú és az alap háromszögalapú baricentrikus számítás nagyszámú művelet elvégzését igényli, ezért a csempebejárási fázis magasabb számítási költségéhez vezet. Eredményeink viszont egyértelműen megmutatják, hogy a referenciaalgoritmus teljesítményéhez képest saját megközelítésünk jelentősen csökkenti a raszterizáció költségét bármelyik tesztelt megközelítés alkalmazásával (a 3.1. táblázat).

3.3. Szellemek meghatározása polinomillesztéssel

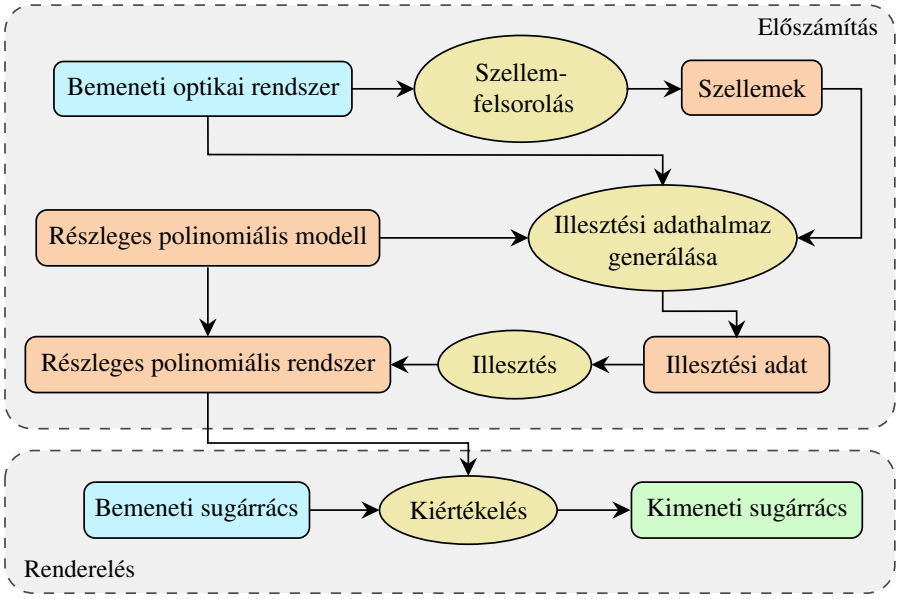
Ahogyan azt már a 3.2.6.2. alfejezetben megjegyeztük, a fényfolt-szimulációs folyamat egyik legköltségesebb fázisa a sugárkövetés. A raszterizációs lépés hatékonyabbá tétele után munkánk második célja a sugárkövetés teljesítményének javítása volt.

A polinomiális optika [6] alkalmazása ígéretes ezen problémák megoldására. Azonban azok a korábbi fényfolt-szimulációs eljárások, amik a polinomiális optika alkalmazását célozzák, mind figyelmen kívül hagyják a becsillanási fényfoltok számos kulcsaspektusát (például a továbbított energia pontos mennyiségét és a helyes takarási viszonyokat), valamint az eredmények precíz validációját. Ezek a hiányosságok azonban erősen korlátozzák a gyakorlati használhatóságot.

Ebben az alfejezetben a 2024-ben megjelent publikációnk [54] alapján mutatom be a saját, polinomiális optikára építkező sugárátviteli modellünket, aminek segítségével az optikai rendszerek becsillanási fényfoltjait szimuláltuk. Eljárásunk segítségével megvalósítható a lencsefényfoltok magas minőségű és valósidejű szimulációja, megfelelően figyelembe véve a jelenség összes fontos tulajdonságát. Az analitikus sugárkövetéshez képest eljárásunk egy kétszeresétől ötszörösig terjedő sebességnövekedést ért el az összes vizsgált esetben, és egy jelentős növekedést mutat fel mind teljesítményben, mind pontosságban a polinomiális optika naiv alkalmazásával szemben.

3.3.1. A mi módszerünk

Munkánk során egy polinomiális optikán alapuló algoritmust építettünk fel, amely a lencsefényfoltok fizikailag megalapozott szimulációjára képes olyan esetekben is, amikor nagyszámú fényvisszaverő felülettel rendelkező, magas komplexitású optikai rendszereket használunk. Módszerünk két fő lépésből épül fel: előszámítás és megjelenítés. Az előszámítási fázisban végezzük el a bemeneti optikai rendszer szellemeire történő ritka polinomok illesztését a megfelelő polinomiális rendszerek meghatározásához. Ehhez egy saját polinomiális modellt alkalmazunk, ami különálló régiókra van osztva, valamint egy kapcsolódó illesztési eljárást. A megjelenítési fázisban az előszámítási lépésben megkapott polinomiális rendszert alkalmazzuk a sugárátvitel meghatározására. A polinomiális rendszer hatékony kiértékelésére GPU-alapú vektorizációt használunk, a valósidejű teljesítmény elérésére pedig a korábban már bemutatott ritka sugárrácsok interpolációjával minimalizáljuk a kiértékelendő sugarak mennyiségét. Algoritmusunk fő lépéseiről egy összefoglaló ábrát láthatunk a 3.8. ábrán.



3.8. ábra. Az általunk létrehozott, polinomiális optikát alkalmazó lencsefényfolt-szimulációs módszer fő lépései.

Módszerünk bemutatásához csak irányított fényforrásokat használunk (tehát párhuzamos beeső fénysugarakkal dolgozunk), ahogyan az a releváns munkákban is történik. Ha a fényforrás nagyon közel van a megfigyelőhöz (tehát a beérkező fénysugarak nem párhuzamosak), a becsillanás mérete és intenzitása tipikusan uralkodó a jelenetben, ezáltal a szellemalakzatok majdnem észrevehetetlenek ebben az esetben. Az ilyen típusú fényforrások jellemzően túl alacsony mennyiségű fényt bocsátanak ki, ami miatt a szellemek szintén közel láthatatlanok. A gyakorlatban a szellemek létrejöttéhez általában a távoli fényforrások a legrelevánsabbak, ahol a beeső fénysugarak közel párhuzamosak.

Ahogyan azt a 2.2. alfejezetben is kiemeltem, munkánk során a sík és szférikus lencseelemeket vettük figyelembe, emiatt azonban feltételezhetjük az optikai rendszer forgásszimmetriáját. Ennek következtében bemeneti paraméterként elegendő csupán egy beesési szöget figyelembe vennünk, így a polinomillesztés ideje jelentősen csökkenthető. Fontos megjegyezni, hogy ezek az egyszerűsítések nem gátolják az eljárásunk alkalmazhatóságát, csupán a modellépítés hosszát és komplexitását hivatottak csökkenteni. Módszerünk triviális módon kiterjeszthető olyan esetekre, amikor az optikai rendszer nem forgásszimmetrikus, valamint a jelenet pontszerű vagy spot fényforrással rendelkezik. Ehhez a polinomiális rendszer paramétereit és a tanítóadatok előállítását szükséges módosítani.

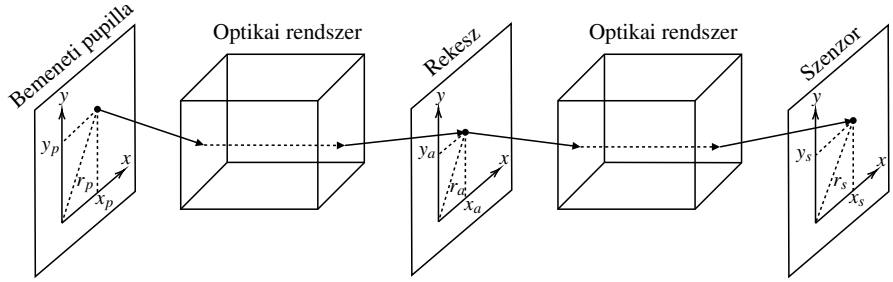
3.3.2. Saját polinomiális modellünk

A polinomiális optika korábbi alkalmazásai leginkább a sugárkövetés offline felhasználására fókuszáltak. Emiatt a létrejövő modellek esetén nem tartották fontosnak a bemeneti változók számának minimalizálását, és azok számos olyan kimeneti változót szolgáltattak, amik a fényfoltok szimulációjához fölöslegesek (elsősorban a kimenő sugárirányt). A korábbi munkák ugyanakkor tipikusan a bemeneti kétdimenziós sugárkoordinátáknak és az irányvektoroknak a közvetlen felhasználására alapoztak, ami viszont a szükséges ritka polinomiális tagok számát szignifikánsan növelheti. A lencsénkénti átvitel, visszaverődés és borítás általi akadályozás helyes modellezése szintén bonyolult ilyen bemeneti változókkal.

Egy, a valósidejű fényfolt-szimulációra alkalmasabb polinomiális modell létrehozásához a következő bemeneti és kimeneti változókat használtuk:

$$(x_p, y_p, r_p, \overline{r_p}, \theta, \lambda) \rightarrow (x_a, y_a, x_s, y_s, \tau_s, \rho_s)\,, \tag{3.4}$$

ahol (x_p, y_p) a sugár pozíciója a bemeneti pupillán, r_p az (x_p, y_p)-hez tartozó polárkoordináták sugárkomponense, $\overline{r_p} = 1 - r_p$, θ a fénysugár beesési szöge, λ pedig a hullámhossz. A kimeneteket illetően (x_a, y_a) és (x_s, y_s) az áthaladó sugár pozíciója a rekesznyíláson és a szenzoron, τ_s és ρ_s pedig a sugárhoz tartozó intenzitás és relatív sugár a szenzoron. A relatív sugár adja meg a végighaladó fénysugár és az optikai tengely közti legnagyobb távolságot, ami a sugár áthaladása során az egyes optikai elemek elérésével tapasztalható, az adott lencse magasságával normalizálva. A 3.9. ábrán láthatjuk a folyamatban részt vevő releváns síkokat és változókat.



3.9. ábra. A saját polinomiális sugárátviteli modellünk által használt fő síkok és koordináta-rendszerek vizualizációja.

A korábbi modellekkel ellentétben, mivel a mi algoritmusunk forgásszimmetriát feltételez, a θ beesési szöget közvetlenül fel tudjuk használni ahelyett, hogy teljes kétdimenziós beesésiirány-vektorokat alkalmaznánk. Ez a megközelítés csökkenti a bemeneti vektorok dimenzióját, ezáltal növeli a polinomkiértékelés teljesítményét. További figyelemre méltó különbség a mi modellünk és a korábbi megközelítések között a sugár- és az inverz sugárkomponensek (r_p és $\overline{r_p}$) bemenetként való felhasználása. Mivel ezeknek a tagoknak a képe általában jelentősen függ a rekesznyílás és az objektív borításának geometriájától, így ezen tagok felhasználásával a polinomjaink kimeneteként a körszerű alakzatok modellezése sokkal precízebb és kevesebb taggal megvalósítható. Mind-

ez közvetlen hatással van a renderelési teljesítményre és kritikus a ρ_s relatív sugár helyes modellezéséhez, ahogyan azt a következőkben ismertetem.

A korábbi munkákhoz hasonlóan [2, 6] a mi polinomiális modellünk is tartalmazza kimenetként a sugár koordinátáit a rekesznyíláson és a szenzoron, ami a végső szellemkoordináták meghatározásához, valamint a rekesz és a szenzor általi akadályozás kezeléséhez szükséges. A korábbi megközelítések megengedték, hogy egy szellemet okozó sugárútvonal többször is keresztülhaladjon a rekesznyíláson [2, 6], emiatt viszont több különböző, a rekesznyílás síkján adott koordináta modellezésére volt szükség a polinomokkal. Ahogyan azt Hullin és mtsai. megjegyezték [5], és ahogyan azt kiemeltem a 2.3. alfejezetben, az ilyen szellemek nagy részét a rekesz akadályozza, ezért a kimenethez való hozzájárulásuk elhanyagolható. Azáltal, hogy módszerünket olyan szellemekre korlátoztuk, amikhez tartozó sugárútvonalak csak egyszer haladnak át a rekesznyíláson, jelentősen csökkentjük a kimeneti változók darabszámát, valamint egyszerűsítjük és gyorsítjuk a sugárkiértékelési folyamatot, az említett korlátozásnak pedig csak minimális hatása van a megjelenítésre.

A polinomiális optika lencsefényfoltok szimulációjára történő korábbi felhasználásai teljesen figyelmen kívül hagyták az egyes elemek Fresnel-féle áteresztő- és visszaverő-képességét (azaz a továbbított és a visszavert energia mennyiségét), legjobb esetben is az első lencsénél vett egyetlen közelítéssel határozták meg az intenzitást [2]. A lencsegyártók jelentős erőfeszítéseket tesznek a szellemkomponensek minimalizálására, amihez speciális, visszaverődést gátló bevonatokat alkalmaznak az egyes lencseelemeken, ezek azonban nagy mértékben befolyásolják a szellemek végső megjelenését. Annak érdekében, hogy tetszőleges lencsét és tükröződésgátló antireflexiós bevonatot helyesen modellezhessünk, egy összesített intenzitástagot vettünk fel a modellünkbe (τ_s). Ezt a tagot a megfelelő Fresnel-transzmissziós és visszaverődési tagok szorzataként kapjuk meg, amely tagokat az optikai rendszer összes fénytörő felületére meghatározunk. Ez az összesített tag kritikus a végső szellemintenzitások hitelességének és valószerűségének biztosításához.

Nagy méretű és magas komplexitású optikai rendszerek esetén az objektív borítása is szignifikánsan hozzájárul a szellem végső alakjához. A korábbi megközelítésekkel ellentétben, ahol a relatív sugár és az elemenkénti borítás

általi akadályozás teljesen figyelmen kívül volt hagyva, a modellünkben lévő ρ_s kimeneti változó lehetővé teszi a borítás általi akadályozás helyes kezelését. Szükséges megjegyezni, hogy mivel a rekesznyílás síkján vett koordinátákat használjuk a rekesznyílás szerinti vágáshoz, a ρ_s érték meghatározása során a rekesznyílást figyelmen kívül hagyjuk, ezáltal csökkentve az adatok komplexitását, amelyekre a polinomokat illesztjük.

A korábbi megközelítések sokszögeket alkalmaztak az objektív rekesznyílásának modellezésére és a sugaraknak a rekesz általi blokkolásának felismerésére. A folyamat egyszerűbbé és gyorsabbá tételére egy rekesznyílást reprezentáló maszkot, valamint hardveresen gyorsított textúra-mintavételezést használtunk azért, hogy meghatározzuk, hogy egy sugár áthalad-e a rekesznyíláson vagy sem. A rekesznyílásmaszkunk pixelenként tartalmazza a legközelebbi rekesznyíláséltől vett előjeles távolságot, amely textúrát a sugárnak a rekesznyílás síkján vett koordinátáit felhasználva mintavételezzük. Az általunk használt megközelítés következményeként képesek vagyunk a rekesznyílás-távolságot közvetlenül felhasználni a rekesznyílás síkján vett koordináták helyett, ily módon pedig a kimeneti változók száma eggyel csökken. A vágási faktor és a relatív sugár paraméterek ugyanakkor egyetlen vágási faktor tagba aggregálhatók, ezáltal kettővel tovább csökkentve a kimeneti paraméterek darabszámát, valamint elkerülve a sugárkövetéses folyamat során a textúra-mintavételezés szükségességét.

A gyakorlatban a valós rekesznyílás-alakzatok matematikailag sokkal komplexebbek, így mindkét megközelítés jelentősen növeli azon polinomiális tagok darabszámát, amik szükségesek a bemeneti adatra való pontos illesztéshez, ezáltal rontva a teljesítményt a bemeneti változók darabszámának csökkentése ellenére. Végezetül a dinamikus, futási idő alatti változtatása a rekesznyílás-alakzatnak szintén lehetetlen ezekkel a megközelítésekkel. Ezen problémák ellenére az alkalmazás tulajdonságaitól függően egyetlen vágási faktor használata egy működőképes megoldás lehet, amivel jelentős javulást érhetünk el a megjelenítés teljesítményét illetően.

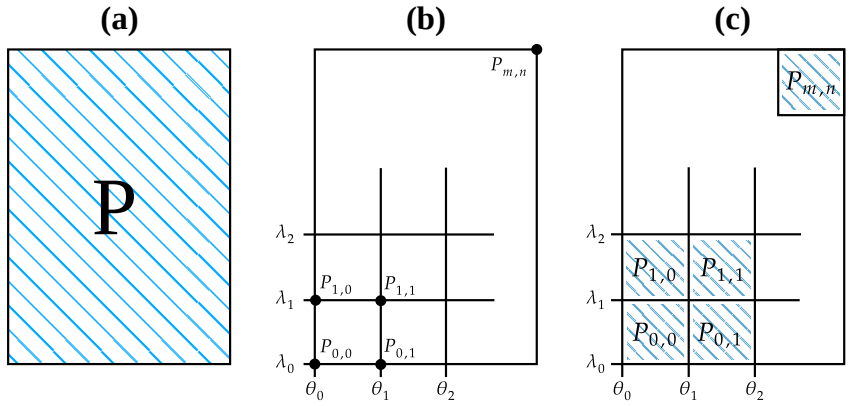
A fenti okokból kifolyólag tesztjeink során hat kimeneti változót használtunk, amelyeket a (3.4) egyenletben soroltam fel. Fontos kiemelni, hogy a rekesznyílás-távolság fent említett közvetlen használata vagy egy kombinált

vágási faktor szintén olyan potenciális megközelítések, amik bármilyen felhasznált optikai rendszer esetén kiértékelhetők.

3.3.3. Részleges illesztési zónák

A (3.4) egyenletben leírt polinomiális fényfolt-reprezentációs modellben egy polinomiális rendszer egy teljes szellem tulajdonságait írja le a beesési szögek és fényhullámhosszok teljes tartománya fölött. Ez a megközelítés egy alkalmas megoldás lehet számos polinomiális optika felhasználás számára (mint ahogyan azt a 3.1.4 alfejezetben bemutatott korábbi munkák demonstrálták), amikor a valós idő nem fontos, vagy a sugárútvonalak kiszámíthatóbbak. Az összesített relatív sugár- és intenzitástagok komplex viselkedése miatt azonban ez a reprezentáció gyakran jelentősen megnövekedett számú polinomiális taggal jár, ezáltal drasztikusan növelve a létrejövő polinomiális rendszerek kiértékelési idejét, mint ahogyan azt a korábbi munkák [2, 6, 48, 49] és a gyakorlati tapasztalataink is megmutatták. A relatív sugár továbbá drasztikus változásokon mehet keresztül a látható régiójának végéhez közeledve (mielőtt az objektív teljesen blokkolná a szellemet), ami nem csak az egyetlen polinomiális rendszerrel történő helyes modellezést teszi bonyolulttá, hanem a létrejövő polinomiális rendszerek általános pontosságára is negatív hatással van.

A fent leírt problémák elkerülése érdekében az egyes hullámhosszokat külön kezeltük, a beesési szögek bemeneti tartományát pedig részleges lokális zónákra osztottuk fel. Egy-egy különálló polinomiális rendszert építettünk fel és illesztettünk az összes diszkrét beesési szög-hullámhossz kombinációra. Az általunk létrehozott, zónaalapú polinomiális reprezentáció egy vizualizációja látható a 3.10. ábrán. A részleges régiók száma egy felhasználó által konfigurálható paraméter, amit a teljes tartomány egyenletes mintavételezésére használunk. A polinomkiértékelés során a köztes tartományokat a szomszédos polinomiális rendszerek lineáris interpolációjával töltjük ki. Ez a megközelítés előnyösebb illesztési tulajdonságokat szolgáltat a lokális régiókban, így csökkenti a szellemalakzat reprezentációjához szükséges polinomiális tagok számát. Továbbá a beesési szög és a hullámhossz-paraméterek konstansok minden egyedi kombináció esetén, így ők elhagyhatók a polinomiális rendszerek bemenetei



3.10. ábra. Egy vizuális összehasonlítása az eredeti teljes polinomiális modellnek (a), a saját részleges modellünknek (b) és a kettő hibrid kombinációjának (c).

közül, a következő redukált formulához vezetve:

$$(x_p, y_p, r_p, \overline{r_p}) \rightarrow (x_a, y_a, x_s, y_s, \tau_s, \rho_s), \tag{3.5}$$

ami jelentősen gyorsabban kiértékelhető, mint a (3.4) egyenletben leírt.

Az egyes polinomiális rendszerekhez szükséges adatmennyiség csökkentésére és a kiértékelés során a szomszédos polinomok hatékony feldolgozására polinombázisok egy egységes rendszerét alkalmaztuk az összes olyan polinom esetén, ami ugyanahhoz a szellemhez tartozik. Polinombázis alatt esetünkben a ritka polinom tagjainak együtthatótól független tényezőit értjük, amelyeket az egy szellemhez tartozó minden polinom esetén egyezőnek választottunk meg. Így csak a polinomtagok együtthatói térnek el a különböző beesési szögek és hullámhosszok esetén. Formálisan az ugyanahhoz a szellemhez tartozó polinom a következőképpen definiálható:

$$p_{mn}(x) = \sum_{k=0}^{t} c_k^{mn} \cdot v_k, \quad v_k = \prod_{j=1}^{d} x_j^{l_{k,j}}, \tag{3.6}$$

ahol m és n a beesési szög és a hullámhosszminták indexei, v_k a megosztott polinomiális bázis, c_k^{mn} pedig a polinom-együttható. Következésképpen az adat mennyisége jelentősen csökken, ami nagy hatással van a polinomkiértékeléshez szükséges memória-sávszélességre. Ez a megközelítés továbbá szignifi-

kánsan növeli az interpolációs sebességet, mivel ugyanaz a megosztott bázis kiértékelhető csupán egyszer sugaranként, és csak a polinomtagok együtthatóit szükséges interpolálni.

Egy hibrid alternatív megoldás lehetne az, ha a lokális polinomiális rendszerek az aktuális és a rákövetkező beesési szögek figyelembevételével lennének felépítve, ezáltal elkerülve az interpolációt és csökkentve a kiértékeléshez szükséges műveletek számát. A különböző reprezentációk közötti különbséget vizualizálja a 3.10. ábra. Ennek a hibrid megközelítésnek a fő hátránya az, hogy komplexebb polinomiális rendszerekhez vezet, ami viszont növeli a pontos polinomiális reprezentációhoz szükséges tagok számát. Ez a felépítés továbbá szükségessé teszi a (3.4) egyenletben leírt teljes polinomiális rendszer használatát, ami negatív hatással van a kiértékelés költségére. Gyakorlati tapasztalataink azt mutatják, hogy ezeknek a szempontoknak a kombinált hatása magasabb számítási költséghez vezet, mint a saját eljárásunk. Emiatt az interpolációalapú megközelítés ideális a polinomkiértékelés teljesítményének maximalizálására.

3.3.4. Illesztési adatok létrehozása

A Hullin és mtsai. [6] által javasolt polinomiális eszköztár legfőbb hátránya, hogy az elemenkénti közelítési hibát végigáramoltatja a teljes rendszeren, ezáltal jelentősen csökkentve a létrejövő polinomok pontosságát. Az elemenkénti polinomok ugyanakkor csak analitikus módon, összetett egyenletek gyökeinek megtalálásával kaphatók meg, amihez költséges eszközök használata szükséges. További megjelent megközelítések [37, 45, 46] tanító adathalmazok használatával feloldják ezt a korlátozást, valamint olyan illesztési módszereket adnak, amikkel a polinomok a teljes optikai rendszerre meghatározhatók. Mivel ez a megközelítés jobb eredményekhez vezet, mi is az illesztést választottuk a kívánt polinomiális rendszerek előállítására.

Annak érdekében, hogy elvégezzük a 3.3.3. alfejezetben bemutatott saját részleges polinomiális modellünkkel való illesztést, első lépésként az illesztési adatokat hoztuk létre. Egy alkalmas adatgenerálási megközelítés létrehozása elengedhetetlen volt, mivel a nem megfelelő adat felhasználása drasztikusan

csökkentheti az eljárás pontosságát. A felhasználandó adat minősége kritikus a létrejövő polinomiális rendszer pontossága és a kívánt kimeneti pontosság eléréséhez szükséges polinomok száma miatt. Így tehát az illesztési adathalmaz létrehozása kiemelt figyelmet kapott azért, hogy ezeket a célokat elérjük vele. A saját adatgenerálási megközelítésünk lépéseit a 4. algoritmusban foglaltuk össze. Koncepcionálisan az adathalmaz-létrehozás egy olyan lépéssorozatból épül fel, amit az összes szellem, hullámhossz és beesési szög kombinációra végrehajtunk. Eljárásunk leírása során egy ilyen paraméterhármashoz generált adathalmaz részleteit mutatom be.

4. algoritmus. Az adatillesztéshez létrehozott metódusunk.

Bemenet: Optikai rendszer paraméterei ($\mathscr{O}$),
hullámhosszak (Λ),
maximális beesési sugarak ($\overline{\theta}$),
beesési szögek száma (n_θ),
határoló és mintavételezési sugarak száma (n_b, n_s),
érvényes és érvénytelen kimeneti minták (n_v, n_i)

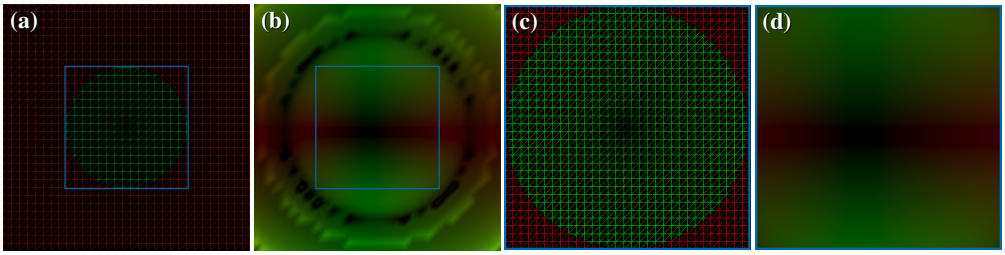
Kimenet: Illesztett adathalmaz (D)

$D \leftarrow \{\}$
$G \leftarrow \text{SzellemekEmumeracioja}(\mathscr{O})$
$\Theta \leftarrow \text{SzogekMintavetelezese}(n_\theta, [0, \overline{\theta}])$
foreach $(g, \theta, \lambda) \in G \times \Theta \times \Lambda$ **do**
  $R_p \leftarrow \text{SugarMeghatarozasaPupillan}(\mathscr{O}, n_b)$
  $R_s \leftarrow \text{SugarakKovetese}(\mathscr{O}, R_p, g, \theta, \lambda)$
  $b \leftarrow \text{ErvenyesSugarhatarokMeghatarozasa}(R_s)$
  $R_p^b \leftarrow \text{HataroltSugarakGeneralasa}(b, n_s)$
  $R_s^b \leftarrow \text{SugarakKovetese}(\mathscr{O}, R_p^b, g, \theta, \lambda)$
  $\hat{D} \leftarrow \text{SugarakUtofeldolgozasa}(R_s^b)$
  $D_v, D_i \leftarrow \text{ErvenyessegSzerintiFelosztas}(\hat{D})$
  $D_v^s \leftarrow \text{VeletlenszeruEgyenletesMintak}(D_v, n_v)$
  $D_i^s \leftarrow \text{VeletlenszeruEgyenletesMintak}(D_i, n_i)$
  $D \leftarrow D \cup D_v^s \cup D_i^s$

Az adatgenerálás érdekében egy durva sugárráccsal végrehajtott analitikus sugárkövetést hajtunk végre az optikai rendszer fizikai struktúráját felhasználva. Első lépésként meghatározzuk az optikai rendszer bemeneti pupillájának

a tengelyekkel párhuzamos határoló négyszögét, ami szorosan behatárolja azt a régiót, ami a sugarak kiindulásához felhasználható. Ezen régió meghatározásának fő célja az, hogy az adathalmazt olyan sugarakra szűkítsük, amiknek megvan a lehetőségük, hogy elérjék a rendszer szenzorát, tehát nincsenek blokkolva semmilyen akadály által (objektív borítása vagy rekesz), nem mennek keresztül teljes belső visszaverődésen és elegendő energiát hordoznak a folyamat végére. A továbbiakban ezeket a sugarakat tekintjük érvényesnek.

Ezen határoló régió meghatározására egy előzetes lépésként egy sűrű sugárrácsot követünk végig az optikai rendszeren, és meghatározzuk azt a tengelyekkel párhuzamos oldalakkal rendelkező határoló négyszöget, ami azt a régiót határolja, ahol a fent leírt tulajdonsággal rendelkező sugarak vannak jelen. Ezen régiónak a létrejövő sugárkövetett tanítóadatra gyakorolt hatása látható a 3.11. ábrán.



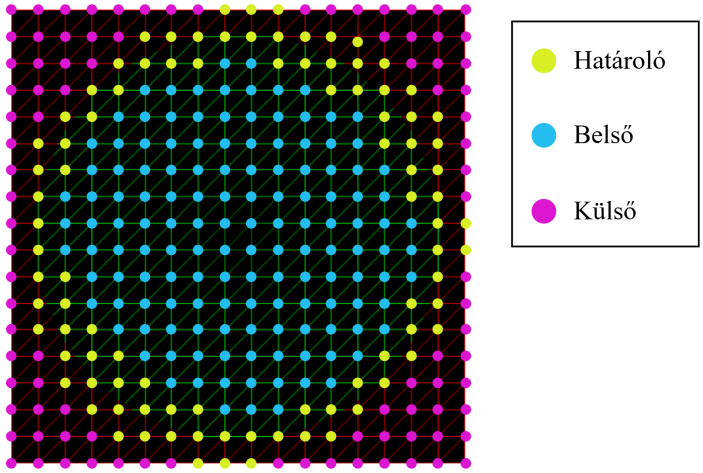
3.11. ábra. (a, b) A sugárkövetéshez használt sugarak vizualizációja az első lencse síkján, valamint a kimeneti változók a kékkel jelölt határoló régióval. (c, d) Csak a határoló régióban sugárkövetett sugarak és a célváltozók vizualizációja.

Ha a tesztelt rácsban nincs jelen egyetlen érvényes sugár sem, megnézzük a szomszédos beesési szögeket. Ha így legalább egy érvényes szomszéd található, akkor az ő határoló régióját használjuk fel. Ha egyetlen beesési szög sem eredményez érvényes sugarakat a közelben, a beesési szöget érvénytelennek jelöljük, és nem adunk vissza határoló régiót. Ez a lépés elengedhetetlen az olyan beesési szögeknél megjelenő vizuális műtermékek elkerüléséhez, ahol a szellem láthatósága megváltozik.

A határoló régió meghatározása után végrehajtjuk a végső sugárráccsal a sugárkövetést (felhasználó által paraméterezhető rácsmérettel), aminek eredményeként egy kezdeti illesztési adathalmazt kapunk. Ezt követően utófeldol-

gozó lépések egy sorozatát végezzük el a végső kimeneti illesztési adathalmaz létrehozására. Az első utófeldolgozó lépésként az összes sugárról eldöntjük, hogy ő a szellem tekintetében belső, külső vagy határoló eleme-e a sugárrácsnak, amihez a rácson az összes szomszédos sugarat érvényesnek (belső), érvénytelennek (külső) vagy hibridnek (határoló) jelöljük meg. Erre a sugárkategorizálásra egy példát láthatunk a 3.12. ábrán.

Ezt követően a generált adatokat egy érvényes és egy érvénytelen részhalmazra bontjuk, ahol a határoló sugarakat a saját érvényességük alapján soroljuk valamelyik részhalmazba. Végezetül az érvényes és érvénytelen halmazok létrehozásához egyenletes eloszlású véletlen mintavételezést alkalmaztunk, amiből a kimeneti illesztési adathalmaz a két mintavételezett adathalmaz egyesítésével áll elő. Az egyenletes eloszlású véletlen mintavételezés természetéből fakadóan az adathalmaz megtartja a kiindulási adathalmaz eredeti tulajdonságait bármilyen szignifikáns információvesztés nélkül, ezáltal a reprezentativitás sértetlen marad. Az illesztési adat két eltérő, érvényes és érvénytelen részadathalmazból való létrehozása garantálja a két régió megfelelő reprezentációját a mintavételezett adathalmazban, ami akkor fontos, ha két régió eltérő eloszlásokkal rendelkezik (például a periférikus területeken). Az érvénytelen régiók megfelelő reprezentációja lényeges a kielégítő illesztési minőség eléréséhez és a megjelenítés során felmerülő interpolációs műtermékek elkerüléséhez.



3.12. ábra. Belső, külső és határoló sugarak vizualizációja.

3.3.5. Polinomillesztés

A részleges polinomiális modell és az illesztési adatok birtokában elvégezhető a polinomillesztés. Ez a folyamat hozza létre azokat a polinomokat, amik az optikai rendszer szellemeihez tartozó sugárátvitelt írják le. Erre a célra számos különböző eljárás érhető el a szakirodalomban, amik jelentősen eltérő számítási időkkel és illesztési pontosságokkal rendelkeznek. Az elérhető eljárások közül munkánk során kiindulási pontként a [46] cikkben leírt megközelítést használtuk az alacsony lépésszámmal történő pontos eredmények létrehozásának képessége miatt. Ezt az algoritmust aztán úgy egészítettük ki, hogy képes legyen egyszerre meghatározni a részleges modell megosztott polinomiális bázisait és a lokális zónákhoz tartozó együtthatókat. Következő lépésként az eljárást úgy módosítottuk, hogy a kívánt pontosság elérését követően minimalizálja a polinomtagok számát.

Az illesztést minden egyedi szellem, hullámhossz és kimeneti sugárjellemző kombinációra elvégezzük. A folyamatot egy olyan polinommal kezdjük, ami csupán egyetlen tagot tartalmaz, amire aztán iteratívan alkalmazzuk a kiterjesztési és csökkentési operátorokat a [46] cikkben javasoltak szerint a polinom finomítása érdekében. A folyamat egy fontos lépése az aktuális polinomba való új tag beszúrása, ha az ezzel kapott hibacsökkentés jelentős (egy adott határérték fölött van) és a polinomtagok száma kellően alacsony. A hibacsökkenés és a polinomtagok számának határa felhasználó által konfigurálható paraméterek. A folyamatot mindaddig végrehajtjuk, amíg egy abszolút hibát el nem érünk. Kísérleteink azt mutatják, hogy a hibának a megjelenítés minőségére gyakorolt hatása elhanyagolható egy bizonyos határérték alatt, a hozzáadott polinomtagok viszont növelik a polinomiális rendszer kiértékelésének költségét. Emiatt a polinomtagok számának minimalizálása elengedhetetlen a megfelelő megjelenítési teljesítmény eléréséhez.

Az operátorok által alkotott lehetséges új állapot kiértékelésére egy egyedi hibafüggvényt és egy aggregációs operátort használunk, hogy meghatározzuk a polinom átfogó pontosságát az egyes lokális illesztési zónák fölött. Az aggregációs operátor fő célja, hogy egy olyan hibaértéket szolgáltasson, ami megfelelően leírja a polinomoknak a lokális zónákban megfigyelt átfogó viselkedé-

sét. Emiatt egy adott állapot hibájának meghatározására az átlagos négyzetes hibaoperátort (MSE) használjuk az alrégiók hibafüggvényeivel:

$$\Omega(\Theta) = \frac{\sum_{\theta \in \Theta} (\omega(\theta))^2}{|\Theta|}, \tag{3.7}$$

ahol Θ az illesztési adathalmazok uniója az összes beesési szögre, θ egy adott beesési szöghöz tartozó illesztési adathalmaz, ω pedig a hibafüggvény egyetlen lokális illesztési zónához, ami a következőképpen írható fel formálisan:

$$\omega(\theta) = \frac{\sum_{r \in \theta} w(r) \cdot (\hat{y}(r) - y(r))^2}{|\theta|}. \tag{3.8}$$

A fenti formulában r egyetlen illesztési adatelem, $\hat{y}(r)$ és $y(r)$ a valós (sugárkövetett) és generált (polinomiális rendszer felhasználásával) értékei az r minta kimeneti változóinak, $w(r)$ pedig r súlya, amit a következőképpen definiálunk:

$$w(r) = \begin{cases} w_r, \text{ ha } y_{min} \leq y(r) \leq y_{max} \\ 1, \text{ egyébként} \end{cases}, \tag{3.9}$$

ahol w_r egy felhasználó által konfigurálható paraméter, ami az r érvényességétől függ, y_{min} és y_{max} pedig az y célváltozó legkisebb és legnagyobb érvényes értékei. Kísérleteink során a következő w_r súlyokat alkalmaztuk: $w_i = 1$ a belső, $w_b = 1$ a határoló és $w_e = 0$ a külső sugarakhoz.

Az általunk javasolt hibafüggvény fókuszában a polinomoknak a belső és külső régiókra történő pontos illesztése áll, ahol az eredmények jelentősen hozzájárulnak a szellemkomponensek látható részeihez. A 3.9. formulában leírt súlyozási sémánk ugyanakkor garantálja azt, hogy a külső régiókban lévő láthatatlan adatelemek csak akkor járulnak hozzá az illesztési hibához, ha ők hibás értékeket generálnának a kiértékelés során, ezáltal növelve az illesztési folyamat átfogó robusztusságát. Mivel az analitikus sugárkövetésből fakadó, matematikailag folytonos értékek viselkedése nem előrejelezhető és nagyon változékony ezekben a régiókban, így egy abszolút pontos illesztés a folyamatot sokkal komplexebbé tenné anélkül, hogy jelentős előnyökkel járna. Mivel a szellem ezekben a régiókban láthatatlan, így tulajdonságainak a pontos értékei irrelevánsak. Az egyetlen fontos szempont az, hogy a polinomok által

előállított értékek vágási határértékekhez való viszonya megegyezzen a tanító-adatokban. Az általunk javasolt hibafüggvény és súlyozási séma kielégíti ezt a feltételt, mivel érvénytelen minták csak akkor járulnak hozzá a hibához, ha az optimalizálás alatt lévő polinomiális modellel generált sugár jellemzői számottevő hibát okoznának a kiértékelés során.

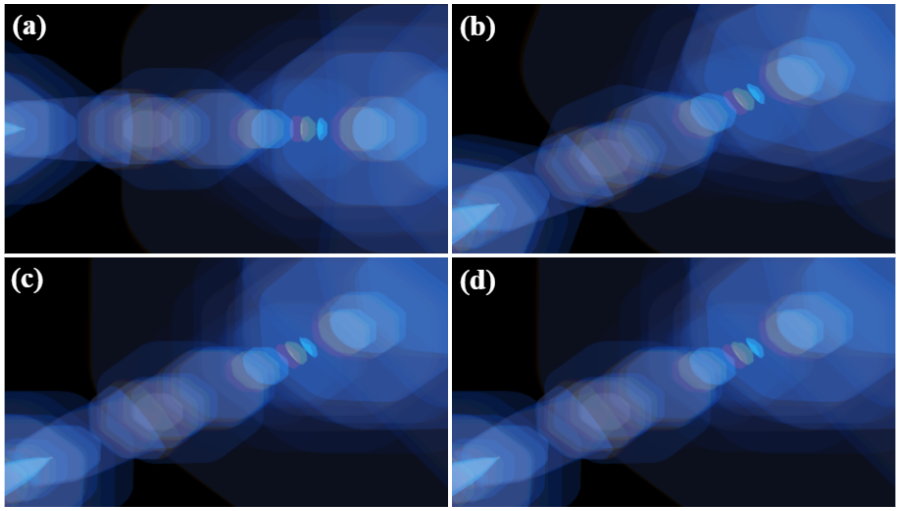
3.3.6. Polinomkiértékelés

A polinomok sikeres illesztése után következik a megjelenítési fázis a GPU-n végrehajtva. A korábbi, polinomiális optikát alkalmazó fényfolt-szimulációs eljárások sűrű sugárkövetést alkalmaztak, ami valósidejű alkalmazásokban történő felhasználásra alkalmatlanná teszi ezeket a megközelítéseket. Ennek a korlátozásnak a feloldására Hullin és mtsai. [5] algoritmusát használtuk fel. Ahogyan azt már korábban megjegyeztük, csupán egy ritka sugárrácsot követünk végig az optikai rendszeren, majd hardveresen gyorsított interpolációval kitöltjük a réseket. Esetünkben a fő eltérés a sugárkövetéses folyamatban rejlik. A lencsefelület-elemek iteratív módon, analitikus metszési formulákkal történő kiértékelése helyett a sugarakat egy lépésben transzformáljuk a bemeneti pupilláról a szenzorra az illesztett polinomok segítségével.

Ahogyan azt a 3.3.3. alfejezetben említettem, modellünk diszkrét horizontális beesési szögeket használ. Ebből kifolyólag a köztes szögek kezelésére a forgásfüggő c_k^{mn} polinomiális együtthatókat interpoláljuk, amihez a fényfoltot generáló irányított fényforrás beesési szögét használjuk és kiértékeljük a polinomiális modellt az interpolált együtthatókkal.

Az optikai tengely körüli forgatás kezeléséhez a polinomkiértékelésből származó sugárrácsot képtérben elforgatjuk a megfelelő tengely körüli forgatási értékkel. Ez a folyamat azonban nem megfelelően transzformálja a textúrakoordinátateret, hibásan forgatott rekesznyílás-alakzatokhoz vezetve ezáltal. Ezen probléma megoldására egy tengely körüli inverz forgatást alkalmazunk a polinomiális rendszer által generált textúrakoordinátákra, ami kiküszöböli a sugárrács transzformációja által bevezetett forgatást. A probléma és a javasolt megoldásunk vizualizációja látható a 3.13. ábrán.

A számítások hatékony elvégzése és a szükséges műveletek számának mi-



3.13. ábra. A végső sugárkövetett szimuláció eléréséhez használt lépések vizualizációja. (a) A polinomkiértékelés kimenete. (b) Az eredmények optikai tengely körüli elforgatása. (c) Inverz módon forgatott textúrakoordináták, amik a végső kimenetet szolgáltatják. (d) Sugárkövetett referencia.

nimalizálása kritikus a GPU-n történő renderelés optimális teljesítményének eléréséhez. Ezért számos lépést tettünk annak érdekében, hogy biztosítsuk azt, hogy a polinomok kiértékelésének költsége alkalmas legyen valósidejű alkalmazásokhoz. Először is meghatározzuk a releváns polinomiális együtthatókat, amik hozzájárulnak az aktuális képkockához, és eltároljuk őket a megosztott GPU memóriában. Ennek eredményeként a gyorsítótárazott polinomok elérésének ideje szignifikánsan csökkent. Továbbá natív GPU vektorizációt alkalmaztunk a különböző kimeneti változókhoz tartozó polinomiális rendszerek párhuzamos kiértékelésére, ami a GPU erőforrások hatékonyabb kihasználását eredményezte, és ezáltal a kiértékelési folyamat jelentősen hatékonyabbá vált.

3.3.7. Eredmények

3.3.7.1. Tesztbeállítások

Az általunk javasolt módszer kiértékeléséhez létrehoztunk egy referenciaimplementációt C++ programozási nyelven és az OpenGL grafikus könyvtár felhasználásával. Az illesztési adatokat GPU-alapú analitikus sugárkövetéssel ge-

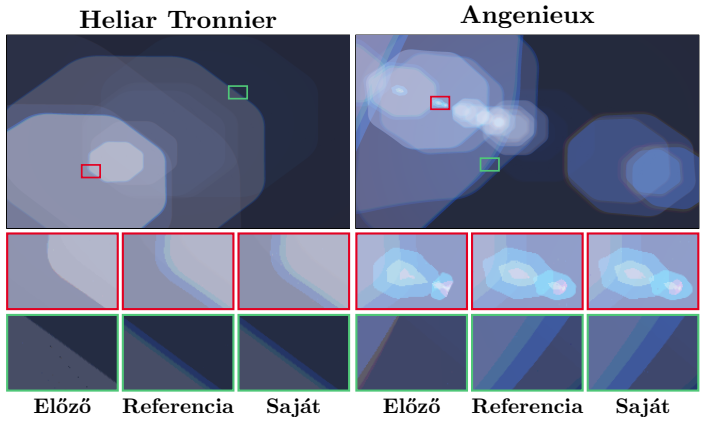
neráltuk, amihez a polinomillesztést CPU-n implementáltuk, és többszálasítást alkalmaztunk. A polinomkiértékelést és a megjelenítést a GPU-n hajtottuk végre a 3.2. alfejezetben bemutatott csempézett raszterizációs megközelítésünkkel.

Szintén elkészítettünk egy implementációt a direkt analitikus sugárkövetéshez, ahogyan azt Hullin és mtsai. [5] javasolták, amit a referenciakimenetek generálására és a módszerünk teljesítményjavításának kiértékelésére használtunk. Végezetül létrehoztuk a naiv polinomillesztés [6, 49] implementációját a (3.4) egyenletben leírt polinomiális rendszerrel, ahol minden szellemet egy különálló polinomiális rendszerrel reprezentáltunk, és nincsenek lokális illesztési zónák felhasználva. A két polinomiális megközelítés vizualizációja a 3.10. ábrán látható a 3.3.3. alfejezetben. Megjegyezzük, hogy az eredeti polinomiális optikát alkalmazó módszerek figyelmen kívül hagyták a relatív sugár- és az intenzitástagokat. Ennek ellenére a naiv megközelítés implementációja során ezt a két változót is a modell részévé tettük, hogy a részleges illesztési eljárásunk kiértékeléséből fakadó javulásokra fókuszálhassunk. A valóságban az előző eljárás és a mi módszerünk közötti különbség jelentősebb lenne, ha figyelmen kívül hagytuk volna az említett két attribútumot a vizsgálataink során, viszont ez nehezebbé tette volna a részleges polinomillesztési megoldásunk hatásának precíz kiértékelését.

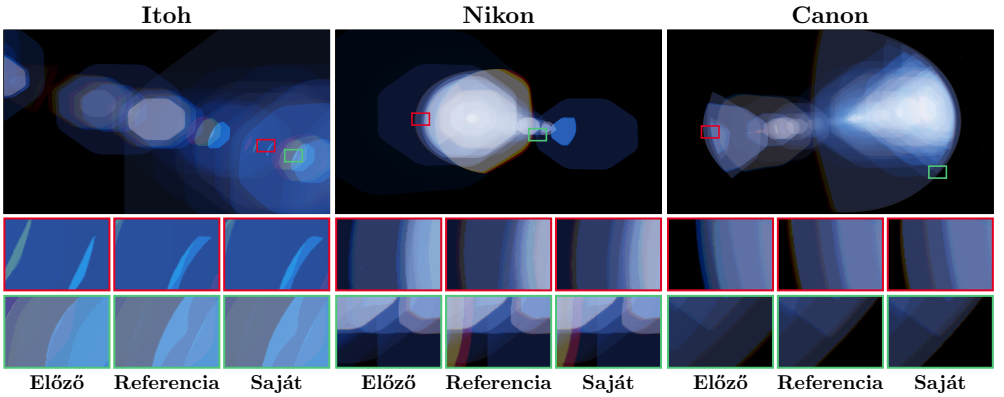
Az általunk javasolt módszer teljesítményének kiértékelésére öt különböző optikai rendszert alkalmaztunk, amik eltérő komplexitással rendelkeznek. Ezen optikai rendszereket az OpenLensFlare keretrendszer [10] példái közül emeltük ki:

- Egy Heliar Tronnier objektív (USP 2645156, f/16, 100 mm fókusztávolság, 8 fénytörő felület, 13 szellem).
- Egy Angenieux Double-Gauss objektív (USP 2701982A, f/11, 100 mm fókusztávolság, 14 fénytörő felület, 42 szellem).
- Egy Itoh zoomobjektív (USP 4196968, f/32, 146 mm fókusztávolság, 18 fénytörő felület, 73 szellem).
- Egy Nikon telezoom objektív (S.53-131852, f/22, 200 mm fókusztávolság, 21 fénytörő felület, 142 szellem).
- Egy Canon telezoom objektív (USP 5537259, f/22, 135 mm fókusztávolság, 33 fénytörő felület, 312 szellem).

A szimulált kimenetek a 3.14. és a 3.15. ábrán láthatók az összes optikai rendszer és renderelési algoritmus felhasználásával. A tesztelésre használt hardver egy AMD Ryzen 7 1700 3,00 GHz-es CPU-t és egy NVIDIA TITAN Xp grafikus kártyát tartalmazott, az összes mérés ezek felhasználásával készült.



3.14. ábra. Lencsefényfolt-szimuláció az alacsony komplexitású kamerarendszerek esetén a teljes polinomiális modellt használva (Előző), analitikus sugárkövetést (Referencia) és a saját részleges polinomiális modellünket alkalmazva (Saját).



3.15. ábra. Lencsefényfolt-szimuláció a közepes és magas komplexitású kamerarendszerek esetén a teljes polinomiális modellt használva (Előző), analitikus sugárkövetést (Referencia) és a saját részleges polinomiális modellünket alkalmazva (Saját).

3.3.7.2. Modellillesztési teljesítmény

Először kiértékelem a naiv és a saját polinomiális modellünk illesztési tulajdonságait. Ehhez megmértük a teljes illesztési időt, a létrejövő polinomtagok átlagos számát és az átlagos illesztési hibát az összes kimeneti változó és optikai rendszer esetén. Mindkét polinomiális beállítással három különböző hullámhosszt és 241 beesési szöget alkalmaztunk, 0,5°-os lépésközzel mintavételezve a $\theta \in [0°, 60°]$ tartományt. 80×80 méretű sugárrácsokat és 3200 mintát használtunk az illesztési adat generálásához, amit egyenletes eloszlású véletlen mintavételezéssel kaptunk a létrejövő adathalmazokból, 1:1 felosztással érvényes és érvénytelen bejegyzésekre. A polinomiális tagok maximális száma öt volt a kisebb (Heliar Tronnier, Angenieux, Itoh) és tíz a nagyobb (Nikon, Canon) optikai rendszerek esetén. Az eredményeket a 3.8. és a 3.9. táblázat foglalja össze.

		Heliar Tronnier			Angenieux		
Változó	**Módszer**	**Idő**	**Tagok**	**Hiba**	**Idő**	**Tagok**	**Hiba**
x_a	Előző	0,16 h	5,00	$1{,}30 \cdot 10^{-1}$	0,40 h	5,00	$2{,}91 \cdot 10^{-2}$
	Saját	0,15 h	4,69	$7{,}22 \cdot 10^{-4}$	0,20 h	4,95	$3{,}03 \cdot 10^{-4}$
y_a	Előző	0,18 h	5,00	$2{,}80 \cdot 10^{-3}$	0,39 h	5,00	$1{,}96 \cdot 10^{-3}$
	Saját	0,07 h	4,54	$1{,}14 \cdot 10^{-4}$	0,09 h	4,79	$7{,}41 \cdot 10^{-5}$
x_s	Előző	0,14 h	5,00	$5{,}40 \cdot 10^{-1}$	0,38 h	5,00	$1{,}42 \cdot 10^{-1}$
	Saját	0,15 h	5,00	$1{,}50 \cdot 10^{-2}$	0,29 h	5,00	$3{,}95 \cdot 10^{-3}$
y_s	Előző	0,19 h	5,00	$4{,}80 \cdot 10^{-2}$	0,41 h	5,00	$2{,}11 \cdot 10^{-2}$
	Saját	0,15 h	5,00	$2{,}90 \cdot 10^{-3}$	0,18 h	5,00	$1{,}22 \cdot 10^{-3}$
τ_s	Előző	0,10 h	4,69	$6{,}90 \cdot 10^{-6}$	0,21 h	4,76	$6{,}54 \cdot 10^{-6}$
	Saját	0,11 h	3,54	$1{,}24 \cdot 10^{-7}$	0,16 h	2,86	$8{,}70 \cdot 10^{-8}$
ρ_s	Előző	0,16 h	5,00	$3{,}04 \cdot 10^{-2}$	0,38 h	5,00	$3{,}73 \cdot 10^{-2}$
	Saját	0,18 h	5,00	$4{,}92 \cdot 10^{-3}$	0,33 h	5,00	$4{,}07 \cdot 10^{-3}$
Polinom	Előző	0,97 h	4,95	$1{,}30 \cdot 10^{-1}$	2,29 h	4,96	$3{,}86 \cdot 10^{-2}$
	Saját	0,85 h	4,63	$3{,}95 \cdot 10^{-3}$	1,36 h	4,60	$1{,}60 \cdot 10^{-3}$

3.8. táblázat. Számítási idők (*Idő*), polinomtagok átlagos száma (*Tagok*), átlagos hibák (*Hiba*) az összes kimeneti változóra és a teljes polinomiális rendszerre megmérve az alacsony komplexitású vizsgált optikai rendszerek esetén a naiv (*Előző*) és a saját részleges (*Saját*) polinomiális megközelítésünkkel.

Ahogyan az a mérésekből is látható, módszerünk magas illesztési pontosságot ér el az összes kimeneti változó és optikai rendszer esetén. A naiv meg-

		Itoh			Nikon			Canon		
Változó	**Módszer**	**Idő**	**Tagok**	**Hiba**	**Idő**	**Tagok**	**Hiba**	**Idő**	**Tagok**	**Hiba**
x_a	Előző	0,12 h	5,00	$1{,}18 \cdot 10^{-2}$	3,06 h	10,00	$1{,}05 \cdot 10^{-2}$	4,61 h	9,95	$8{,}16 \cdot 10^{-3}$
	Saját	0,18 h	3,82	$4{,}13 \cdot 10^{-5}$	0,30 h	5,77	$1{,}83 \cdot 10^{-4}$	0,44 h	5,92	$7{,}88 \cdot 10^{-4}$
y_a	Előző	0,10 h	4,96	$1{,}20 \cdot 10^{-3}$	1,85 h	8,32	$1{,}89 \cdot 10^{-3}$	2,61 h	8,96	$1{,}17 \cdot 10^{-3}$
	Saját	0,01 h	3,66	$2{,}67 \cdot 10^{-5}$	0,14 h	4,97	$4{,}00 \cdot 10^{-5}$	0,21 h	4,77	$1{,}40 \cdot 10^{-4}$
x_s	Előző	0,13 h	5,00	$1{,}31 \cdot 10^{-1}$	3,21 h	10,00	$6{,}82 \cdot 10^{-2}$	4,26 h	10,00	$3{,}62 \cdot 10^{-2}$
	Saját	0,07 h	4,97	$1{,}83 \cdot 10^{-4}$	0,99 h	7,30	$9{,}64 \cdot 10^{-3}$	1,09 h	7,59	$3{,}73 \cdot 10^{-3}$
y_s	Előző	0,15 h	5,00	$1{,}35 \cdot 10^{-2}$	3,28 h	9,78	$8{,}76 \cdot 10^{-3}$	4,33 h	9,92	$7{,}65 \cdot 10^{-3}$
	Saját	0,03 h	4,93	$2{,}23 \cdot 10^{-5}$	0,53 h	6,36	$1{,}30 \cdot 10^{-3}$	0,56 h	6,48	$7{,}01 \cdot 10^{-4}$
τ_s	Előző	0,08 h	4,33	$1{,}69 \cdot 10^{-6}$	0,70 h	5,58	$1{,}68 \cdot 10^{-6}$	0,89 h	5,08	$7{,}23 \cdot 10^{-6}$
	Saját	0,02 h	1,44	$2{,}29 \cdot 10^{-8}$	0,12 h	1,80	$5{,}43 \cdot 10^{-8}$	0,21 h	1,73	$4{,}13 \cdot 10^{-8}$
ρ_s	Előző	0,14 h	5,00	$2{,}39 \cdot 10^{-2}$	3,20 h	10,00	$5{,}42 \cdot 10^{-2}$	4,68 h	10,00	$2{,}55 \cdot 10^{-2}$
	Saját	0,11 h	5,00	$5{,}43 \cdot 10^{-4}$	1,46 h	8,13	$9{,}76 \cdot 10^{-3}$	2,92 h	9,32	$2{,}36 \cdot 10^{-3}$
Polinom	Előző	0,77 h	4,88	$3{,}01 \cdot 10^{-2}$	15,54 h	8,95	$2{,}40 \cdot 10^{-2}$	21,73 h	8,99	$1{,}32 \cdot 10^{-2}$
	Saját	0,30 h	3,97	$1{,}36 \cdot 10^{-4}$	3,76 h	5,72	$3{,}49 \cdot 10^{-3}$	5,78 h	5,97	$1{,}29 \cdot 10^{-3}$

3.9. táblázat. Számítási idők (*Idő*), polinomtagok átlagos száma (*Tagok*), átlagos hibák (*Hiba*) az összes kimeneti változóra és a teljes polinomiális rendszerre megmérve a közepes és magas komplexitású vizsgált optikai rendszerek esetén a naiv (*Előző*) és a saját részleges (*Saját*) polinomiális megközelítésünkkel.

közelítéshez képest a mi módszerünk által generált hiba legalább egy nagyságrenddel alacsonyabb az összes optikai rendszerre, jelentősen magasabb megfigyelhető nyereséggel az egyes kimeneti változókra. Ez a javított pontosságot ugyanakkor sokkal kevesebb polinomiális taggal értük el, ami az optimális, valósidejű renderelési teljesítmény eléréséhez kritikus.

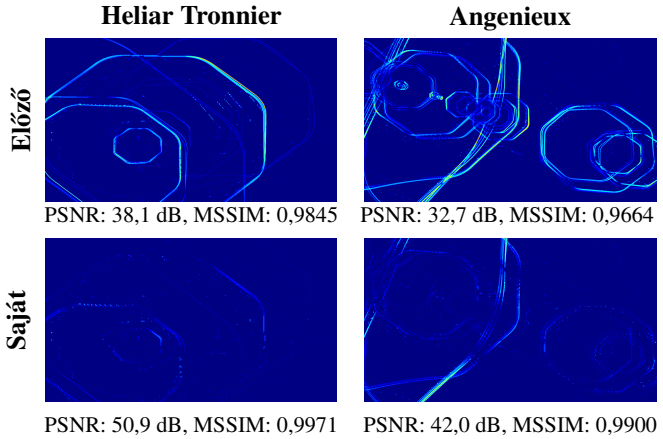
Az illesztési teljesítményt illetően a teljes számítási idő skálázódik az optikai rendszer komplexitásának növekedésével, ami várható, tekintve a rendszerhez tartozó szellemek darabszámának (ezáltal az elvégzendő illesztés mennyiségének) növekedését a komplexitással együtt. Algoritmusunk esetén az illesztés időtartama pár perctől (Heliar Tronnier) pár óráig (Canon Zoom) terjedhet. Véleményünk szerint ez a teljesítmény tökéletesen alkalmas valós alkalmazásokhoz, mivel az illesztést csupán egyszer szükséges végrehajtani, annak eredménye újrafelhasználható az alkalmazásban.

A naiv megközelítést illetően az illesztés teljes ideje az összes tesztesetben magasabb, és majdnem négyszer több időt vesz igénybe a legkomplexebb esetben (Canon Zoom). Ezen viselkedés fő oka a polinomiális tagok számában rejlik. A polinomiális regressziós algoritmus [46], ami az illesztési eljárásunk alapját szolgáltatja, az új állapotjelöltek generálását a polinom aktuális tagjain

alkalmazott bővítési és szűkítési operátorokkal végzi, ami a számítás exponenciális növekedését eredményezi a polinom méretének növekedésével. Emiatt a polinom méretének csökkentése kritikus a megjelenítési teljesítmény szempontjából és fontos szerepe van az illesztési idő alacsonyan tartásában.

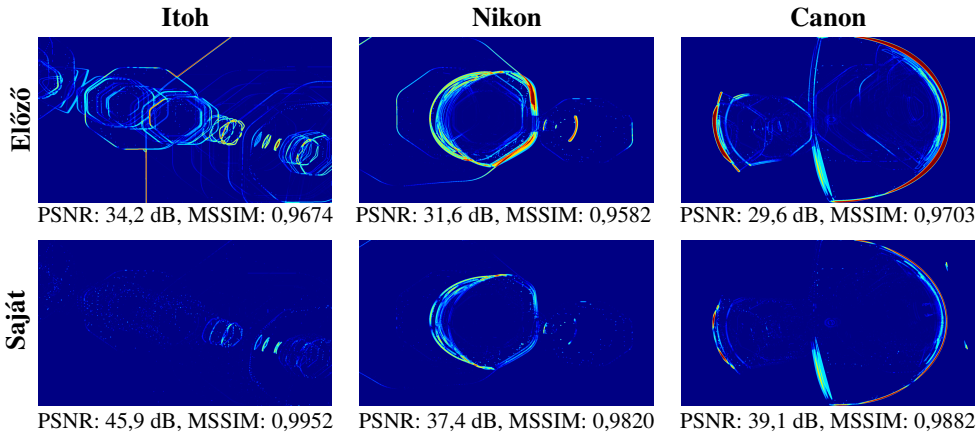
3.3.7.3. Renderelési pontosság

Ezt követően kiértékeltük a megjelenítési minőségét a naiv és a saját polinomiális modellünknek. Összehasonlítottuk a saját szimulált kimeneteinket az analitikus referenciákkal, amihez maximális jel-zaj viszonyt (PSNR) és strukturális hasonlósági indexet (SSIM) használtunk metrikaként. A pixelenkénti SSIM térkép, a PSNR és az átlagos SSIM értékek a 3.16. és a 3.17. ábrán láthatók.



3.16. ábra. Strukturális hasonlósági index (SSIM) térképek, maximális jel-zaj viszony értékek (PSNR) és átlagos SSIM értékek (MSSIM) a kis komplexitású optikai rendszerek esetén a naiv és a saját részleges polinomillesztési megközelítés alkalmazásával.

Amint azt az eredmények mutatják, módszerünk magas pontossággal közelíti az analitikus referenciát. A pontatlanságok nagy része a szellemek körvonalainál figyelhető meg. Ezek a ρ_s relatív sugárparaméterrel való pixelenkénti vágásból származnak, ami felerősíti a polinomiális közelítés pontatlanságait. A 3.14. és a 3.15. ábrán kiemeltünk néhány olyan területet, ahol a legnagyobbak az észrevehető eltérések. Ezek egyértelműen megmutatják, hogy a hibák marginálisak, a szimuláció ezek ellenére valószerű kimeneteket szolgáltat.

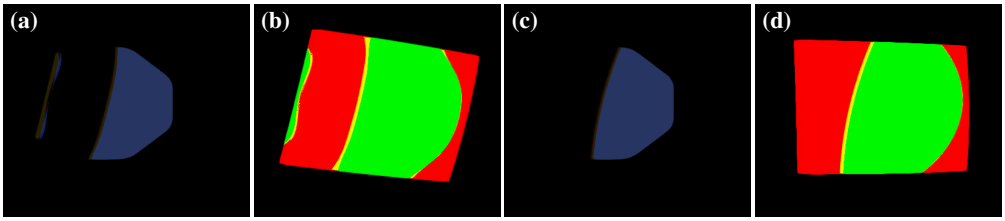


3.17. ábra. Strukturális hasonlósági index (SSIM) térképek, maximális jel-zaj viszony értékek (PSNR) és átlagos SSIM értékek (MSSIM) a magas komplexitású optikai rendszerek esetén a naiv és a saját részleges polinomillesztési megközelítés alkalmazásával.

A naiv megközelítéshez képest a részleges modellünk jelentősen jobb szimulációs pontosságot ér el. Ahogyan azt az SSIM térképek mutatják a 3.16. és a 3.17. ábrán, az eredeti modell hibája szintén a szellem élei körül jelenik meg a leginkább. A pontatlanul közelített vágási paraméterek azonban eltolódott spektrális csatornákban és gyakran erősen torzult szellemalakzatokban nyilvánulnak meg, amit tovább ront a sugár képtérbeli pozíciójának megnövekedett hibája. Ennek következményeként a szimuláció nagyobb mértékben tér el az analitikus referenciától, és sokkal alacsonyabb a fizikai helyessége. Módszerünk jelentősen csökkenti az ilyen pontatlanságokat.

Egy különös kihívást jelentő eset, amikor a relatív sugár szenzoros képének alakja irreguláris. Ennek fő oka az, hogy a közös polinomiális bázis nem képes a vágási információ gyorsan változó alakját helyesen reprezentálni. A 3.18. ábrán látható egy példa egy ilyen esetre a Nikon zoomobjektív esetén, ahol egyetlen hibás szellem van kiemelve megnövelt intenzitással. Ahogyan azt a példa is demonstrálja, a különbség nehezen észrevehető a megnövelt intenzitás ellenére is. Ennek oka, hogy ilyen esetek tipikusan akkor történnek, amikor a beérkező sugárrács nagy részét akadályozza az objektív borítása, ami a létrejövő szellemet nagy részben láthatatlanná teszi. Emiatt az ilyen pontatlanságokból fakadó szellemek eldobhatók a szimuláció bármilyen valószerűségcsökkenése nélkül.

Ilyen esetekben az analitikus sugárkövetéshez való visszatérés is egy alternatíva lehet a probléma elkerülése érdekében. A finomabb polinomiális bemeneti változók használata (mint például a bemeneti sugárrácsnak a bemeneti pupillára való vetülete) vagy a hibafüggvény hangolása az egyedi optikai rendszerre szintén segíthet.



3.18. ábra. A Nikon zoomobjektív 100-as szellemének 10-szeres szorzóval való nagyítása a jobb láthatóság érdekében. (a, b) A saját részleges modellünkkel generált kimeneti szimuláció és a hozzá tartozó relatív sugár vizualizációja. (c, d) Az analitikus sugárkövetéssel létrehozott referenciaképek.

3.3.7.4. Renderelési teljesítmény

A modellünk illesztési és megjelenítési minőségének megállapítása és validációja után az analitikus és polinomiális megközelítések renderelési teljesítményét értékeltük ki. Erre a célra megmértük a sugárkövetéses és a teljes lencsefényfolt-szimulációs folyamatok hosszát a 3.14. és a 3.15. ábrán bemutatott kimenetek esetén. Az eredményeket a 3.10. táblázat foglalja össze.

	Analitikus		**Polinomiális (előző)**		**Polinomiális (saját)**	
	Sugárátvitel	**Teljes**	**Sugárátvitel**	**Teljes**	**Sugárátvitel**	**Teljes**
Heliar Tronnier	0,75 ms	1,95 ms	0,52 ms	2,13 ms	0,32 ms	1,82 ms
Angenieux	3,32 ms	4,54 ms	1,37 ms	2,66 ms	0,89 ms	2,23 ms
Itoh	3,72 ms	4,48 ms	1,27 ms	2,21 ms	0,80 ms	1,76 ms
Nikon	10,54 ms	11,79 ms	4,35 ms	5,63 ms	2,27 ms	3,52 ms
Canon	33,65 ms	35,53 ms	8,54 ms	10,65 ms	4,51 ms	6,59 ms

3.10. táblázat. A sugárátviteli lépés és a teljes lencsefényfolt-szimuláció futási ideje a 3.14. és a 3.15. ábrán bemutatott összes kimenet esetén.

Ahogyan azt a mérések mutatják, módszerünk jelentősen jobban teljesít

az összes többi tesztelt megoldásnál. Még a legegyszerűbb (Heliar Tronnier) esetben is a részleges polinomiális megközelítésünk közel kétszer gyorsabb az analitikus sugárkövetésnél, és négy-ötször gyorsabb a nyers erő módszernél a legkomplexebb esetben (Nikon, Canon). Algoritmusunk valósidejű fényfoltszimulációt valósít meg tetszőleges optikai rendszer esetén, ami korábban lehetetlen volt az analitikus sugárkövetéssel, vagy az előző polinomiális modellekkel csupán a kimenet minőségének feláldozása árán lett volna megvalósítható.

Fontos megjegyezni, hogy a valós kamerák tartalmazhatnak olyan alkotóelemeket, amik nem írhatók le síkbeli és szférikus felületekkel. Az ezek kezeléséhez szükséges számítások általában jelentősen lassabbak, ezért a polinomiális approximáció valószínűleg még nagyobb hatással lenne a szimuláció sebességére olyan optikai rendszerek esetén, amik ilyen alkotóelemekkel rendelkeznek.

3.3.7.5. Memóriahasználat

A polinomiális sugárátviteli megközelítésünk kiértékelésének utolsó lépésében megmértük a teljes és a részleges modellek memóriaigényét az összes tesztelt optikai rendszer esetén. Az eredményeket a 3.11. táblázat foglalja össze.

	Heliar Tronnier	**Angenieux**	**Itoh**	**Nikon**	**Canon**
Előző	10,55 KB	34,18 KB	58,46 KB	208,41 KB	459,92 KB
Saját	1,00 MB	3,21 MB	4,82 MB	13,52 MB	30,99 MB

3.11. táblázat. A teljes polinomiális modell (Előző) és a saját részleges polinomiális modellünk (Saját) memóriaigénye.

Ahogyan az várható volt, a naiv polinomiális modellhez képest a mi részleges módszerünk növeli a memórialábnyomát a polinomiális rendszereknek. Azonban a közös polinomiális bázis biztosítja azt, hogy a létrejövő adatmennyiség továbbra is ésszerű kereteken belül marad. Modellünk tárhelyigénye mind az offline (merevlemezen), mind az online (CPU-n és GPU-n) esetben alkalmas a széles körben elérhető hardverekhez. Emiatt úgy gondoljuk, hogy a módszerünk tökéletesen alkalmas valósidejű alkalmazásokra.

4. Mikroszkopikus élőlény viselkedését vizualizáló keretrendszer

Tudományos kutatási projektek nagy részének számos fázisában témaspecifikus adattömegek jönnek létre ([55]), amiknek a feldolgozása és a bennük rejlő tudás kinyerése a kutatási folyamat előrehaladásában kritikus lehet. Az értekezés ezen részében a 2020-ban és 2021-ben megjelent publikációink [56, 57] alapján ismertetem saját, kutatásspecifikus igényeket kielégítő vizualizációs keretrendszerünk részleteit. A létrehozott eszköz megalkotásának célja a bemenő adatok kezelése, azoknak három dimenzióban történő megjelenítése és a hatékony elemezhetőség volt. Létrehoztunk továbbá egy saját számítási modellt a baricentrikus koordináták felhasználásával történő objektumlokalizáció céljára, amelyet a keretrendszerünkben is alkalmaztunk. Új módszerünk részleteit és az így elért eredményeket szintén ebben a fejezetben ismertetem.

4.1. Kutatási háttér

A nagy mennyiségű adatból való információkinyerés gyakran egy jelentős kihívást okozó feladatnak bizonyul. Mivel az emberi szem a szövegektől eltérően képes a képeken tárolt leíró jellemzőket párhuzamosan feldolgozni [58], a rendelkezésre álló adatoknak egy vizuális formában történő megjelenítése az esetek túlnyomó többségében jelentősen hatékonyabbá teheti az elemzést. Az adattartalom átláthatóvá válik, átfogó jellemzőiről az első benyomás nagyon hamar kialakul, és az egyes várt vagy rejtett információk, mint például anomáliák vagy mintázatok, nagyon hamar felismerhetővé válnak. Gyakran fordul elő olyan helyzet is, amikor a vizualizáció nem csak a magasabb szintű megfigyelés céljából hasznos, hanem már az adatok alapvető értelmezésének is szerves részét képezi.

A kutatási projektek során létrejövő adatok a tudományos jellegükből fakadóan gyakran specifikus, nemtriviális karakterrel rendelkeznek. Emiatt az ilyen típusú adatok feldolgozására, kezelésére és megjelenítésére általában személy-

reszabott keretrendszerre van szükség. Ez a probléma merült fel a Debreceni Egyetem Pszichológiai Intézetének egyik kutatása kapcsán is. A projekten dolgozó kutatók a mikroszkopikus élőlények evolúciójában részt vevő szerkezeti tényezőket vizsgálták [59, 60]. Ezen munka során létrehoztak egy *futópadmodellnek* nevezett szimulációs algoritmust, amelynek elnevezése abból fakad, hogy egy futópad működéséhez hasonlóan a mikroorganizmus tulajdonképpen nem mozdul el, hanem a mozgó szalag analógiájaként az élőlény környezetének elemei váltanak pozíciót ellentétes irányban. A szakirodalomban hasonló reprezentációs megközelítést alkalmaznak lokalizációs problémák esetén [61–63]. Ezekben az esetekben is egy mozgó objektum szenzorai (pl. kamera, lézer) által észlelt különböző típusú információk kerülnek rögzítésre, amik a környezet észlelt változásait reprezentálják. A futópadmodell esetében a mikroorganizmus fiziológiai szenzorai objektumkoordinátákat rögzítettek.

A szimulációs modell felhasználásával adathalmazoknak egy hosszú sorát generálták a kutatók, amely adathalmazok mikroorganizmusok viselkedésének leírását tartalmazták bizonyos strukturális tulajdonságoktól (pl. érzékelőiknek darabszáma és elrendezése) függően. Vizuális reprezentáció nélkül azonban a csupán számokat tartalmazó adatsorokból lehetetlen volt végrehajtani bármilyen, az élőlény viselkedésével kapcsolatos magasabb szintű megfigyelést.

4.1.1. Szimulációs modell leírása

A szimulációs projekt célja egy olyan sematikus, mikroszkopikus, szabadmozgású élőlény megfigyelése volt, amely a 3D térben, egy nedves környezetben mozgott. Ebben a mérettartományban és környezetben a viszkózus (súrlódási) erők erősebbek a tehetetlenségi erőnél, így a mozgás az úgynevezett alacsony Reynolds-számok szerint történt [64]. Ebben az esetben ez egy kúszó mozgást eredményezett a nedves környezetben. A környezet tartalmazott továbbá néhány, az élőlény számára releváns ingereket kiváltó elemet, amelyek pontszerű fényforrásokként táplálékokat reprezentáltak. A táplálékok mindegyike rendelkezett egy ismert intenzitással, ami azt jelezte, hogy ha az élőlény érzékelte őt, akkor azt milyen erősséggel tette.

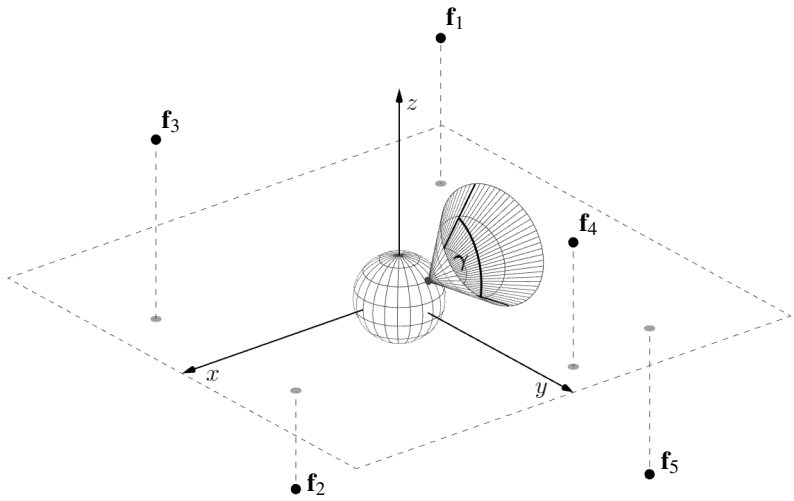
Strukturálisan a mikroorganizmus a saját testéből, valamint néhány ismert

látószögű érzékelőből épült fel. Az egyes szimulációs adatsorok forrásai az érzékelők számában, ezek látószögében, valamint az élőlény testén való elhelyezkedésében tértek el, mivel az eredeti vizsgálat célja ezen tulajdonságok szerepének megfigyelése volt.

Az élőlény táplálékkeresési célból mozgott a környezetében. A táplálékok közül bizonyos (előre megadott) feltételeknek megfelelőeket érzékelt, ezek közül a kiválasztott felé pedig elindult. A kiválasztott táplálékot elfogyasztotta, ha sikerült elérnie (tehát át is haladt azon a pozíción, amin a táplálék volt), egyébként új táplálékot választott célul. Az élőlény akkor érzékelt egy táplálékot, ha az legalább az egyik érzékelőjének a látóterébe esett, valamint ha a táplálék által kiváltott inger erőssége elég magas volt ahhoz, hogy releváns legyen. A táplálékok jelentősen kisebbek voltak, mint az élőlény, tehát ahhoz, hogy egy táplálékot elfogyasszon, az élőlénynek egy precízen tájolt mozgásra volt szüksége.

A 4.1. ábra a szimuláció fő elemeit mutatja be. Az origó középpontú gömb reprezentálja a mozgó élőlényt, amelynek ebben az esetben egy darab, γ látószöggel rendelkező érzékelője van. Az $\mathbf{f}_i\,(i = 1,2,\ldots,5)$ pontok jelölik a táplálékok pozícióit a környezetben.

Minden táplálék rendelkezett egy-egy rögzített értékű élettartammal, ami



4.1. ábra. A szimulációs környezet fő elemei.

az adott táplálléknak egy másik egyed általi elfogyasztását volt hivatott reprezentálni. Egy táplálék élettartama lejárhatott, mielőtt a megfigyelt organizmusunk elfogyaszthatta volna őt. Ez azt jelentette, hogy az élőlényünk nem tudta időben elérni, és azt egy másik egyed hamarabb elfogyasztotta. Ilyenkor egy másik célt választott magának a mikroorganizmus, és elindult felé. A szimuláció egyik vizsgálandó pontja volt az a kérdés, hogy ebben az esetben melyik táplálékot választja, és miért pont azt. A táplálékok egy idő után eltűnhettek a környezetből, aminek két oka lehetett: az élőlényünk elfogyasztotta vagy az élettartama lejárt (tehát egy másik egyed fogyasztotta el). Minden olyan pillanatban, amikor egy táplálék eltűnt, helyette egy új jelent meg.

A kiindulási kutatás során létrehozott modell reprezentálta a fent részletezett környezetet, szimulálta a mikroorganizmus viselkedését, valamint adatokat szolgáltatott annak mozgásáról.

4.1.2. Adatstruktúra

Az adatokat szolgáltató kiindulási szimulációs modell egyedüli célja az élőlény viselkedésének és a táplálékokhoz való viszonyának a meghatározása volt. Mivel a rendszer egyedül a szimulációra koncentrál, így semmilyen vizuális visszajelzést nem szolgáltat. A kimenet minden esetben egy JSON formátumú adathalmaz, ami lépésről lépésre írja le a mozgást.

Ahogyan azt a 4.1.1. kódrészlet mutatja, a létrejövő fájlnak része az élőlény strukturális leírása. Ez tartalmazza a mikroorganizmus fix méretét, az érzékelőit és egy látószöget. A leírás szintén tartalmazza az érzékelők gömbi koordinátáit. Ezek az érzékelők elhelyezkedését jelzik az élőlény testén és az élőlény saját lokális koordináta-rendszerében vannak megadva. Ebben a példában az élőlény négy érzékelővel rendelkezik, amik egyenlően vannak elosztva a testén, valamint mindegyikről tudjuk, hogy 85°-os látószöggel érzékel.

Az adathalmaz további része a lépésenkénti mozgást írja le. A lépéseket tartalmazó képkockasorozat definiálja az élőlény mozgását, amihez minden képkocka azonos struktúrával rendelkezik. Amint azt a 4.1.2. kódrészlet illusztrálja, egy képkocka számos attribútumot tartalmaz, mint például a képkocka sorszáma, a forgatási vektor, az eltolási vektor és a táplálékok gyűjteménye.

```
"structure" : {
  "size" :0.05,
  "sensors" : [
    { "theta" :0.523599, "phi" :0.0 },
    { "theta" :0.523599, "phi" :1.5708 },
    { "theta" :0.523599, "phi" :3.14159 },
    { "theta" :0.523599, "phi" :4.71239 }
  ],
  "angleofview" :85
}
```

4.1.1. kódrészlet. Részlet a szimulációs keretrendszer egy kimenetéből, ami a mikroorganizmus struktúráját írja le.

Egy táplálék adatai annak az élőlényhez viszonyított relatív koordinátáit, egy intenzitásértéket és egy újragenerálási indexet tartalmaznak. Az újragenerálási index jelzi a táplálék potenciális újrapozícionálását az adott képkockában.

```
"frames" :[
  { "frame_num" :0,
    "rv" : [0, 0, 0],
    "tv" : [0, 0, 0],
    "foods" :[
      { "x" : -1.08496, "y" :0.838925,
        "z" : 1.13508, "i" :0.587424,
        "reset" :1,
      }, ...
    ]
  }, ...
]
```

4.1.2. kódrészlet. Részlet a szimulációs keretrendszer egy kimenetéből, ami a szimulált környezet egy képkockában lévő állapotát írja le.

4.2. Baricentrikus számítási modell

4.2.1. Probléma leírása, motiváció

Azt a kiindulási problémát, ami miatt triviális módon nem működhetett a vizualizáció, az adat létrehozásának koncepciója adja. Az organizmus viselkedését szimuláló algoritmus működési megközelítése miatt a létrejövő adathalmazok a környezet és az abban szereplő objektumok leírását speciális módon tartalmazzák. Amint azt a 4.1.1. alfejezetben leírtam, ez egy mozdulatsorozat, amelyet a mozgó objektum környezetének viselkedése közvetett módon ír le. Ez a feltételezés teszi szükségessé az objektum pozíciójának meghatározását a globális koordináta-rendszerben a saját környezetére alapozva.

Az adatsor az egyes lépésekben a referenciapontok változását tartalmazza az objektum lokális koordináta-rendszerében az előző lépéshez viszonyítva. Ugyanakkor a mozgó objektum megjelenítése a célunk úgy, hogy a referenciapontok mozdulatlanok maradjanak a globális rendszerben. Szintén tudjuk, hogy az objektum a mozgását a globális origóból indítja. Ha az objektum lokális koordináta-rendszerét tekintjük, akkor ő minden lépésben az origóban fog elhelyezkedni. Ezen két információ birtokában (referenciapontok mozdulatlansága és mozgás origóból indítása) felhasználhatjuk a környezetben lévő objektumok pozícióit a megfigyelt mozgó objektum meghatározásához.

Az adat egy mozgási lépéshez egy azonosítót és egy négyelemű tömböt tartalmaz. Az összes elem esetén tartalmazza annak az adott lépésben ismert relatív pozícióját. További információt nem tartalmaz az adathalmaz.

Már létező, mikroorganizmusok vizualizációjával kapcsolatos rendszerek elérhetők [55, 65], az adathalmazok nemtriviális jellege azonban további feldolgozást és számításokat tett szükségessé ahhoz, hogy a szimuláció eredménye vizuálisan megjeleníthető legyen. Emiatt viszont nem állt rendelkezésünkre olyan, már létező megoldás, amit gyakorlatba ültetve könnyedén vizualizálhattuk volna az adatokat. Így egy saját vizualizációs keretrendszert hoztunk létre, amely speciálisan erre a problémára szabva elégítette ki az adatok feldolgozásával kapcsolatos igényeket. A keretrendszernek kezelnie kellett a bemeneti adathalmaz indirektségét és meg kellett határoznia a környezeti elemek

képkockánkénti elhelyezkedését. Ehhez egy olyan eljárást hoztunk létre a lokális koordináta-rendszerből globálisba történő konverzióra, amely baricentrikus koordinátákra építkezik.

4.2.2. Szakmai előzmények

Számos területen merülnek fel olyan tudományos problémák, amelyek megoldásához gyakran alkalmazott eljárások a koordináta-rendszerek közötti konverziók, valamint koordináta-transzformációk. Ilyen terület például a számítógépes grafika és a képfeldolgozás [66], geometriai problémák [67], fizika [68] és térinformatika [69, 70]. Az általunk javasolt algoritmus olyan problémákra szolgáltat egy hatékony megoldást, ahol különböző koordináta-rendszerek közötti konverzió elvégzése is szükséges.

A megoldandó probléma bemutatásához feltételezzük, hogy rendelkezünk egy mozdulatsorozattal, amelyet annak egyes lépései írnak le. Célunk a mozgó objektum pozíciójának meghatározása minden lépésben. Egy objektum lokalizációja egy gyakori probléma különböző tudományos területeken, viszont számos esetben az ehhez elérhető és felhasználható információ erősen korlátozott, vagy valamilyen értelemben speciális [71–73]. Ahogyan egy objektum (például robot vagy járókelő) mozog, az érzékelői koordinátainformációkat gyűjthetnek a környező statikus objektumokról saját magához képest [74, 75].

Esetünkben a lokalizáláshoz elérhető adat különleges abban a tekintetben, hogy csupán néhány, a mozgó objektum környezetében elhelyezkedő referenciapont (és ezeknek a változó koordinátái) adott, amelyek az objektum lokális koordináta-rendszerében vannak definiálva. A feladat minden lépésben a mozgó elem globális koordinátáinak meghatározása. Ehhez az objektum lokális rendszerében adott koordinátákat szükséges a globális koordináta-rendszerbe konvertálni a rendelkezésünkre álló információk alapján. Ugyanakkor szükség van új számítási módszerünk pontosságának elemzésére is, mivel ez a pontosság kardinális lehet különböző helymeghatározási alkalmazásokban [76, 77].

Ebben az alfejezetben egy olyan számítási modellt mutatok be, amellyel a kívánt mozgási adat meghatározható a rendelkezésre álló adatok alapján. Módszerünk a baricentrikus koordinátáknak a környezetben lévő referenciapontok-

kal történő használatán alapszik.

A számítógépes grafika területén széles körben alkalmazottak a baricentrikus koordináták. Számos alkalmazásuk közül megemlítendő az interpoláció és deformáció [78–80], karakterartikuláció [81] és a poligonháló-paraméterezés [82, 83]. Gyakorlati hasznuk más területeken is bizonyított, mint például a szenzorhálózatok [72] vagy robotlokalizáció [71, 74].

4.2.3. Transzformációs mátrixokon alapuló alternatíva

Amikor a keresett koordináták meghatározása céljából próbáltuk létrehozni a megfelelő eljárást, kézenfekvő megoldásnak tűnt a hasonló problémákhoz használható mátrixtranszformációk alkalmazása. Ehhez az aktuális és az azt megelőző lépés referenciapontjait tekintettük, majd meghatároztuk azt a mátrixot, amely az előző lépésbeli pozícióhalmazt az aktuálisba transzformálja. Minden aktuális képkockában az azt megelőző lépésekben hasonló módon meghatározott mátrixok sorozatába fűztük az újonnan kapott mátrixot (mátrixszorzás alkalmazásával), majd ezt felhasználva megkaptuk az objektum keresett pozícióját.

Legyenek $x_{i,k}$, $y_{i,k}$ és $z_{i,k}$ $(i \in \{1,2,3,4\})$ az i-edik környezetbeli referenciapont x, y és z koordinátái a k-adik lépésben. Legyen továbbá G_i a referenciapontok lokális koordinátáit tartalmazó mátrix:

$$G_i \coloneqq \begin{pmatrix} x_{1,i} & x_{2,i} & x_{3,i} & x_{4,i} \\ y_{1,i} & y_{2,i} & y_{3,i} & y_{4,i} \\ z_{1,i} & z_{2,i} & z_{3,i} & z_{4,i} \end{pmatrix}. \tag{4.1}$$

Ezeket a jelöléseket használva egy olyan mátrixot keresünk, amire a következő teljesül:

$$T_{k+1} \cdot G_k = G_{k+1}. \tag{4.2}$$

T_{k+1} tehát az a mátrix, amelyik a k-adik lépés referenciapontjait a $k+1$-edik lépés referenciapontjaivá transzformálja. Ennek megfelelően a transzformációs mátrix meghatározható a következő módon:

$$T_{k+1} = G_{k+1} \cdot G_k^{-1}. \tag{4.3}$$

A fenti felírásra és azon feltételezésünkre alapozva, hogy az objektum az első lépésben az origóban helyezkedik el, a megfigyelt objektum i-edik lépésbeli pozícióját meghatározhatjuk az összes addigi lépésben meghatározott transzformációs mátrix összesítésével. A keresett pozíció tehát az alábbi:

$$\left(\prod_{k=1}^{i} T_k^{-1}\right) \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{4.4}$$

A (4.4) egyenlet átrendezhető a következő alakra:

$$\left(\prod_{k=0}^{i-1} T_{i-k}\right)^{-1} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{4.5}$$

A mozgási lépések nagy száma miatt azonban ez a módszer a szimuláció előrehaladásával emelkedő tendenciájú számítási hibákat eredményez a tesztesetek többségében a lebegőpontos hiba felhalmozódása miatt.

4.2.4. Globális koordináták meghatározása baricentrikus koordinátákkal

Ahhoz, hogy a baricentrikus koordinátákat használjuk fel a felvetett probléma megoldására, fontos, hogy képesek legyünk meghatározni a mozgó objektum baricentrikus koordinátáit a négy fix referenciapontra (amik a mozgó elemet körbeveszik) támaszkodva az ő lokális koordináta-rendszerében.

Ahogyan azt a 4.2.3. alfejezetben ismertettem, $x_{i,k}$, $y_{i,k}$ és $z_{i,k}$ $(i \in \{1,2,3,4\})$ az i-edik környezetbeli referenciapont x, y és z koordinátái a k-adik lépésben. Legyenek továbbá $w_{i,k}$ $(i \in \{1,2,3,4\})$ a mozgó objektum baricentrikus koordinátái a k-adik lépésben a fenti referenciarendszerre vonatkozóan. Szintén tudjuk, hogy a számítás ezen fázisában minden a mozgó objektum lokális koordináta-rendszerében van megadva, így maga az objektum mindig az origóban fog elhelyezkedni, $(0,\ 0,\ 0)$ koordinátákkal.

Baricentrikus számításunkhoz a 2.5. alfejezetben ismertetett 2.12. egyenlet-

tel definiált kétdimenziós esetet terjesztjük ki három dimenzióra. A háromdimenziós esetben a keresett baricentrikus koordinátákat egy tetraéder csúcsaira vonatkozólag határozzuk meg. Tegyük fel, hogy $\sum_{i=1}^{4} w_{i,k} = 1$, így $w_{4,k}$ kifejezhető a többi súly felhasználásával a következő módon:

$$w_{4,k} = 1 - \sum_{i=1}^{3} w_{i,k}. \tag{4.6}$$

Annak a ténynek, hogy a megfigyelt objektum koordinátái $(0,\ 0,\ 0)$ minden lépésben, valamint a fenti számításnak a felhasználásával a következő egyenletrendszert kapjuk a k-adik lépésben:

$$\sum_{i=1}^{3} w_{i,k} \cdot x_{i,k} + \left(1 - \sum_{i=1}^{3} w_{i,k}\right) \cdot x_{4,k} = 0 \tag{4.7}$$

$$\sum_{i=1}^{3} w_{i,k} \cdot y_{i,k} + \left(1 - \sum_{i=1}^{3} w_{i,k}\right) \cdot y_{4,k} = 0 \tag{4.8}$$

$$\sum_{i=1}^{3} w_{i,k} \cdot z_{i,k} + \left(1 - \sum_{i=1}^{3} w_{i,k}\right) \cdot z_{4,k} = 0. \tag{4.9}$$

A fenti egyenletrendszer triviálisan átalakítható a következő alakra:

$$\sum_{i=1}^{3} w_{i,k} \cdot \left(x_{i,k} - x_{4,k}\right) = -x_{4,k} \tag{4.10}$$

$$\sum_{i=1}^{3} w_{i,k} \cdot \left(y_{i,k} - y_{4,k}\right) = -y_{4,k} \tag{4.11}$$

$$\sum_{i=1}^{3} w_{i,k} \cdot \left(z_{i,k} - z_{4,k}\right) = -z_{4,k}, \tag{4.12}$$

ami átírható mátrixos alakba a következő módon:

$$A_k \cdot \mathbf{w}_k = \mathbf{b}_k, \tag{4.13}$$

ahol

$$A_k \coloneqq \begin{pmatrix} x_{1,k} - x_{4,k} & x_{2,k} - x_{4,k} & x_{3,k} - x_{4,k} \\ y_{1,k} - y_{4,k} & y_{2,k} - y_{4,k} & y_{3,k} - y_{4,k} \\ z_{1,k} - z_{4,k} & z_{2,k} - z_{4,k} & z_{3,k} - z_{4,k} \end{pmatrix}, \tag{4.14}$$

$$\mathbf{w}_k \coloneqq \begin{pmatrix} w_{1,k} \\ w_{2,k} \\ w_{3,k} \end{pmatrix}, \quad \mathbf{b}_k \coloneqq - \begin{pmatrix} x_{4,k} \\ y_{4,k} \\ z_{4,k} \end{pmatrix}. \tag{4.15}$$

A fenti mátrixos alakból kiszámíthatjuk a w_i súlyokat a következőképpen:

$$\mathbf{w}_k = A_k^{-1} \cdot \mathbf{b}_k. \tag{4.16}$$

A (4.6) egyenletből $w_{4,k}$ egyszerűen kiszámítható vektorműveletekkel:

$$w_{4,k} = 1 - \langle \mathbf{w}_k, \mathbf{1} \rangle, \tag{4.17}$$

ahol $\langle , \rangle$ a skaláris szorzatot, $\mathbf{1}$ pedig az $(1,\ 1,\ 1)^T$ vektort jelöli.

Mindezidáig a $(0,\ 0,\ 0)^T$ pont lokális koordináta-rendszerbeli baricentrikus koordinátáit határoztuk meg az aktuális lépésben. Következő fázisként az újonnan meghatározott baricentrikus koordinátákat használjuk fel arra, hogy kiszámítsuk a mozgó objektum pozícióját a globális koordináta-rendszerben. Ehhez egy statikus referenciabázist szükséges megállapítanunk a globális rendszerben, amire a négy referenciapont pozícióvektorát választjuk a legelső lépésből. Rájuk úgy tekintünk, mint egy tetraéder csúcsaira, így ők a kiszámított baricentrikus koordinátákkal felhasználhatók a globális koordináta-rendszerben keresett pozíció meghatározására.

Feltételezzük, hogy a mozgó objektum útját a globális origóból indítja. Ezáltal a mozgás elején a négy referenciapont pozíciója megegyezik azoknak a lokális koordináta-rendszerben definiált koordinátáival. Ezt az ismeretet felhasználva az egyes lépésekben kiszámított baricentrikus koordináták felhasználhatók a kiindulási referenciapontok súlyaiként. Az aktuális lépésben a pozíciókat az első képkocka adatai alapján létrehozott fix referenciához képest határozzuk meg, majd ezt súlyozzuk az aktuális képkockában meghatározott baricentrikus koordinátákkal.

A k-adik lépés baricentrikus koordinátáit a $\hat{\mathbf{w}}_k$ vektorba rendezzük:

$$\hat{\mathbf{w}}_k \coloneqq \left(w_{1,k}, w_{2,k}, w_{3,k}, w_{4,k}\right)^T. \tag{4.18}$$

A 4.1. egyenlet alapján a G_i mátrix a referenciapontok lokális koordinátáit tartalmazza:

$$G_i \coloneqq \begin{pmatrix} x_{1,i} & x_{2,i} & x_{3,i} & x_{4,i} \\ y_{1,i} & y_{2,i} & y_{3,i} & y_{4,i} \\ z_{1,i} & z_{2,i} & z_{3,i} & z_{4,i} \end{pmatrix}.$$

Ezt a jelölést felhasználva a mozgás indulását képező első képkockában, a referenciapontok globális koordinátái összegyűjthetők a G_1 mátrixba, amik ugyanazok lesznek, mint a lokális koordináták a mozgás első lépésében. Az aktuális lépéshez tartozó globális koordináta meghatározására így rendelkezésünkre áll minden szükséges adat. Jelöljük a keresett globális koordinátákat x_k, y_k és z_k-val, amik a következő módon határozhatók meg:

$$\begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix} = G_1 \cdot \hat{\mathbf{w}}_k. \tag{4.19}$$

4.2.5. Eredmények

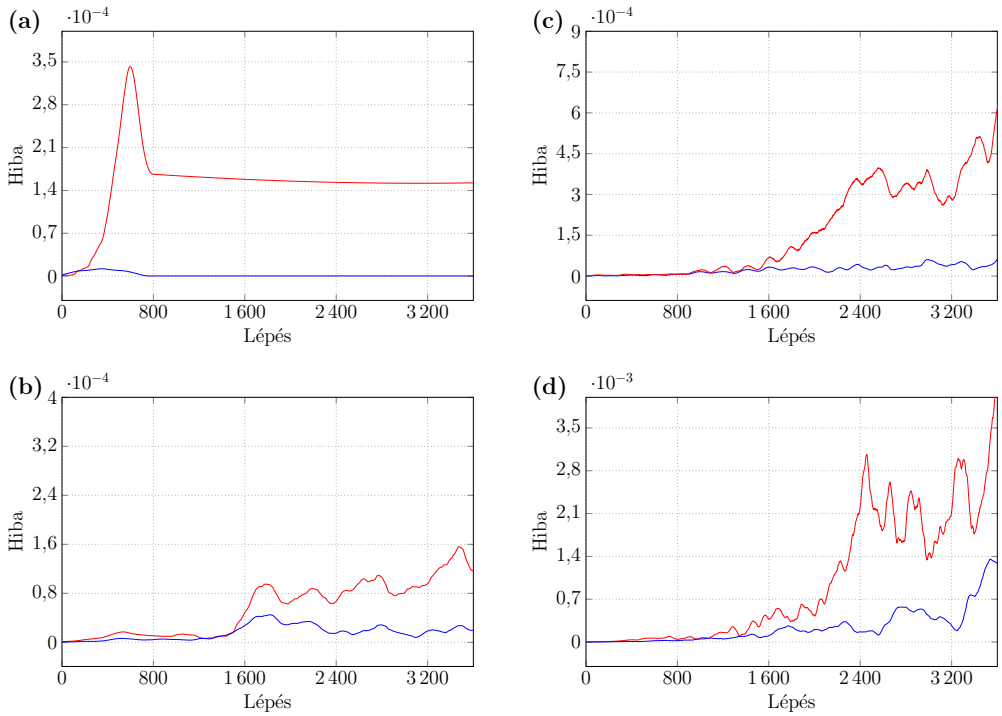
Mivel a probléma az volt, hogy a kapott koordináták egy adott koordinátarendszerben voltak, amiből a számunkra megfelelőbe volt szükséges konvertálni, így nem álltak rendelkezésünkre az összehasonlításhoz szükséges referenciaadatok. A módszer ellenőrzését csak szintetikusan generált adatokkal lehetett elvégezni. Ezen célra létrehoztunk néhány tesztadathalmazt, amik az objektum különböző típusú mozgásait tartalmazzák. Ezek a szintetikus lépéssorozatok a mozgás irányában térnek el egymástól, a különböző típusú mozdulatoknak az approximációs hibákra gyakorolt hatásának vizsgálata miatt.

Az alábbiakban négy különböző tesztesetet mutatok be, amik eltérő eltolási és forgatási konfigurációkat tartalmaznak. Az első esetben az objektum egy egyszerű mozgást végez néhány irányváltással, elfordulás nem része ennek a mozgásnak. A második tesztesetben az objektum egy x tengely menti spirálútvonalat követ a mozgása során. A harmadik esetben az első eset közel konstans mozgása van kiegészítve egy lépésenkénti véletlenszerű elfordulással az x és y

tengely körül. A negyedik tesztesetben az előző eset egészül ki egy véletlenszerű eltolással minden lépésben.

4.2.5.1. Pontosság

Az összes generált esetben összegyűjtöttük a mátrixos és a baricentrikus megközelítés abszolút hibáját, amihez az eljárások által szolgáltatott pozíciók és a referencia tesztadathalmazokban lévő pozíciók között lévő euklideszi távolságot határoztuk meg. Az így létrejövő értékeket használtuk az egyes tesztesetek teljesítményindikátoraiként. Mérési eredményeinket a 4.2. ábra foglalja össze.



4.2. ábra. Pontosságmérések. A kék vonal jelöli a baricentrikus megközelítés, a piros pedig a mátrixos megközelítés hibáját. Az egyes diagramokon szereplő mérések a mozgás típusában térnek el: (a) konstans mozgás néhány irányváltással, (b) spirális mozgás, (c) konstans irány véletlenszerűen generált elfordulásokkal, (d) véletlenszerű elmozdulásvektorok véletlenszerű elforgatásvektorokkal. Az egyes tesztesetekhez tartozó mozgás jellemzőinek pontos leírása a 4.2.5. alfejezetben olvasható.

Amint azt az eredményeink is mutatják, a mátrixos megközelítés által generált számítási hiba nagyságrendekkel nagyobb, mint amit a vizualizációs keretrendszerben is felhasznált, általunk javasolt algoritmus eredményezett. A baricentrikus koordinátákat alkalmazó eljárásunk nagyjából 10^{-6} nagyságrendű hibákhoz vezet, ezért nehéz ugyanazon a skálán vizualizálni a mátrixos megközelítés hibájával. Ebből kifolyólag a baricentrikus megközelítés hibája nullának tűnhet néhány helyen, viszont a nagyságrendbeli eltérés egyértelműen észrevehető.

A periodikus mozgást tartalmazó tesztesetben, ahol az objektum mozgása csak néhány irányváltást tartalmaz bármilyen forgás nélkül, a mátrixos módszer hibája nagyjából konstans, 10^{-4} nagyságrendű (4.2. ábra (a) diagram). Amikor az objektum egy spirál útvonalon, valamint egy konstans útvonalon véletlenszerű elfordulásokkal mozog (4.2. ábra (b) diagram, 4.2. ábra (c) diagram), ugyanez a hiba a szimuláció elején 10^{-5} mértékű, azonban a szimuláció előrehaladtával növekvő tendenciát mutat. Így rövid animációk esetén elfogadható eredményeket produkál a mátrixos eljárás, viszont hosszabb adatsorok szimulációjára nem megfelelő. Az utolsó eset viszont teljesen más képet mutat. Ebben a példában az objektum véletlenszerű irányokba mozog lépésenkénti véletlenszerű elfordulásokkal (4.2. ábra (d) diagram). Ez a bonyolult eset nagyobb skálán mozgó, a baricentrikus módszer esetében enyhén, a mátrixos módszer kapcsán pedig drasztikusabban növekvő tendenciájú hibát eredményez. A két eljárás közötti eltérés folyamatosan növekszik.

4.2.5.2. Teljesítmény

Mindkét algoritmus esetén lemértük a futási időket különböző hosszúságú adatsorok esetén. Három tesztesetet használtunk a futási idő mérésére, egy 10 000, egy 100 000 és egy 200 000 lépést tartalmazó lépéssorozatot. Az összes esetben megmértük a fő objektum pozíciójának meghatározási idejét. A teljesítménymméréseket egy kétmagos Intel Core i5-3230M 2,60 GHz-es processzoron végeztük. A mérési eredményeket a 4.1. táblázatban foglaltuk össze.

Az eredmények egyértelmű különbséget mutatnak a baricentrikus megközelítés és a mátrixos módszer teljesítménye között. Az egyes tesztesetek egy

növekvő tendenciájú eltérést produkálnak a két algoritmus futási ideje esetén. A legrövidebb eset (10 000 lépés) 6 tizedmásodperc eltérést eredményez, míg a leghosszabb eset (200 000 lépés) 240 tizedmásodpercet. Esetünkben ez egy 200%-os futásiidő-növekedést jelent.

4.3. Vizualizációs keretrendszer

A kutatási együttműködés végső célja egy olyan háromdimenziós megjelenítő keretrendszer kidolgozása volt, ami a létrejövő kutatásspecifikus adattömeg vizuális formában történő reprezentációjával segíti elő a kutatás előrehaladását. A vizualizációs eszköz létrehozásához elengedhetetlen volt az adatok feldolgozása és a megjelenítéshez szükséges információk meghatározása. A kiindulási kutatás során eredményül kapott speciális struktúrájú adatokat a 4.2. alfejezetben bemutatott metodológia felhasználásával dolgoztuk fel, majd egy háromdimenziós animált jelenetben ábrázoltuk. Értekezésem ezen részében a létrejött vizualizációs keretrendszert mutatom be.

4.3.1. Az élőlény pozíciójának meghatározása

Mivel az adatsor nem tartalmazta az aktuális képkockánkénti pozícióját az élőlénynek, így ezeket a mozgás indirekt, az élőlényhez viszonyított relatív leírásából kellett meghatároznunk. Erre a célra a 4.2.4. alfejezetben bemutatott baricentrikus eljárást implementáltuk, aminek a segítségével lokális rendszerből globális rendszerbe tudtunk konvertálni.

Amint az a 4.2.5.1. alfejezetben bemutattam, a felmerülő probléma egy mátrixokat alkalmazó megközelítéssel történő megoldása lényegesen magasabb lebegőpontos hiba felhalmozódásához vezetett, ami a korábbi képkoc-

Lépésszám	Baricentrikus megközelítés	Mátrixos megközelítés
10 000	14,5 ms	20,7 ms
100 000	54,9 ms	85,9 ms
200 000	101,4 ms	343,5 ms

4.1. táblázat. A baricentrikus és a mátrixos megközelítés futási idői.

kákra történő építkezésnek egy direkt következménye. A baricentrikus eljárás erőssége az ilyen pontosságproblémák elkerülésében és egy objektum globális rendszerben történő pozícionálásában rejlik, bármilyen korábbi képkocka információinak felhasználása nélkül.

A szimulált környezet tulajdonságainak előzetes ismeretei alapján a táplálékok pozícióit használjuk, mint referenciapontok. Az így létrehozott referenciaképkockával a birtokunkban, a javasolt eljárás két fő lépésből épül fel. Az első lépés célja a mikroorganizmus baricentrikus koordinátáinak meghatározása az ő saját lokális koordináta-rendszerében. A második lépés az előzőleg meghatározott baricentrikusok alkalmazásából áll, amivel a mozgó élőlény tényleges globális pozícióját kapjuk meg az adott lépésben egy statikus bázisra vonatkozóan. A mikroorganizmus érzékelőit szintén a baricentrikus eljárásunk felhasználásával pozícionáltuk.

4.3.2. Új és eltűnő táplálékok kezelése

Az általunk megalkotott, a 4.2. alfejezetben ismertetett, baricentrikus koordinátákat alkalmazó algoritmussal képesek vagyunk a mozgó organizmus globális pozíciójának meghatározására statikus környezetben. Ugyanakkor, ahogyan azt a 4.1.1. alfejezetben ismertettem, a referenciatáplálékok tetszőleges módon eltűnhetnek és megjelenhetnek új helyeken. Emiatt az algoritmusunk gyakorlatba ültetése során a referenciabázis táplálékkoordinátái nem mindig állandóak. Ez azonban szükségessé teszi a baricentrikus módszer által alkalmazott bázis frissítését minden ilyen táplálékpozíció módosulását követően.

Ha egy táplálék eltűnik, minden esetben megjelenik helyette egy új egy másik pozícióban. Az új táplálék koordinátái a többi táplálékhoz hasonlóan szintén az élőlény lokális koordináta-rendszerében vannak megadva. Ilyen helyzetekben egy új referenciabázist szükséges létrehoznunk az új elemekkel. Ehhez csak azon táplálékok pozícióit használhatjuk fel, amelyeknek ismerjük vagy meg tudjuk határozni a globális koordinátáit. Ám mivel a felhasználható környezeti referenciapontok között lehet újonnan megjelent táplálék is, így azok esetén már nem érvényes az a feltételezés, amit az első képkocka esetén felhasználtunk. Tehát az új táplálékok koordinátái ugyanúgy relatív módon lesz-

nek definiálva, mint az összes többi korábbi (nem első képkockában megadott) táplálék esetén, így az ő pozíciójukat a globális koordináta-rendszerben hasonló módon kell meghatároznunk, mint a mozgó objektumnak.

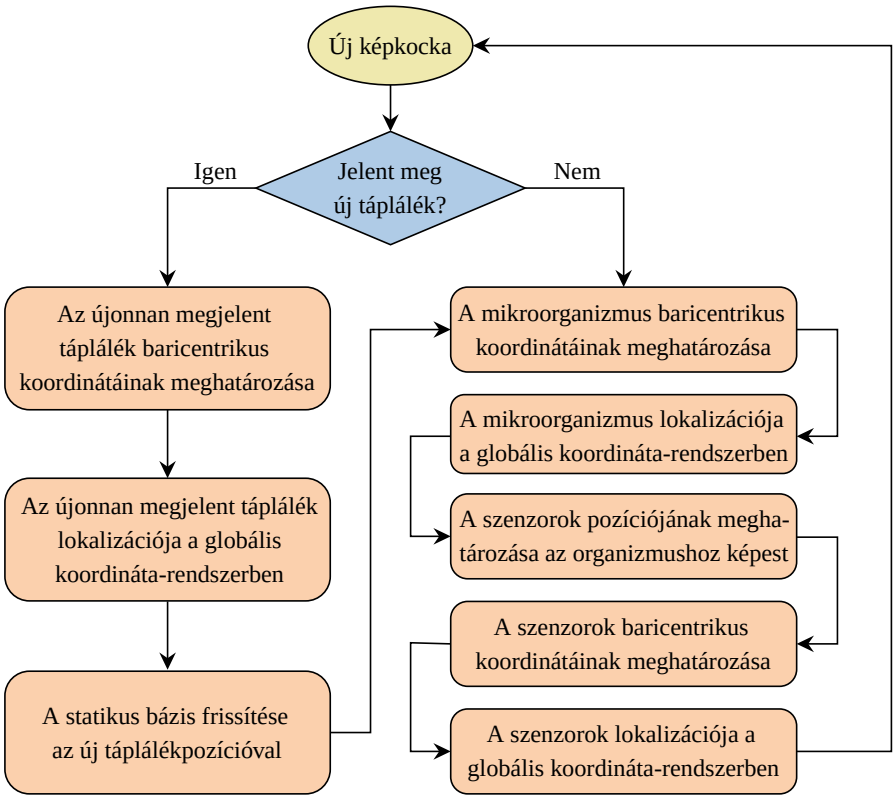
A pozícionálási probléma mindig az új táplálékok megjelenésének vizsgálatával kezdődik. Az újonnan megjelent táplálék pozíciójának konverziója egészen biztosan lehetséges, mivel minden képkocka garantáltan tartalmaz öt referenciapontot, tehát ha egy eltűnik, a fennmaradó négy továbbra is felhasználható a baricentrikus számítás referenciapontjaiként függetlenül attól, hogy minek keressük a baricentrikus koordinátáit.

Az első lépésben meghatározzuk az új referenciaobjektumok baricentrikus koordinátáit a többi négyre támaszkodva. Ezeket aztán a fennmaradó referenciapontok súlyaiként alkalmazzuk a globális koordináták meghatározására ugyanolyan módon, mint tettük azt a mozgó élőlényünk esetén a 4.3.1. alfejezetben bemutatottak szerint. Az utolsó lépésben az újonnan meghatározott pozíciókat eltároljuk, mint a referenciabázis új elemei, amit aztán a későbbiekben a baricentrikus számításhoz alkalmazhatunk.

Az újonnan megjelenő táplálékok kezelésével kiegészített baricentrikus megközelítés lépéseit tartalmazza a 4.3. ábra.

4.3.3. Funkciók

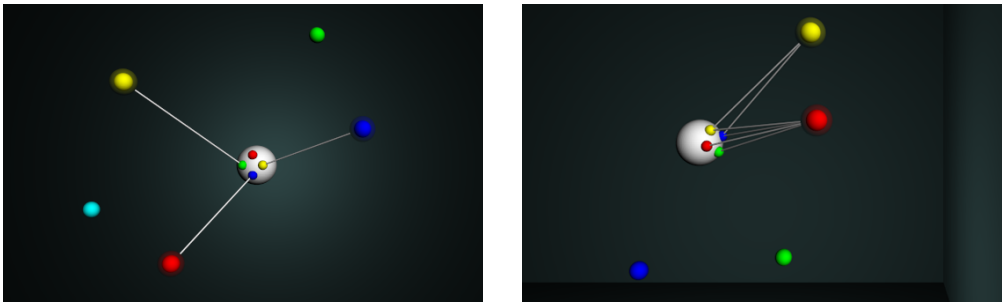
A keretrendszer fejlesztése során számos olyan funkciót hoztunk létre, ami a felhasználók számára nélkülözhetetlen az egyszerű és átlátható adatelemzéshez, valamint kérdéseik megválaszolásához. Az egyik ilyen funkció a képkockasorozatban történő lépkedés és tetszőleges időpillanatra ugrás előre- vagy hátramutató irányba. Elérhetővé tettük az animáció tetszőleges időpillanatban történő befagyasztását annak érdekében, hogy a felhasználó körül tudja járni a jelenetet, valamint az adott pillanatban megvizsgálhassa az élőlény viselkedését több nézőpontból is. A kamera ugyanakkor tetszőlegesen áthelyezhető bármelyik érzékelőbe, ami által a nézőpontot le tudjuk szűkíteni arra a régióra, amit az adott érzékelő észlel. A felhasználó képes továbbá kezelni az egyes vizualizációs beállításokat a grafikus felhasználói felületen keresztül, mint például az objektumméretek, vonalvastagságok, valamint objektumok láthatósá-



4.3. ábra. Az újonnan megjelenő táplálékok problémájával kiegészített lokalizációs metódus építőelemei.

ga, ami megkönnyíti az egyes szimulációs aspektusok vizsgálatát.

Vizuális analitikai szempontból az érzékelhető táplálékok megjelölése kiemelt fontossággal bír. Amint azt a 4.1.1. alfejezetben bemutattam, az élőlény akkor érzékel egy adott táplálékot, ha az valamelyik érzékelőjének látóterébe esik, valamint az intenzitása elég magas ahhoz, hogy releváns legyen. Azokat a táplálékokat, amiket legalább egy szenzor érzékel, egy áttetsző burokba helyeztünk, aminek mérete állítható. Az érzékelt táplálékokat továbbá összekötjük azokkal az érzékelőkkel, amik észlelik azt. Ezáltal egyszerűvé válik annak a vizsgálata, hogy az élőlény melyik táplálékot választja elfogyasztásra, valamint melyik irányba mozdul el ennek érdekében. Ezenkívül egy befoglaló dobozt is elhelyeztünk a teljes jelenet körül, amit a táplálékoknak és az élőlénynek a teljes animáció alatti legtávolabbi lehetséges pozíciói szerint határoztunk



4.4. ábra. Két példajelenet a vizualizációs rendszerből, eltérő táplálékészlelésekkel.

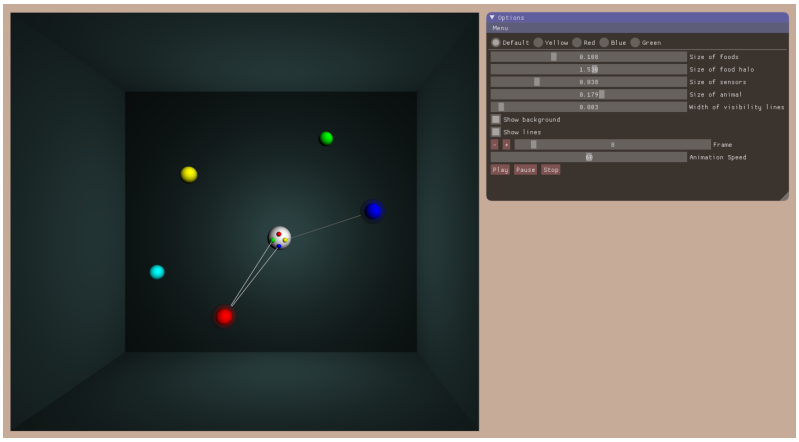
meg a betöltött adathalmaz alapján.

A 4.4. ábra két különböző pillanatképet mutat ugyanannak az adathalmaznak a vizualizációjából egy közelített nézőpontból. A burkok és az elemek közti kapcsolatok segítségével egyszerűen meghatározhatók a táplálékok érzékelhetőségei. A bal oldalon három eltérő érzékelő észleli a három érzékelt táplálékot. Ehhez képest a jobb oldalon az élőlény csupán két táplálékot észlel, viszont abból az egyiket az összes érzékelőjével, a másikat pedig kettővel.

4.3.4. Eredmények

A vizualizációs keretrendszert C++ nyelven készítettük el az OpenGL grafikus könyvtár felhasználásával. A 4.5. ábrán látható a rendszer egy pillanatképe, ami a fő elemeket tartalmazza. Amint ezen a példán is látszik, a szimulációs elemek gömbökként reprezentáltuk. Az egyes elemek színe az adott táplálékoknak és az élőlény testén lévő szenzoroknak egy hatékony és egyértelmű beazonosítását hivatottak elősegíteni.

Keretrendszerünk olyan információkat szolgáltatott, ami az adatoknak egy vizuális formába történő alakítása nélkül nem lett volna megfigyelhető. A vizsgálatot segítette például olyan pillanatokban, amikor egy táplálékot pont az élőlényünk elől fogyasztott el egy másik egyed. Ilyen esetben fontos információt szolgáltatott az a megfigyelés, hogy az élőlényünk milyen irányba tervezi újra útvonalát, illetve melyik másik táplálékot választja célpontul.

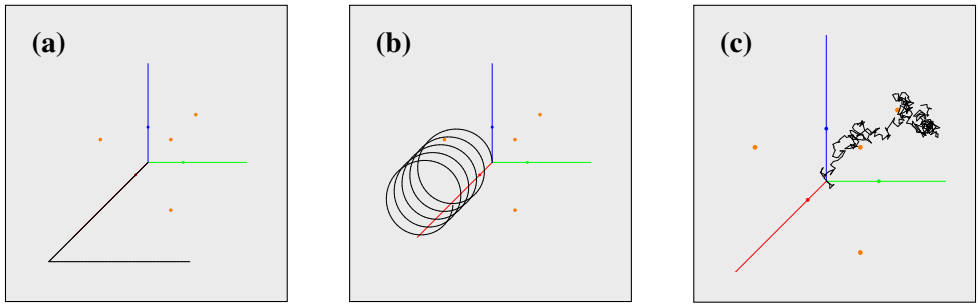


4.5. ábra. A vizualizációs keretrendszer egy pillanatképe. Ebben a jelenetben a középső fehér gömb reprezentálja a mozgó mikroorganizmust, ami a piros táplálékot két szenzorával, a kék táplálékot pedig eggyel érzékeli. Az áttetsző burkok az érzékelt táplálékok körül az észlelés tényét, az egyes szenzorokhoz kötődő kapcsolatok pedig az adott táplálékot érzékelő szenzorokat jelölik.

4.3.4.1. Pontosság

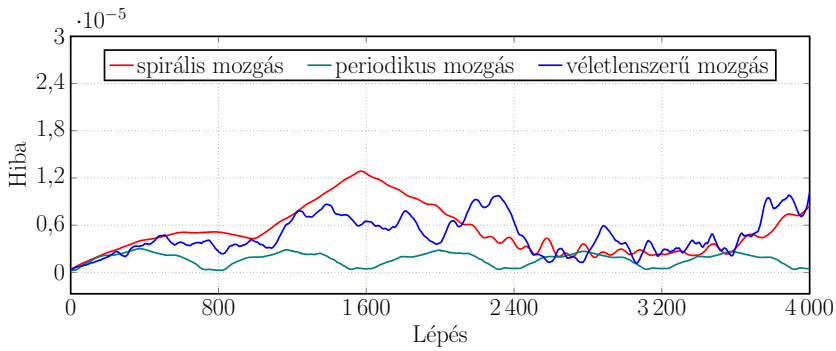
Vizualizációs rendszerünk pontosságának meghatározásához azt számos különböző teszteseten keresztül is megvizsgáltuk. Ahogyan azt a 4.2.5. alfejezetben kiemeltem, a szimuláció kimeneteként kapott adathalmazok nem tartalmazzák a valódi elhelyezkedési információkat, ami az összehasonlításhoz szükséges lenne, így első lépésként alkalmas összehasonlítási alapokat kellett létrehoznunk a pontosságmérésekhez. Emiatt számos különböző mesterséges mozgássorozatot generáltunk, amelyeknek a felhasználásával egy bázishoz jutottunk az általunk javasolt baricentrikus eljárás pontosságának méréséhez.

A 4.2.5. alfejezetben bemutatott generált útvonalakból ebben az esetben hármat használtunk fel a pontosságméréshez, amelyek a 4.6. ábrán láthatók. A fekete vonalak vizualizálják a generált mozgási útvonalat, a narancssárga pontok pedig a referencia táplálékpozíciókat. Mivel az adathalmazok létrehozása során nem vettük figyelembe az élőlény fiziológiai tulajdonságait, valamint az élőlény környezetének jellemzőit, ezért ezek a mozgássorozatok sokkal egyszerűbbek és szabályosabbak, mint amilyeneket az élőlény viselkedését szimuláló eljárással generált valós adathalmazok tartalmaznak.

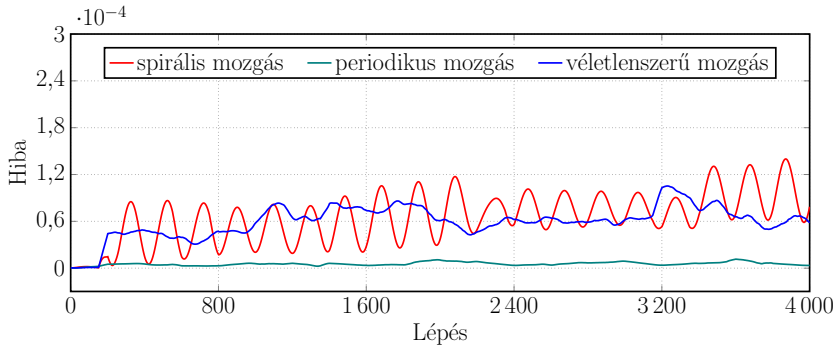


4.6. ábra. Mesterségesen generált mozgásútvonalak. (a) Egyenes vonalú mozgás néhány irányváltással. (b) Spirális mozgás. (c) Véletlenszerű mozgás és elfordulás.

A mérések eredményei a 4.7. és a 4.8. ábrákon láthatók. A piros vonaldiagram vizualizálja a spirális mozgás időben változó pozícióeltérését, a zöld mutatja a néhány irányváltást tartalmazó periodikus mozgás, a kék pedig a véletlenszerű mozgás eredményének pontosságát. A 4.7. ábra azokat a teszteseteket tartalmazza, amelyeknek nem része táplálékok eltűnése és újak megjelenése, míg a 4.8. ábra olyan példákat mutat, amik 200 képkockánként tartalmaznak egy új megjelent táplálékot. Az objektumpozíciók koordinátái a $[0, 15]$ intervallumból vehetnek fel értékeket. A 4.7. és a 4.8. ábrákon látható, hogy a eltérés az összes esetben nagyjából három-öt nagyságrenddel kisebb ettől.



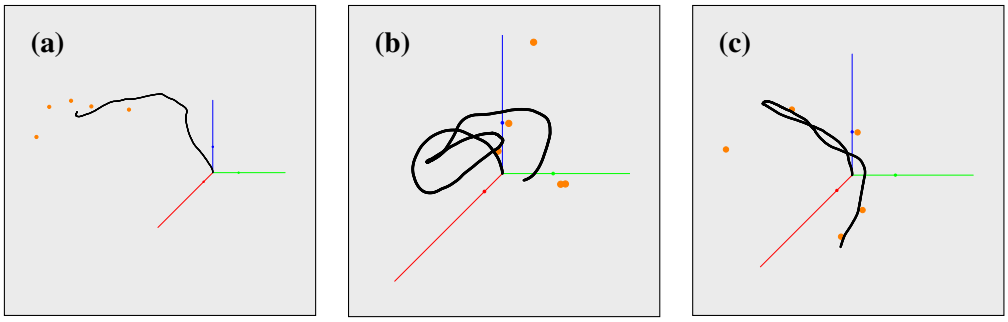
4.7. ábra. A 4.6. ábrán látható mesterségesen generált jelenetek vizualizációjának pontosságmérései újonnan megjelent táplálék nélkül.



4.8. ábra. A 4.6. ábrán látható jelenetek vizualizációjának pontosságmérései 200 képkockánként megjelenő új táplálék esetén.

A hibametrikák elemzése megmutatta, hogy a pozíciómeghatározás hibája és a táplálékok átlagos távolsága az élőlénytől minden képkockában korrelál. Ezt a legjobban a táplálék-újragenerálásokat tartalmazó spirális mozgás esetében létrejövő, oszcilláló hiba mutatja. Ebben az esetben a referenciapontok csoportokba gyűlnek egy bizonyos számú újragenerálás után, emiatt viszont az élőlény és a táplálék átlagos távolsága folyamatosan növekszik, majd csökken, ahogy az élőlény bejárja a spirális útvonalat.

Végezetül a keretrendszerünket néhány valós, az élőlény szimulációjából származó adathalmazzal is kipróbáltuk. A 4.9. ábrán látható néhány példa ilyen valós adathalmazokra.



4.9. ábra. Valós adathalmazokkal generált mozgásútvonalak. A narancssárga pontok az animáció utolsó képkockájában lévő táplálékokat reprezentálják.

5. Összefoglalás

Értekezésemben bemutattam a kameraobjektívek által létrehozott lencsefényfoltok fizikailag megalapozott hatékony szimulációját érintő területen (3. fejezet), valamint a saját, baricentrikus koordinátákon alapuló objektumlokalizációs megoldásunkat alkalmazó adatvizualizációs keretrendszerünk (4. fejezet) kapcsán elért új eredményeinket.

A lencsefényfoltokkal kapcsolatos munkánk során a jelenség szellemkomponenseire fókuszáltunk, amiknek a fizikailag hiteles és hatékony szimulációja volt a kutatásunk középpontjában. A valós idejű alkalmazásokban történő felhasználhatóság érdekében először egy csempézett megközelítést alkalmaztunk a fényfoltok raszterizációjára. Létrehoztunk egy olyan metódust, ami a ritka sugárrácsok optikai rendszeren történő végigkövetése után kapott sugárrácsot dolgozza fel. Első lépésként a szenzorra vetített sugárrácsból négyszögeket építünk, amiket a következő lépésben egy kétszintű hierarchikus megközelítéssel vizsgálunk meg. Az alacsony komplexitású területeket alkotó négyszögeket összevonjuk, így csökkentve a feldolgozandó adat mennyiségét. Ezt követően csoportosítjuk a négyszögeket a csempék általi tartalmazás szerint, majd a létrejövő puffert bejárva összesítjük az egyes szellemeknek az adott pixel színéhez történő hozzájárulását. Csempealapú raszterizációs algoritmusunkkal jelentősen növeljük a rendszer áteresztőképességét. A 3.2. alfejezet ismerteti új módszerünket, amit a 2023-ban megjelent cikkünkben [50] publikáltunk a tudományos közösség számára.

Létrehoztunk továbbá egy polinomiális optikán alapuló fényfoltszimulációs megközelítést is. Ehhez a korábbi polinomiális modell elemeit módosítottuk. Ezáltal az illesztéssel kapott polinomiális rendszer pontosabban közelíti a jelenséget, mint a korábbi megoldások, valamint hatékonyabban kiértékelhető GPU-n, mint az analitikus sugárkövetés. Képes továbbá több olyan jellemző kezelésére is, amelyeket a korábbi polinomiális megközelítések figyelmen kívül hagytak, nagyban megnövelve a szimuláció hitelességét. Az eljárás hatékonyságának további növelése érdekében megalkottunk egy

részleges zónákon alapuló illesztési algoritmust. A létrehozott modellel és illesztési metódussal a szükséges polinomiális tagok darabszáma csökkenthető, lényegesen megnövelve az illesztési folyamat és a szimuláció sebességét is. A polinomiális megközelítéssel a sugárkövetés költsége átemelhető egy előszámítási lépésbe, amit elegendő egyszer, a szimuláció előtt végrehajtani. A megjelenítés során már csak a létrejövő polinomiális modell kiértékelésére van szükség, ami GPU-n valós időben megvalósítható. A 3.3. alfejezet mutatja be az új megközelítésünket, amit a 2024-ben megjelent publikációnkban [54] foglaltunk össze. Módszereink referenciaimplementációját publikusan elérhetővé tettük[1].

Az objektumlokalizációs probléma megoldására megalkottunk egy baricentrikus koordinátákon alapuló eljárást. Módszerünk képes egy mozgó mikroorganizmus lokális koordináta-rendszerében definiált környezetváltozásából megállapítani az organizmus pozícióját a globális koordináta-rendszerben. Ehhez egy statikus bázist alkottunk meg a környezetben található referenciapontokból. Ezt követően a bázist szintén a baricentrikus megközelítésünk felhasználásával frissítettük az új referenciapontok megjelenése esetén. Létrehoztunk továbbá egy megjelenítő keretrendszert, amely az új algoritmusunk felhasználásával képes a lokális koordináta-rendszerben definiált speciális mozdulatsorozatot tartalmazó egyedi adathalmaz háromdimenziós megjelenítésére. A rendszer továbbá a hatékony adatelemzést is elősegíti számos eszközön keresztül. A lokalizációs algoritmusunk és a keretrendszerünk részleteit a 4.2. és a 4.3. alfejezetek prezentálják, valamint a 2020-ban [56] és 2021-ben [57] megjelent publikációink foglalják össze. Vizualizációs keretrendszerünk referenciaimplementációja publikusan is elérhető[2].

[1] https://github.com/bodonyiandi94/LensFlareFramework. Elérés dátuma: 2026. február 26.
[2] https://github.com/bodonyiandi94/AnimalSimulation. Elérés dátuma: 2026. február 26.

6. Summary

In my dissertation, I presented our new results regarding the efficient and physically based rendering of lens flare ghosts (Chapter 3) and a data visualization framework applying our object localization solution based on barycentric coordinates (Chapter 4).

Our work related to rendering lens flares focused on the ghost components of the phenomenon, aiming for physically based and efficient simulation. To ensure real-time usability, we first developed a tile-based approach for the rasterization of the ghosts. To this end, we created an algorithm that processes the ray grid resulting from tracing a sparse ray grid through the optical system. In the first step, we build quadrilaterals out of the ray grid projected onto the sensor. Next, we process the quads with a two-level hierarchical approach. The quads representing low-complexity areas are merged to decrease the amount of data to be processed. In the next step, we group the quads according to their tile overlaps, and by traversing the resulting buffer, we accumulate the contribution of each ghost to the final pixel color. Our tile-based rasterization algorithm significantly increases the throughput of the system. Our new approach is presented in Chapter 3.2 and was published in 2023 [50].

We also developed a new method based on polynomial optics to increase the efficiency of the ray tracing phase by modifying the model elements of the previous approaches. As a result, the fitted polynomial system approximates the phenomenon more accurately than existing solutions and is much more efficiently evaluated on GPU than the analytical ray tracing. It is also capable of handling several characteristics previously unhandled by polynomial approaches, increasing the plausibility of the simulation. To further improve efficiency, we created a polynomial fitting algorithm based on partial zones. Using the developed model and fitting method, the number of required polynomial terms can be reduced, significantly increasing both the speed of the fitting process and the simulation. With the polynomial approach, the cost of ray tracing is shifted to a precomputation step, which only needs to be performed once before the simulation. During rendering, only the polynomial system needs to

be evaluated, which can be done in real time on the GPU. Our new method is presented in Chapter 3.3 and was published in 2024 [54]. The reference implementation of our proposed tiled rendering method and polynomial simulation is publicly available[1].

To solve the object localization problem, we developed a method based on barycentric coordinates. Our barycentric approach can determine the position of a moving microorganism in the global coordinate system based on the changes in its environment defined in its local coordinate system. To achieve our goals, we first created a static basis from the reference points available in the environment. When new reference points appear, we update the basis with our barycentric approach. We also developed a visualization framework that is capable of the three-dimensional rendering of a specific movement defined in a local coordinate system by the application of our barycentric approach. The system also provides a variety of tools to facilitate the efficient data analysis of the simulation data. Our barycentric approach and visualization framework are presented in Chapters 4.2 and 4.3 and were published in 2020 [56] and 2021 [57]. The reference implementation of our visualization framework is available online[2].

[1] https://github.com/bodonyiandi94/LensFlareFramework/. Access date: February 26, 2026
[2] https://github.com/bodonyiandi94/AnimalSimulation. Access date: February 26, 2026

Köszönetnyilvánítás

Ezúton szeretném kifejezni köszönetemet témavezetőmnek, dr. Kunkli Roland Imrének, aki témavezetői munkájával már hosszú ideje támogatja az előrehaladásomat. Magas szintű szakmai igényessége, türelmes útmutatásai és precíz visszajelzései formálták kutatómunkámat, valamint jelentősen hozzájárultak a dolgozatban bemutatott eredmények létrejöttéhez.

Köszönettel tartozom férjemnek, Csoba Istvánnak, aki mindvégig kitartóan bíztatott és támogatott szakmai jelenlétével, valamint a saját előrehaladása alatt megszerzett tapasztalataival. Szeretném továbbá megköszönni a családomnak és a barátaimnak a türelmet és a támogatást, amit az évek során biztosítottak számomra.

Végezetül szeretném még megköszönni a kutatásomat segítő támogatásokat is[1,2].

[1] A kutatást az „Integrált kutatói utánpótlás-képzési program az informatika és számítástudomány diszciplináris területein” (EFOP-3.6.3-VEKOP-16-2017-00002) című projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális alap társfinanszírozásával valósult meg.

[2] Hálásan köszönöm az NVIDIA Corporationnek a kutatáshoz használt, az „NVIDIA GPU Grant” program keretein belül adományozott Titan Xp videokártyát.

Irodalomjegyzék

[1] Pixar. *The imperfect lens: Creating the look of Wall-E*. Wall-E Three-DVD Box. 2008.

[2] E. Pekkarinen és M. Balzer. „Physically Based Lens Flare Rendering in «The Lego Movie 2»". *Proceedings of the 2019 Digital Production Symposium*. DigiPro '19. Los Angeles, USA: ACM, 2019, 1:1–1:3. DOI: https://doi.org/10.1145/3329715.3338881.

[3] G. Zou, H. Bai, Y. Yuan, T. Deng, Z. Yin és J. Wei. „Research on Flare Removal Network Based on Channel Attention Mechanism and Depth-wise Over-parameterized Convolution". *Proceedings of the 4th International Conference on Artificial Intelligence and Computer Engineering (ICAICE 2023)*. New York, USA: ACM, 2024, 919–926. DOI: https://doi.org/10.1145/3652628.3652781.

[4] Y. Dai, Y. Luo, Z. Shangchen, C. Li és C. C. Loy. „Nighttime Smartphone Reflective Flare Removal Using Optical Center Symmetry Prior". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Los Alamitos, USA: IEEE, 2023, 20783–20791. DOI: http://doi.org/10.1109/CVPR52729.2023.01991.

[5] M. Hullin, E. Eisemann, H.-P. Seidel és S. Lee. „Physically-Based Real-Time Lens Flare Rendering". *ACM Transactions on Graphics* 30.4 (2011), 108:1–108:9. DOI: https://doi.org/10.1145/2010324.1965003.

[6] M. B. Hullin, J. Hanika és W. Heidrich. „Polynomial Optics: A Construction Kit for Efficient Ray-Tracing of Lens Systems". *Computer Graphics Forum* 31.4 (2012), 1375–1383. DOI: https://doi.org/10.1111/j.1467-8659.2012.03132.x.

[7] I. Csoba és R. Kunkli. „Efficient Rendering of Ocular Wavefront Aberrations using Tiled Point-Spread Function Splatting". *Computer Graph-*

ics Forum 40.6 (2021), 182–199. DOI: https://doi.org/10.1111/cgf.14267.

[8] O. Olsson és U. Assarsson. „Tiled Shading”. *Journal of Graphics, GPU, and Game Tools* 15.4 (2011), 235–251. DOI: https://doi.org/10.1080/2151237X.2011.621761.

[9] R. Kingslake. *Optics in Photography*. 6. köt. Roca Baton, USA: SPIE Press, 1992. ISBN: 978-0-8194-0763-4. DOI: https://doi.org/10.1117/3.43160.

[10] I. Csoba. „OpenLensFlare: an Open-Source, Lens Flare Designing and Rendering Framework”. *WSCG 2017: Short Papers Proceedings*. Computer Science Research Notes. Plzen, Czech Republic, 2017, 195–203. URL: http://wscg.zcu.cz/WSCG2017/!!_CSRN-2702.pdf. Elérés dátuma: 2025. május 11.

[11] E. Hecht. *Optics, Global Edition*. 5. kiad. Harlow, UK: Pearson Education, 2016. ISBN: 978-0-1339-7722-6.

[12] S. Marschner és P. Shirley. *Fundamentals of Computer Graphics*. 4. kiad. Boca Raton, USA: A K Peters/CRC Press, 2015. ISBN: 978-1-4822-2939-4.

[13] J. Hughes, A. van Dam, M. McGuire, D. Sklar, J. Foley, S. Feiner és K. Akeley. *Computer Graphics: Principles and Practice*. 3. kiad. New Jersey, USA: Addison-Wesley Professional, 2013. ISBN: 978-0-3213-9952-6.

[14] V. Skala. „Barycentric coordinates computation in homogeneous coordinates”. *Computers & Graphics* 32 (2008), 120–127. DOI: https://doi.org/10.1016/j.cag.2007.09.007.

[15] K. Hormann és M. Tarini. „A Quadrilateral Rendering Primitive”. *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*. HWWS ’04. Grenoble, France: Association for Computing Machinery, 2004, 7–14. DOI: https://doi.org/10.1145/1058129.1058131.

[16] E. L. Wachspress. *A Rational Finite Element Basis*. New York, USA: Academic Press, 1975. ISBN: 978-0-1272-8950-2.

[17] M. Meyer, H. Lee, A. Barr és M. Desbrun. „Generalized barycentric coordinates and applications”. *Journal of Graphics Tools* 7.1 (2002), 13–22. DOI: https://doi.org/10.1080/10867651.2002.10487551.

[18] C. T. Loop és T. D. DeRose. „A multisided generalization of Bézier surfaces”. *ACM Transactions on Graphics* 8.3 (1989), 204–234. DOI: https://doi.org/10.1145/77055.77059.

[19] B. Haycock, J. L. Campos, N. Koenraad, M. Potter és S. Advani. „Creating headlight glare in a driving simulator”. *Transportation Research Part F: Traffic Psychology and Behaviour* 61 (2019), 93–106. DOI: https://doi.org/10.1016/j.trf.2017.10.006.

[20] T. Ritschel, M. Ihrke, J. R. Frisvad, J. Coppens, K. Myszkowski és H.-P. Seidel. „Temporal Glare: Real-Time Dynamic Simulation of the Scattering in the Human Eye”. *Computer Graphics Forum* 28.2 (2009), 183–192. DOI: https://doi.org/10.1111/j.1467-8659.2009.01357.x.

[21] Y. Wu, Q. He, T. Xue, R. Garg, J. Chen és A. Veeraraghavan. „How to Train Neural Networks for Flare Removal”. *Proceedings of the IEEE/CVF International Conference on Computer Vision*. Los Alamitos, USA: IEEE, 2021, 2219–2227. DOI: https://doi.org/10.1109/ICCV48922.2021.00224.

[22] Z. Jin, F. Huajun, Z. Xu és C. Yueting. „A Data Generation Method for Image Flare Removal Based on Similarity and Centrosymmetric Effect”. *Photonics* 10 (2023), 1072:1–1072:19. DOI: https://doi.org/10.3390/photonics10101072.

[23] Y. Dai, C. Li, Z. Shangchen, R. Fei és C. C. Loy. „Flare7K: A Phenomenological Nighttime Flare Removal Dataset”. *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*. 2022. URL: https://proceedings.neurips.cc/paper_files/paper/2022/

hash/1909ac72220bf5016b6c93f08b66cf36-Abstract-Datasets_and_Benchmarks.html. Elérés dátuma: 2026. február 24.

[24] S. A. Cholewiak, G. D. Love, P. P. Srinivasan, R. Ng és M. S. Banks. „ChromaBlur: Rendering Chromatic Eye Aberration Improves Accommodation and Realism". *ACM Transactions on Graphics* 36.6 (2017), 210:1–210:12. DOI: https://doi.org/10.1145/3130800.3130815.

[25] C. Kolb, D. Mitchell és P. Hanrahan. „A realistic camera model for computer graphics". *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '95. New York, USA: ACM, 1995, 317–324. DOI: https://doi.org/10.1145/218380.218463.

[26] S. Lee, E. Eisemann és H.-P. Seidel. „Real-Time Lens Blur Effects and Focus Control". *ACM Transactions on Graphics* 29.4 (2010), 65:1–65:7. DOI: https://doi.org/10.1145/1778765.1778802.

[27] B. Steinert, H. Dammertz, J. Hanika és H. P. A. Lensch. „General Spectral Camera Lens Simulation". *Computer Graphics Forum* 30.6 (2011), 1643–1654. DOI: https://doi.org/10.1111/j.1467-8659.2011.01851.x.

[28] M. J. Kilgard. *Fast OpenGL-rendering of Lens Flares*. 2000. URL: https://www.opengl.org/archives/resources/features/KilgardTechniques/LensFlare/. Elérés dátuma: 2026. február 24.

[29] Y. King. „2D Lens Flare". *Game Programming Gems*. Newton, USA: Charles River Media, 2000, 515–518. ISBN: 978-1584500490.

[30] C. Maughan. „Texture Masking for Faster Lens Flare". *Game Programming Gems 2*. Newton, USA: Charles River Media, 2001, 474–480. ISBN: 978-1584500544.

[31] D. Sekulic. „Efficient Occlusion Culling". *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*. Glenview, USA: Addison-Wesley, 2004, 487–503. ISBN: 978-0321228321.

[32] C. Oat. „A Steerable Streak Filter”. *ShaderX*^{3}*: Advanced Rendering with DirectX and OpenGL*. 2. köt. Newton, USA: Charles River Media, 2004, 341–348. ISBN: 978-1584503576.

[33] T. Alspach. *Vector-based representation of a lens flare*. US Patent 7,526,417. 2009.

[34] J. Chaumond. *Realistic Camera - Lens Flares*. 2007. URL: https://web.archive.org/web/20210303190357/. Elérés dátuma: 2026. február 24. Eredeti elérés: https://graphics.stanford.edu/wikis/cs348b-07/JulienChaumond/FinalProject Archiválás dátuma: 2024. december 28.

[35] A. Keshmirian. „A Physically-Based Approach for Lens Flare Simulation”. Diplomamunka. University of California, San Diego, 2008. URL: https://escholarship.org/uc/item/5n07m4p6. Elérés dátuma: 2026. február 24.

[36] M. Lendermann, J. Shi Quan Tan, J. Ming Koh és K. Hao Cheong. „Computational Imaging Prediction of Starburst-Effect Diffraction Spikes”. *Scientific Reports* 8 (2018), 16919:1–16919:8. DOI: https://doi.org/10.1038/s41598-018-34400-z.

[37] J. Hanika és C. Dachsbacher. „Efficient Monte Carlo rendering with realistic lenses”. *Computer Graphics Forum* 33.2 (2014), 323–332. DOI: https://doi.org/10.1111/cgf.12301.

[38] A. Walch, C. Luksch, A. Szabo, H. Steinlechner, G. Haaser, M. Schwärzler és S. Maierhofer. „Lens flare prediction based on measurements with real-time visualization”. *The Visual Computer* 34.9 (2018), 1155–1164. DOI: https://doi.org/10.1007/s00371-018-1552-4.

[39] S. Lee és E. Eisemann. „Practical Real-Time Lens-Flare Rendering”. *Computer Graphics Forum* 32.4 (2013), 1–6. DOI: https://doi.org/10.1111/cgf.12145.

[40] P. Hennessy. *Implementation Notes: Physically Based Lens Flares*. 2015. URL: https://placeholderart.wordpress.com/2015/01/19/

implementation - notes - physically - based - lens - flares. Elérés dátuma: 2026. február 24.

[41] M. Kakimoto, K. Matsuoka, T. Nishita, T. Naemura és H. Harashima. „Glare Generation Based on Wave Optics”. *Computer Graphics Forum* 24.2 (2005), 185–193. DOI: https://doi.org/10.1111/j.1467-8659.2005.00842.x.

[42] T. J. T. P. van den Berg, M. P. J. Hagenouw és J. E. Coppens. „The Ciliary Corona: Physical Model and Simulation of the Fine Needles Radiating from Point Light Sources”. *Investigative Ophthalmology & Visual Science* 46.7 (2005), 2627–2632. DOI: https://doi.org/10.1167/iovs.04-0935.

[43] H. Joo, K. Soonhyeon, S. Lee, E. Eisemann és S. Lee. „Efficient Ray Tracing Through Aspheric Lenses and Imperfect Bokeh Synthesis”. *Computer Graphics Forum* 35.4 (2016), 99–105. DOI: https://doi.org/10.1111/cgf.12953.

[44] L. Scandolo, S. Lee és E. Eisemann. „Quad-Based Fourier Transform for Efficient Diffraction Synthesis”. *Computer Graphics Forum* 37.4 (2018), 167–176. DOI: https://doi.org/10.1111/cgf.13484.

[45] E. Schrade, J. Hanika és C. Dachsbacher. „Sparse high-degree polynomials for wide-angle lenses”. *Computer Graphics Forum* 35.4 (2016), 89–97. DOI: https://doi.org/10.1111/cgf.12952.

[46] Q. Zheng és C. Zheng. „Adaptive sparse polynomial regression for camera lens simulation”. *The Visual Computer* 33 (2017), 715–724. DOI: https://doi.org/10.1007/s00371-017-1402-9.

[47] T. Goossens, Z. Lyu, J. Ko, G. C. Wan, J. Farrell és B. Wandell. „Ray-transfer functions for camera simulation of 3D scenes with hidden lens design”. *Optics Express* 30.13 (2022), 24031–24047. DOI: https://doi.org/10.1364/OE.457496.

[48] S. Dilorio. „Optimizations for Rendering Realistic Lens Flares in Polynomial Optics”. Honors thesis. Union College Schaffer Library Special

Collections. Schenectady, NY, USA, 2015. URL: https://arches.union.edu/do/fff31b3d-3ecb-4c4a-979d-b550dc65e38e#mode/2up. Elérés dátuma: 2026. február 24.

[49] E. Sabatschus. „Polyflare: Sparse Polynomial Modeling for Efficient Approximate Lens Flare Rendering". Szakdolgozat. University of Bonn, Bonn, Germany, 2022. URL: https://emmabyte.de/projects/polyflare/thesis.pdf. Elérés dátuma: 2026. február 24.

[50] A. Bodonyi és R. Kunkli. „Efficient tile-based rendering of lens flare ghosts". *Computers & Graphics* 115 (2023), 472–483. DOI: https://doi.org/10.1016/j.cag.2023.07.019.

[51] K. E. Hillesland és J. C. Yang. „Texel Shading". *Eurographics 2016 – Short papers*. Goslar, DEU: The Eurographics Association, 2016, 73–76. DOI: http://doi.org/10.2312/egsh.20161018.

[52] P. Clarberg, R. Toth, J. Hasselgren, J. Nilsson és T. Akenine-Möller. „AMFS: adaptive multi-frequency shading for future graphics processors". *ACM Transactions on Graphics* 33.4 (2014), 141:1–141:12. DOI: https://doi.org/10.1145/2601097.2601214.

[53] U. Sara, M. Akter és M. S. Uddin. „Image Quality Assessment through FSIM, SSIM, MSE and PSNR–A Comparative Study". *Journal of Computer and Communications* 7.3 (2019), 8–18. DOI: https://doi.org/10.4236/jcc.2019.73002.

[54] A. Bodonyi, I. Csoba és R. Kunkli. „Real-time ray transfer for lens flare rendering using sparse polynomials". *The Visual Computer* 41.5 (2025), 3645–3662. DOI: https://doi.org/10.1007/s00371-024-03625-7.

[55] T. Ishikawa. „Suspension biomechanics of swimming microbes". *Journal of The Royal Society Interface* 6.39 (2009), 815–834. DOI: https://doi.org/10.1098/rsif.2009.0223.

[56] A. Bodonyi és R. Kunkli. „Efficient object location determination and error analysis based on barycentric coordinates". *Visual Computing for*

Industry, Biomedicine, and Art 3 (2020), 18:1–18:7. DOI: https://doi.org/10.1186/s42492-020-00052-y.

[57] A. Bodonyi, Gy. Kurucz, G. Holló és R. Kunkli. „A barycentric coordinates-based visualization framework for movement of microscopic organisms”. *Annales Mathematicae et Informaticae* 53 (2021), 61–72. DOI: https://doi.org/10.33039/ami.2021.04.006.

[58] M. O. Ward, G. Grinstein és D. Keim. *Interactive Data Visualization: Foundations, Techniques, and Applications*. 2. kiad. Boca Raton, USA: A K Peters/CRC Press, 2015. ISBN: 978-1-4822-5737-3.

[59] G. Holló és M. Novák. „The manoeuvrability hypothesis to explain the maintenance of bilateral symmetry in animal evolution”. *Biology Direct* 7.1 (2012), 22:1–22:7. DOI: https://doi.org/10.1186/1745-6150-7-22.

[60] G. Holló. „A new paradigm for animal symmetry”. *Interface Focus* 5.6 (2015), 20150032:1–20150032:10. DOI: https://doi.org/10.1098/rsfs.2015.0032.

[61] D. Scaramuzza és F. Fraundorfer. „Visual Odometry [Tutorial]”. *IEEE Robotics & Automation Magazine* 18.4 (2011), 80–92. DOI: https://doi.org/10.1109/MRA.2011.943233.

[62] M. Achtelik, A. Bachrach, R. He, S. Prentice és N. Roy. „Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments”. *Unmanned Systems Technology XI*. 7332. köt. Orlando, United States: SPIE, 2009, 733219:1–733219:10. DOI: https://doi.org/10.1117/12.819082.

[63] J. Zhang és S. Singh. „Laser–visual–inertial odometry and mapping with high robustness and low drift”. *Journal of Field Robotics* 35.8 (2018), 1242–1264. DOI: https://doi.org/10.1002/rob.21809.

[64] S. Vogel. *Comparative Biomechanics: Life's Physical World*. 2. kiad. Princeton, USA: Princeton University Press, 2013. ISBN: 978-0-6911-5566-1.

[65] A. Compagnoni, V. Sharma, Y. Bao, M. Libera, S. Sukhishvili, P. Bidinger, L. Bioglio és E. Bonelli. „Bioscape: A modeling and simulation language for bacteria-materials interactions”. *Electronic Notes in Theoretical Computer Science* 293 (2013), 35–49. DOI: https://doi.org/10.1016/j.entcs.2013.02.017.

[66] D. Malcolm H., K. Alireza, F. Duane P. és H. Steven E. „A Physics-Based Coordinate Transformation for 3-D Image Matching”. *IEEE Transactions on Medical Imaging* 16.3 (1997), 317–328. DOI: https://doi.org/10.1109/42.585766.

[67] E. F. D'Azevedo. „Optimal Triangular Mesh Generation by Coordinate Transformation”. *SIAM Journal on Scientific Computing* 12.4 (1991), 755–786. DOI: https://doi.org/10.1137/0912040.

[68] L. J. F. Broer és J. A. Kobussen. „Conversion from material to local coordinates as a canonical transformation”. *Applied Scientific Research* 29 (1974), 419–429. DOI: https://doi.org/10.1007/BF00384163.

[69] J. Zhu. „Conversion of Earth-centered Earth-fixed coordinates to geodetic coordinates”. *IEEE Transactions on Aerospace and Electronic Systems* 30.3 (1994), 957–961. DOI: https://doi.org/10.1109/7.303772.

[70] L. Marcin. „Cartesian to geodetic coordinates conversion on a triaxial ellipsoid”. *Journal of Geodesy* 86 (2011), 249–256. DOI: https://doi.org/10.1007/s00190-011-0514-7.

[71] D. E. Manolakis. „Efficient Solution and Performance Analysis of 3-D Position Estimation by Trilateration”. *IEEE Transactions on Aerospace and Electronic Systems* 32.4 (1996), 1239–1248. DOI: https://doi.org/10.1109/7.543845.

[72] Y. Diao, Z. Lin és M. Fu. „A Barycentric Coordinate Based Distributed Localization Algorithm for Sensor Networks”. *IEEE Transactions on Signal Processing* 62.18 (2014), 4760–4771. DOI: https://doi.org/10.1109/TSP.2014.2339797.

[73] Y. Wu, F. Tang és H. Li. „Image-based camera localization: an overview”. *Visual Computing for Industry, Biomedicine, and Art* 1 (2018), 1–8. DOI: https://doi.org/10.1186/s42492-018-0008-z.

[74] F. Thomas és L. Ros. „Revisiting Trilateration for Robot Localization”. *IEEE Transactions on Robotics* 21.1 (2005), 93–101. DOI: https://doi.org/10.1109/TRO.2004.833793.

[75] G. Kang, T. Pérennou és M. Diaz. „Barycentric Location Estimation for Indoors Localization in Opportunistic Wireless Networks”. *2008 Second International Conference on Future Generation Communication and Networking*. Hainan, China: IEEE, 2008, 220–225. DOI: https://doi.org/10.1109/FGCN.2008.85.

[76] W. Tian. „The GPS coordinates conversion and error analysis in radar precision test”. *Radar & Ecm* 2 (2007), 54–56. URL: https://caod.oriprobe.com/articles/21188047/The_GPS_coordinates_conversion_and_error_analysis_in_radar_precision_t.htm. Elérés dátuma: 2026. február 24.

[77] C. Park, D. J. Cho, E. J. Cha, D.-H. Hwang és S. J. Lee. „Error Analysis of 3-Dimensional GPS Attitude Determination System”. *International Journal of Control, Automation, and Systems* 4.4 (2006), 480–485. URL: https://koreascience.kr/article/JAKO200633242345517.page. Elérés dátuma: 2026. február 24.

[78] O. Weber, M. Ben-Chen és C. Gotsman. „Complex Barycentric Coordinates with Applications to Planar Shape Deformation”. *Computer Graphics Forum* 28.2 (2009), 587–597. DOI: https://doi.org/10.1111/j.1467-8659.2009.01399.x.

[79] J. Zhang, B. Deng, Z. Liu, G. Patanè, S. Bouaziz, K. Hormann és L. Liu. „Local barycentric coordinates”. *ACM Transactions on Graphics* 33.6 (2014), 188:1–188:12. DOI: https://doi.org/10.1145/2661229.2661255.

[80] L. Torsten. „On Generalized Barycentric Coordinates and Their Applications in Geometric Modeling”. Disszertáció. MPI Informatik, 2008.

URL: https://diglib.eg.org/server/api/core/bitstreams/419045c9-d97b-47c9-94d0-0cb59cabacbf/content. Elérés dátuma: 2026. február 24.

[81] P. Joshi, M. Meyer, T. DeRose, B. Green és T. Sanocki. „Harmonic Coordinates for Character Articulation”. *ACM Transactions on Graphics* 26.3 (2007), 71:1–71:10. DOI: https://doi.org/10.1145/1276377.1276466.

[82] M. S. Floater. „Parametrization and smooth approximation of surface triangulations”. *Computer Aided Geometric Design* 14.3 (2007), 231–250. DOI: https://doi.org/10.1016/S0167-8396(96)00031-3.

[83] M. Desbrun, M. Meyer és P. Alliez. „Intrinsic Parameterizations of Surface Meshes”. *Computer Graphics Forum* 21.3 (2003), 209–218. DOI: https://doi.org/10.1111/1467-8659.00580.

Az értekezés alapjául szolgáló publikációk

Referált folyóiratcikkek

[F1] **A. Bodonyi**, I. Csoba és R. Kunkli. „Real-time ray transfer for lens flare rendering using sparse polynomials”. *The Visual Computer* 41.5 (2025), 3645–3662. DOI: https://doi.org/10.1007/s00371-024-03625-7.
Folyóirat besorolása: Q2 (impakt faktor: 3).

[F2] **A. Bodonyi** és R. Kunkli. „Efficient tile-based rendering of lens flare ghosts”. *Computers & Graphics* 115 (2023), 472–483. DOI: https://doi.org/10.1016/j.cag.2023.07.019.
Folyóirat besorolása: Q1 (impakt faktor: 2,5).

[F3] **A. Bodonyi**, Gy. Kurucz, G. Holló és R. Kunkli. „A barycentric coordinates-based visualization framework for movement of microscopic organisms”. *Annales Mathematicae et Informaticae* 53 (2021), 61–72. DOI: https://doi.org/10.33039/ami.2021.04.006.
Folyóirat besorolása: Q3.

[F4] **A. Bodonyi** és R. Kunkli. „Efficient object location determination and error analysis based on barycentric coordinates”. *Visual Computing for Industry, Biomedicine, and Art* 3 (2020), 18:1–18:7. DOI: https://doi.org/10.1186/s42492-020-00052-y.

Konferencia-előadások

[E1] **A. Bodonyi** és R. Kunkli. „Lencsefényfoltok hatékony csempealapú szin-tézise”. XI. Magyar Számítógépes Grafika és Geometria Konferencia (GRAFGEO 2024). Budapest, 2024.

[E2] **A. Bodonyi** és R. Kunkli. „Efficient tile-based rendering of lens flare ghosts”. 18th International Conference on Computer-Aided Design and Computer Graphics (CAD/Graphics 2023). Shanghaj, Kína, 2023.

[E3] **A. Bodonyi** és R. Kunkli. „Efficient tile-based lens flare rendering”. 2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS 2022). Debrecen, 2022.

[E4] **A. Bodonyi**, Gy. Kurucz, G. Holló és R. Kunkli. „Implementing a Barycentric Coordinates-based Visualization Framework for Movement of Microscopic Organisms”. 2020 IEEE 1st Conference on Information Technology and Data Science (CITDS 2020). Debrecen, 2020.

Poszterprezentáció

[P1] R. Kunkli és **A. Bodonyi**. „A Barycentric Coordinates based Object Location Determination Method for Animation Purposes”. The 12th Asian Forum on Graphic Science (AFGS 2019). Kunming, Kína, 2019.

DEBRECENI EGYETEM
EGYETEMI ÉS NEMZETI KÖNYVTÁR
H-4002 Debrecen, Egyetem tér 1, Pf.: 400
Tel.: 52/410-443, e-mail: publikaciok@lib.unideb.hu

Nyilvántartási szám: DEENK/330/2025.PL
Tárgy: PhD Publikációs Lista

Jelölt: Csoba-Bodonyi Andrea Beatrix
Doktori Iskola: Informatikai Tudományok Doktori Iskola
MTMT azonosító: 10073731

A PhD értekezés alapjául szolgáló közlemények

Idegen nyelvű tudományos közlemények hazai folyóiratban (1)

1. **Bodonyi, A. B.**, Kurucz, G., Holló, G., Kunkli, R.: A barycentric coordinates-based visualization framework for movement of microscopic organisms.
Ann. Math. Inform. 53, 61-72, 2021. ISSN: 1787-5021.
DOI: http://dx.doi.org/10.33039/ami.2021.04.006

Idegen nyelvű tudományos közlemények külföldi folyóiratban (3)

2. **Bodonyi, A. B.**, Csoba, I., Kunkli, R.: Real-time ray transfer for lens flare rendering using sparse polynomials.
Visual Comput. 41 (5), 3645-3662, 2025. ISSN: 0178-2789.
DOI: http://dx.doi.org/10.1007/s00371-024-03625-7
IF: 3 (2023)

3. **Bodonyi, A. B.**, Kunkli, R.: Efficient tile-based rendering of lens flare ghosts.
Comput. Graph.-UK. 115, 472-483, 2023. ISSN: 0097-8493.
DOI: http://dx.doi.org/10.1016/j.cag.2023.07.019
IF: 2.5

4. **Bodonyi, A. B.**, Kunkli, R.: Efficient object location determination and error analysis based on barycentric coordinates.
Vis. Comput. Ind. Biomed. Art. 3 (1), 1-7, 2020. EISSN: 2524-4442.
DOI: http://dx.doi.org/10.1186/s42492-020-00052-y

Idegen nyelvű absztrakt kiadványok (2)

5. **Bodonyi, A. B.**, Kurucz, G., Holló, G., Kunkli, R.: Implementing a barycentric coordinates-based visualization framework for movement of microscopic organisms.
In: The 1st Conference on Information Technology and Data Science. Ed.: Fazekas István, Hajdu András, Debreceni Egyetem, Debrecen, 40-42, 2020.

DEBRECENI EGYETEM Egyetemi és Nemzeti Könyvtár

DEBRECENI EGYETEM
EGYETEMI ÉS NEMZETI KÖNYVTÁR
H-4002 Debrecen, Egyetem tér 1, Pf.: 400
Tel.: 52/410-443, e-mail: publikaciok@lib.unideb.hu

6. Kunkli, R., **Bodonyi, A. B.**: A Barycentric Coordinates based Object Location Determination Method for Animation Purposes.
In: Graphics and Application : the 12th Asian Forum on Graphic Science (AFGS 2019). Ed.: Baoling Han, Xiao Luo, Hongliang Fan, Beijing Institute of Technology Press, China Graphics Society, Beijing, 149-150, 2019. ISBN: 9787893910319

További közlemények

Idegen nyelvű absztrakt kiadványok (1)

7. **Bodonyi, A. B.**, Kunkli, R.: Improved Algorithm for Simulating Glare in the Human Eye.
In: Graphics and Application : the 12th Asian Forum on Graphic Science (AFGS 2019). Ed.: Baoling Han, Xiao Luo, Hongliang Fan, Beijing Institute of Technology Press, China Graphics Society, Beijing, 130-132, 2019. ISBN: 9787893910319

A közlő folyóiratok összesített impakt faktora: 5,5
A közlő folyóiratok összesített impakt faktora (az értekezés alapjául szolgáló közleményekre): 5,5

A DEENK a Jelölt által a Tudóstérbe feltöltött adatok bibliográfiai és tudománymetriai ellenőrzését a tudományos adatbázisok és a Journal Citation Reports Impact Factor lista alapján elvégezte.

Debrecen, 2025.05.27.

DEBRECENI EGYETEM Egyetemi és Nemzeti Könyvtár