Képalkotó rendszerek aberrációinak hatékony szimulációját támogató algoritmusok és szoftverek fejlesztése

Egyetemi doktori (PhD) értekezés

A szerző neve: Csoba István
A témavezető neve: Dr. Kunkli Roland Imre

DEBRECENI EGYETEM
Természettudományi és Informatikai Doktori Tanács
Informatikai Tudományok Doktori Iskola
Debrecen, 2024.

Ezen értekezést a Debreceni Egyetem Természettudományi és Informatikai Doktori Tanács Informatikai Tudományok Doktori Iskola Diszkrét matematika, adatfeldolgozás 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, 2024. . . . . .

. . . . . .
Csoba István
jelölt

Tanúsítom, hogy Csoba István doktorjelölt 2017–2021 között a fent megnevezett Doktori Iskola Diszkrét matematika, adatfeldolgozás 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, 2024. . . . . .

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

Képalkotó rendszerek aberrációinak hatékony szimulációját támogató algoritmusok és szoftverek fejlesztése

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

Írta: Csoba István okleveles programtervező informatikus

Készült a Debreceni Egyetem
Informatikai Tudományok doktori iskolája
(*Diszkrét matematika, adatfeldolgozás 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

Bevezetés

A képalkotó rendszerek számítógépes szimulációja hosszú múltra visszatekintő probléma. Legyen szó a rendszer teljesítményének közvetlen vagy valamilyen számszerűsíthető metrika alapján történő vizsgálatáról, a szimulációs eszközök nagymértékben elősegítik ezeknek a feladatoknak az elvégezhetőségét. Hatalmas fontossága van tehát a képalkotó rendszerek kimenetét hatékonyan és hitelesen szimuláló számítógépes módszereknek.

Kiemelt szerepe van a képalkotó rendszerek között az emberi szemnek, hiszen az meghatározó eleme mindennapi életünknek. A látásszimuláció számos területen felhasználható [1–3], mint például a látástesztelés [4] és látásélesség-metrikák számítása [5–7], az emberi szem tulajdonságainak és működési mechanizmusainak vizsgálata [8–10], valamint a szembetegségek vizualizációja szintetikus [11–15], virtuális valóság [16, 17] és kiterjesztett valóság [17, 18] környezetekben.

Az emberi szem rendkívül érzékeny, képalkotó kvalitását külső és belső hatások nagymértékben befolyásolják. Éppen ezért kardinális szerepe van a testreszabható látásszimuláló algoritmusoknak, amelyek képesek egy adott egyén látását hitelesen reprodukáló kimeneteket előállítani. A fent leírt kérdéskörökön túl ezek a módszerek számos további területen is alkalmazhatók. Egyrészt személyre szabott tervezéssel csökkenthető a szembeavatkozások utáni komplikációk esélye, amelyek felmerülésének lehetősége hangsúlyos kockázati tényező például lézeres szemműtétek [19] és intraokulárislencse-beültetések [20] esetén. Továbbá a progresszív lencsék [21–23], a látásjavító kijelzők [24, 25], a fejre szerelt kijelzők [26] és a holografikus lencsék [27, 28] alkalmazásakor szintén nagyban növelhető a sikeresség személyre szabott látásszimuláló technikákkal. Ezenkívül a fejre szerelt kijelzők használatakor jelentkező diszkomfortérzet is jelentősen csökkenthető speciális betűtípusokkal [29] és a megjelenített képek módosításával a felhasználó vizuális aberrációi alapján [30–33]. Az utóbbi módszerrel akár a kijelző szemüveg nélküli használata is lehetővé tehető [34, 35], tovább fokozva a kijelző használhatóságát.

Az emberi látást szimuláló módszerek széleskörű alkalmazhatóságának hála mára már számos eljárás létezik az aberrációkkal terhelt emberi látásnak megfelelő képek előállítására. Ezen technikák bemutatását, a főbb tulajdonságaik vizsgálatát, valamint a lehetséges felhasználási területeik összefoglalását a közelmúltban elérhetővé tettük a tudományos közösség számára [36]. Gyakran felmerülő

probléma a látásszimuláló algoritmusokban a személyreszabhatóság teljes figyelmen kívül hagyása, a valósidejű rendszerek számára alkalmatlan sebesség, a szimulálható aberrációk típusának korlátoltsága, valamint a perifériás látás támogatásának hiánya.

Mindezen problémák megoldására létrehoztunk több önállóan is alkalmazható eljárást, amelyek együttesen egy, az imént említett hiányosságok mindegyikét orvosolni képes rendszert alkotnak. Céljaink eléréséhez megalkottunk egy paraméteres szemmodellt és két hozzá tartozó paraméterkereső módszert, amelyekkel képesek vagyunk megbecsülni a szimulált szem fizikai struktúráját mindössze egyetlen bemeneti aberrációmérés adataiból. Az aberrációkkal terhelt látáshoz hű renderelés problémájára először megalkottunk egy komplex fázorokat alkalmazó módszert, amellyel a szem alacsonyrendű aberrációi (defókusz, asztigmia) valós időben szimulálhatók. Ezt követően létrehoztunk egy másik, a valós pontszórásfüggvény képpontonkénti közelítésére épülő algoritmust is, amely képes tetszőleges vizuális aberráció és a perifériás látás valósidejű szimulációjára. Végezetül a rendereléshez szükséges pontszórásfüggvények hatékony előállítására kifejlesztettünk egy GPU-alapú módszert, amellyel az előfeldolgozási idő nagyságrendekkel csökkenthető és akár az interaktív teljesítmény is megvalósítható. Új algoritmusaink részleteit és elért eredményeinket az 1. fejezetben ismertetem.

Az emberi szemen kívül a kamerarendszerek is rendkívül fontos képalkotó rendszerek, hiszen azok a fotográfia és a cinematográfia területének elengedhetetlen eszközei. A kamerarendszerek működésükből fakadóan számos sajátossággal rendelkeznek, amelyek hangsúlyos művészi eszköznek számítanak. Ezenkívül a kamerák sajátosságainak számítógépes reprodukciójával a szintetikus képek valószerűsége nagymértékben fokozható [37], ennélfogva azokat a gyakorlatban is rendszeresen alkalmazzák [38, 39]. Mindezen okokból kifolyólag a tudományos közösség jelentős erőfeszítéseket tesz a kamerák sajátosságainak számítógépes szimulációjára. Ezeknek az algoritmusoknak a részleteibe betekintést nyújt például Akenine-Möller és mtsai. [40] könyve.

Mindezen sajátosságok közül munkánk során a lencsefényfoltok szintézisére fókuszáltunk. Ennek oka, hogy a szóban forgó fényfoltok hangsúlyos szerepet játszanak a szintetikus képek hitelességének maximalizációjában, a jelenség fizikai alapú szimulációja pedig kiváltképp komplex feladat. Ennélfogva először elkészítettem egy nyílt forrású keretrendszert[1], amely egy használatra kész programkönyvtárat nyújt a fizikai alapú lencsefényfolt-szimulációhoz interaktív és

[1] https://github.com/csobaistvan/OpenLensFlare. Elérés dátuma: 2024. június 3.

valósidejű rendszerek számára. Ezenkívül rendszerem egy dedikált tervezői és vizualizációs modullal a fényfoltok megjelenésének tervezését és a keletkezésük mélyebb megértését is lehetővé teszi. A szóban forgó módszerek között kiemelten fontos szerepe van Hullin és mtsai. ritka rácsokkal végzett sugárkövetéses módszerének [41]. Ezt az algoritmust gyakran használják valós alkalmazásokban [39] és a keretrendszerem egyik fő szimulációs eljárását is képezi. Hullin és mtsai. módszerének az egyik hangsúlyos problémája egy hosszadalmas és a szimuláció minőségét nagymértékben befolyásoló előfeldolgozási lépés. A probléma orvoslására megalkottunk egy olyan alternatív megoldást, amellyel a folyamat eredménye magas pontossággal és teljesen valósidejű módon approximálható. Munkánk és elért eredményeink részleteit a 2. fejezetben prezentálom.

1. Vizuális aberrációkkal terhelt emberi látás személyre szabott szimulációja

Bár az emberi látás gyors és személyre szabott szimulációjának számos felhasználási területe létezik, a feladat hiteles és mindenre kiterjedő elvégzése egy messzemenően komplex probléma. Munkánkkal a szükséges feladatokat hatékonyan megoldó algoritmusokat hoztunk létre. Az értekezés első részében ezen új eredményeinket mutatom be.

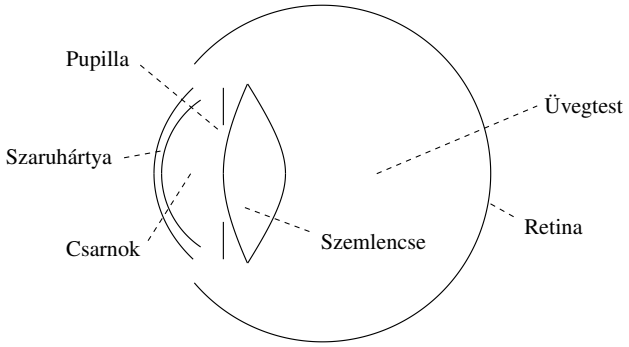
1.1. Matematikai háttér

Értekezésem ezen részében röviden ismertetem a vizuális aberrációkkal terhelt emberi látás szimulációját érintő új eredményeink bemutatásához szükséges legfőbb alapfogalmakat.

1.1.1. Az emberi szem optikai modellje

Az emberi látás optikai szimulációjához először is szükségünk van az emberi szem optikai modelljére. Erre a célra a szakirodalomban számos úgynevezett *sematikus szemmodell* elérhető [42], amelyeket jellemzően az emberi szem klinikai méréseiből készített populációs adatbázisok alapján hozzák létre. A sematikus szemmodellek főbb elemeit tipikusan a fény fókuszálását végző szaruhártya és szemlencse, a fény mennyiségét szabályozó pupilla, illetve a fénysugarak érzékelését végző retina képezik. Ezenkívül fontos szerepe van a szaruhártya és a szemlencse között elhelyezkedő csarnoknak, valamint a lencsét és a retinát összekötő üvegtestnek. Az egyes elemek viszonyát az 1.1. ábra szemlélteti.

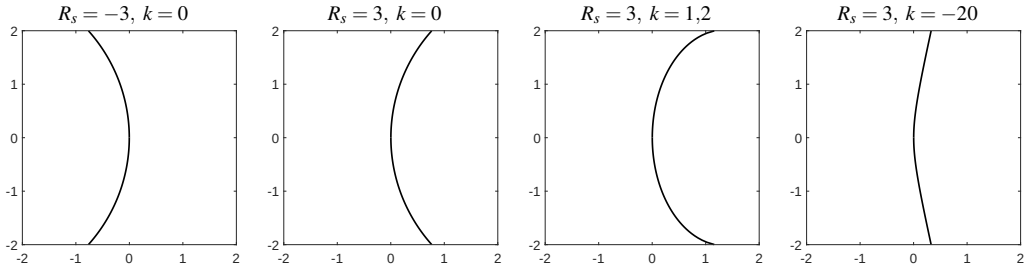
Mindezen sematikus szemmodellek között kiemelt szerepe van Navarro aszférikus modelljének [43], ugyanis az értekezés későbbi részében bemutatott saját paraméteres szemmodellünket is annak kiterjesztésével alkottuk meg. Matematikailag a modell legfőbb fénytörő felületei (a szaruhártya és a szemlencse elülső és hátsó felszíne) kvadratikus forgásfelületek darabjaiként írhatók le, amihez munkánk során az alábbi formulát alkalmaztuk [44]:



1.1. ábra. Az emberi szem sematikus rajza. Legfőbb elemeit a szaruhártya, a pupilla, a szemlencse és a retina, valamint az elemeket összekötő csarnok és üvegtest képezik.

$$z(x,y) = \frac{x^2+y^2}{R_s \cdot \left(1+\sqrt{1-(1+k)\frac{x^2+y^2}{R_s^2}}\right)}, \tag{1.1}$$

ahol R_s a felület görbületi sugara, k pedig az úgynevezett *kónikus konstans*, amelynek megfelelően a felület hiperboloid ($k<-1$), paraboloid ($k=-1$), orsószferoid ($-1<k<0$), gömb ($k=0$) vagy lencseszferoid ($k>0$). Az eltérő R_s és k paraméterértékek hatását az 1.2. ábra szemlélteti.



1.2. ábra. Példa az (1.1) egyenlettel definiált felületek keresztmetszetére, eltérő R_s és k paraméterértékek felhasználásával.

1.1.2. Vizuális aberrációk

A képalkotó rendszerek jellemzésének fontos eszközei a rendszer által generált hullámfront-aberrációk. Koncepcionálisan egy hullámfront azonos fázisban mozgó pontokat jelöl, amelyek a térben egy felületet képeznek. A hullámfront aberrációja pedig a hullámfront egy előre kijelölt referenciától való eltérését jelöli.

A hullámfront aberrációit a szakirodalomban jellemzően a W hullámaberrációs vagy a Φ fázisaberrációs alakban adják meg [45]. Az aberrációs függvény mindkét esetben egy, az optikai rendszer belépő pupillája fölött értelmezett felülettel írható le, amely a tényleges és a referencia hullámfront pontonkénti eltérését méri. A két alak közötti fő különbség az aberrációk mértékegységében található, ugyanis míg W a generált és a referencia fénysugarak optikai úthosszkülönbségét írja le (jellemzően mikrométerekben mérve), addig Φ a fénysugarak fáziseltérését adja meg. A két leírási mód azonban könnyen származtatható egymásból az alábbi módon [45]:

$$\Phi = 2\pi W/\lambda, \tag{1.2}$$

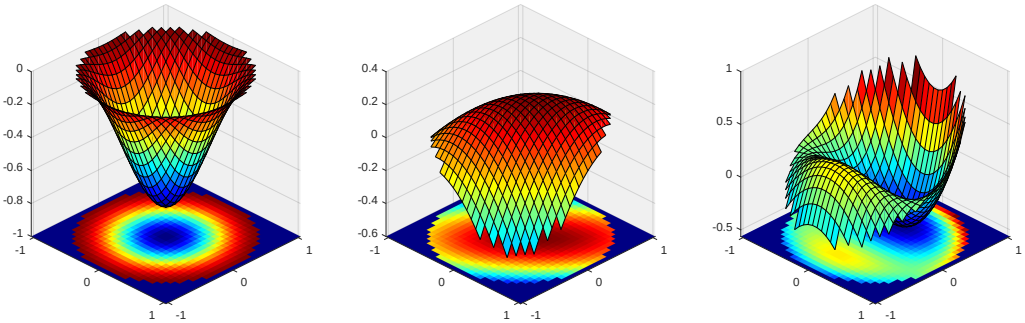
ahol λ a fény hullámhosszát jelöli. Az aberrációs függvény leírására a szakirodalomban jellemzően Zernike-polinomokat [46] alkalmaznak [47]:

$$\Phi(\rho,\phi) = \sum_{n,m} \alpha_n^m \cdot U_n^m(\rho,\phi), \tag{1.3}$$

$$U_n^m(\rho,\phi) = R_n^{|m|}(\rho) \begin{cases} \cos m\phi, & \text{ha } m \geq 0, \\ -\sin m\phi, & \text{egyébként,} \end{cases} \tag{1.4}$$

$$R_n^m(\rho) = \sum_k^{\frac{n-m}{2}} \frac{(-1)^k \cdot (n-k)!}{k! \cdot \left(\frac{n-m}{2}-k\right)! \cdot \left(\frac{n+m}{2}-k\right)!} \cdot \rho^{n-2k}, \tag{1.5}$$

ahol (ρ,ϕ) normalizált polárkoordinátákat jelöl a síkban, U_n^m ($n \geq |m| \geq 0$, $n-|m|$ páros) az n-edik radiális és m-edik azimutális fokú Zernike-polinom, α_n^m pedig a hozzá tartozó Zernike-együttható. Három Zernike-polinomokkal definiált példa aberrációs felületet szemléltet az 1.3. ábra.



1.3. ábra. Három példa az (1.3) egyenlettel definiált aberrációs felületre.

Az emberi szem kiemelt szerepe miatt a szem hullámfront-aberrációinak (amelyre gyakran alkalmazzák a *vizuális aberráció* elnevezést) vizsgálata hangsúlyos kutatási területet jelent és jelentős gyakorlati fontossággal bír. A vizuális aberrációkat a gyakorlatban jellemzően hullámfrontszenzorokkal határozzák meg [47], amely során a szemből kilépő hullámfront síktól való eltérését mérik. Végezetül az így keletkező aberrációs felülethez tartozó Zernike-együtthatókat tipikusan legkisebb négyzetes közelítéssel [48] állítják elő.

Fontos kiemelnem, hogy a vizuális aberrációk meghatározásához költséges eszközök szükségesek, így az jellemzően csak klinikai környezetben végezhető el. Ugyanakkor a szükséges eszközök hiányában populációs adatok beszerezhetők a szakirodalomban elérhető tanulmányokból [49], illetve bizonyos együtthatók megbecsülhetők a látást korrigáló szemüveg paramétereiből [47].

1.1.3. A pontszórásfüggvény kiszámítása

Egy képalkotó rendszer pontszórásfüggvénye egy diffrakciós mintázat, amelyet a rendszer egy ideális pontszerű fényforrás vizsgálatakor generál. A pontszórásfüggvény azonban nagymértékben függ számos tényezőtől, mint például az objektum- és fókusztávolság, a pupillaátmérő, illetve a fény hullámhossza és beesési iránya. Felhasználását tekintve az emberi szem pontszórásfüggvényének meghatározó szerepe van az értekezés bevezetésében említett feladatok (mint például a látásélesség számszerűsítése [5, 6], a látás szimulációja [14, 31], vagy pedig a szem aberrációinak kompenzációja [24, 25]) elvégzésében.

A pontszórásfüggvény kiszámításának számos módja elérhető a szakirodalomban. Ezen módszerek közül a gyors Fourier-transzformációt (FFT) [50] és a kiterjesztett Nijboer–Zernike-féle (ENZ) módszert [51] a gyakorlatban is rendszeresen alkalmazzák [50, 52]. Mivel az FFT-alapú megközelítés csak korlátolt esetekben alkalmazható [52], így munkánk során az ENZ módszert alkalmaztuk. A továbbiakban röviden felidézem az ENZ módszer emberi szem pontszórásfüggvényeivel kapcsolatos főbb elméleti eredményeit.

Először is egy aberrációkkal terhelt optikai rendszer általánosított pupillafüggvénye az alábbi komplex értékű függvény [45]:

$$\mathscr{P}(\rho,\phi) = A(\rho,\phi)\exp\{i\Phi(\rho,\phi)\}, \tag{1.6}$$

ahol (ρ,ϕ) normalizált polárkoordinátákat jelöl a kilépő pupillán, A az amplitúdóátviteli függvény (a pupillán belül egy, azon kívül nulla), Φ pedig az

(1.3) egyenlettel definiált fázisaberrációs függvény. $\mathscr{P}$ Zernike-polinomokkal közvetlenül is felírható [53]:

$$\widehat{\mathscr{P}}(\rho,\phi) = \sum_{n,m} \beta_n^m R_n^{|m|}(\rho) \exp\{im\phi\}, \tag{1.7}$$

ahol β_n^m az n és az m fokhoz tartozó komplex értékű együttható. A fenti definíció nagy előnye, hogy a β együtthatók az (1.3) egyenletben található α együtthatókból egy belső szorzat kiértékelésével egyszerűen kiszámíthatók [54].

Ezt követően az általánosított pupillafüggvényhez tartozó pontszórásfüggvény egy integrál segítségével kiszámítható [45]. A gyakorlatban ez a számítás tipikusan csak numerikus módon végezhető el, nagyban növelve a számítási időt és a generált hibák mértékét. A $\widehat{\mathscr{P}}$ fenti definíciójához tartozó pontszórásfüggvény azonban integrandusok kiértékelése nélkül is meghatározható [55]:

$$U(r,\phi,f) = 2\sum_{n,m} \beta_n^m i^{|m|} V_n^{|m|}(r,f) \exp\{im\phi\}, \tag{1.8}$$

ahol (r,ϕ) polárkoordinátákat jelöl a képsíkban (λ/NA-val normalizálva, ahol NA a numerikus rekesz a képtérben), f a defókuszparaméter, a V_n^m függvények pedig a Zernike radiális polinomok alábbi integráljai [55]:

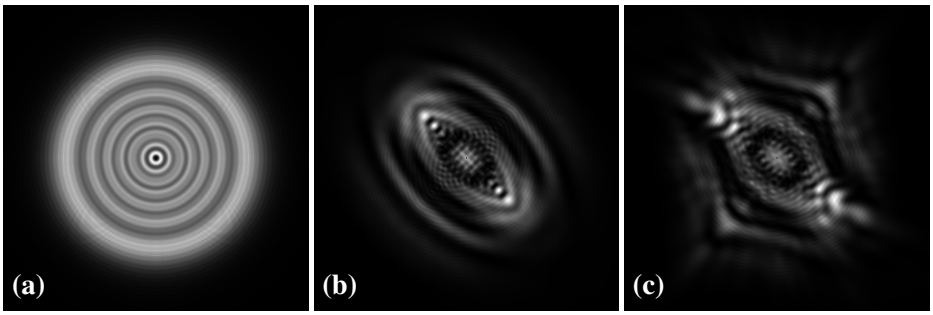
$$V_n^m(r,f) = \int_0^1 \exp\left\{if\rho^2\right\} R_n^{|m|}(\rho) J_m(2\pi r\rho)\, \rho d\rho, \tag{1.9}$$

ahol R_n^m az (1.5) egyenlettel definiált radiális Zernike-polinom, J_m pedig az elsőfajú Bessel-függvény.

Az ENZ modell hatalmas előnye, hogy a V_n^m függvény számos hatékonyan kiértékelhető módon közelíthető. A közelítések közötti egyik fő eltérést a numerikus rekesztől és a defókusztól függő közelítési hibák mértéke képezi. Az emberi szem esetén a numerikus rekesz értéke alacsony (8 mm-es pupillaátmérővel is megközelítőleg 0,23 [56]), a látásszimuláció pedig tetszőlegesen nagy defókusz használatát követeli meg. Mindezen feltételek mellett alkalmazható a skalárdiffrakciós V_n^m közelítés [57]:

$$V_n^m(r,f) = \exp\left(\tfrac{if}{2}\right) \sum_{k=0}^{\infty} (2k+1) i^k j_k\left(\tfrac{f}{2}\right) \sum_{l=\max(0,k-q,p-k)}^{k+p} (-1)^l w_{kl} \frac{J_{m+2l+1}(2\pi r)}{2\pi r}, \tag{1.10}$$

ahol J_k az elsőfajú Bessel-függvény, j_k az elsőfajú gömbi Bessel-függvény, $p = \frac{1}{2}(n-m)$, $q = \frac{1}{2}(n+m)$, a w_{kl} együtthatók pedig a Zernike radiális polinomok (n és m foktól függő) lineáris szorzatai, ahogyan azt az [57] cikk (27) egyenlete definiálja. A külső összegzést a gyakorlatban egy konfigurálható K paraméter korlátozza, amely a közelítés pontosságát kontrollálja. K megválasztását az [57] cikk szerzői az r és az f paraméter legnagyobb abszolút értéke alapján javasolják. Az 1.4. ábra demonstrál néhány ily módon előállított pontszórásfüggvényt.



1.4. ábra. Példa az (1.8) és az (1.10) egyenlettel kiszámított pontszórásfüggvényekre. A Φ függvény (a) nem tartalmaz aberrációkat, (b) csak alacsonyrendű ($n \leq 2$) aberrációkat tartalmaz, (c) alacsony- és magasrendű aberrációkat is tartalmaz.

1.2. Szakmai előzmények, motiváció

1.2.1. Konvolúciós eljárások

Az egyik legelső látásszimulációs eljárás Camp és mtsai. munkájának eredménye [58], akik a szem pontszórásfüggvényével végzett konvolúcióval szimulálták az emberi látást. A konvolúcióhoz szükséges pontszórásfüggvényeket szaruhártya-topográfiás felvételekből számították ki paraxiális sugárkövetéssel. Bár módszerük fontos mérföldkő a látásszimulációs eljárások történelmében, a leegyszerűsített szemmodell nem alkalmas a szem belső aberrációinak kezelésére, a paraxiális sugárkövetés miatt pedig a hullámfront-aberrációs és a diffrakciós hatások sem modellezhetők helyesen. Ezenkívül, mivel módszerük pusztán egy darab pontszórásfüggvényt alkalmazott, így a szimuláció a végtelen távoli objektumsíkra korlátozódik, a perifériás aberrációk figyelembevétele nélkül. Az eljárás problémáinak zömét Greivenkamp és mtsai. orvosolták [5], akik egzakt sugárkövetéssel és egy teljes, aszférikus lencséket alkalmazó szemmodellel

helyettesítették a korábbi megközelítés egyszerűbb elemeit. Módszerük azonban továbbra is csak egyetlen pontszórásfüggvény alkalmazására volt képes, és a sematikus szemmodell személyreszabhatóságának kérdésével sem foglalkozott.

A szimuláció személyreszabhatóságának növelésére Barsky Shack-Hartmann hullámfrontszenzoros méréseket alkalmazott [59]. Módszere a mérésekkel kapott aberrációs felületeken végzett sugárkövetéssel határozza meg az objektumtávolságtól függő pontszórásfüggvények kis elemszámú halmazát. Ezt követően a bemeneti képet mélység szerinti szeletekre bontja, majd a szeleteken konvolúciót hajt végre a mélységtartományhoz tartozó pontszórásfüggvénnyel. Végezetül a feldolgozott szeletekből mélység szerinti alfa keveréssel állítja elő az eredményt. A bemenet szeletekre bontása diszkretizációs műtermékeket okoz a mélységszeletek határain, aminek megoldására Barsky később élkeresést alkalmazott az összefüggő objektumok minden releváns szeleten való elhelyezéséhez [60]. Barsky módszerének fő előnye a korábbi konvolúciós módszerekkel szemben a mélységfüggő pontszórásfüggvények alkalmazásában rejlik, ily módon ugyanis háromdimenziós jelenetek feldolgozása is lehetővé vált. Ugyanakkor a diszkrét rétegek használatából fakadóan a részleges takarási viszonyok nem kezelhetők helyesen. Továbbá a közelítő pontszórásfüggvény nem kezeli a perifériás aberrációkat és szignifikánsan eltérhet a képpontonkénti valós pontszórásfüggvénytől. Végezetül a rétegzés miatt a feldolgozandó képek száma és a hibák elkerüléséhez szükséges élkeresés a szimuláció idejét is számottevően megnövelik.

Rodríguez Celaya és mtsai. az objektumtávolságon túl a horizontális és a vertikális beesési szöget is felhasználták egy háromdimenziós magfüggvényrácsban a perifériás pontszórásfüggvények kezelésére [61]. Módszerük a szimuláció eredményét szintén konvolúcióval állítja elő, amely során a rács trilineáris interpolációjával közelíti a képpontonkénti valós magfüggvényt. Gonzalez Utrera is egy hasonló háromdimenziós pontszórásfüggvényrácsot alkalmazott [23], viszont nagyobb lépésközt használt a beesési szögek mintavételezéséhez, a konvolúciót pedig Barsky algoritmusához hasonlóan mélységfüggő szeletekkel hajtotta végre. A pontszórásfüggvények rendkívül alacsony száma miatt mindkét módszer a perifériás látás csak nagyon durva közelítésére képes. Ennélfogva ezek a megközelítések nem alkalmasak a dinamikusan változó perifériás aberrációk szimulációjára, ami már egy egyszerű rövidlátás vizsgálatához is szükséges lehet [62]. Ezenkívül a képpontonként változó magfüggvény meghatározásának költsége nagyságrendekkel megnöveli a futási időt Barsky módszeréhez képest. Ily módon a két megközelítés felhasználási lehetőségei erősen korlátoltak.

1.2.2. Sugárkövetéses módszerek

Az aberrációkkal terhelt emberi látás szimulációjának másik fő módját a sugárkövetéses eljárások képezik. Az egyik legelső sugárkövetéses megközelítést Mostafawy és mtsai. demonstrálták [63], akik Gullstrand sematikus szemmodelljét és elosztott sugárkövetést alkalmaztak. A személyreszabhatóság növelésére további lencséket helyeztek el a szem előtt, valamint törési zónákat definiáltak a szaruhártya külső felületén. Ezekkel az elemekkel megközelítésük minden kimeneti képponthoz egy sugárrácsot követ végig a szemmodellen. A sugarak a szemmodellen belülről véletlen irányba indulnak és a háromdimenziós jelenet pontjaiban végződnek. A kimeneti képpontok intenzitását a képponthoz tartozó sugárrács mintáinak összegyűjtése szolgáltatja. Módszerük fő hátrányait a sugárkövetés számítási ideje, a véletlen mintavételezésből fakadó zaj, a lapos retinaforma, valamint a sematikus szemmodell egyszerűsége jelentik.

A szemmodell felépítéséből eredő problémák javítására Wu és mtsai. pontosabb, aszférikus elemeket is tartalmazó modelleket alkalmaztak [64], Dias és mtsai. pedig az eltérő retinaalakzatok látásra gyakorolt hatását vizsgálták [65]. Mindezen eredményeket Lian és mtsai. egy nyílt forráskódú látásszimulációs szoftverben gyűjtötték össze [66], amely olyan további aspektusokat is képes kezelni, mint a kromatikus aberráció és a fényelhajlás. Bár ezek a módszerek pontosabban modellezik a valós emberi szemet, a görbült retinafelülettel végzett sugárkövetés jól látható torzulásokat és hiányos régiókat eredményez a kimenetek szélein. Továbbá a sugárkövetés számítási költsége is jellemzően sokkal magasabb a konvolúciós eljárásokénál. A perifériás látás kezelése ily módon azonban triviális, hiszen tetszőleges irányú sugár feldolgozása lehetséges a szemstruktúra birtokában. Ugyanakkor a személyre szabott szimulációhoz szükség van a szem minden elemének fizikai paramétereire, amelyek beszerzése költséges, körülményes és hibára erősen érzékeny klinikai méréseket igényel.

Fink és Micol a szemmodell testreszabhatóságának növelésére Zernike-polinomokkal modellezte a szem fénytörő felületeit [67]. Módszerük a bemeneti kép minden képpontjától egy sugárrácsot követ végig a szemmodellen. A folyamat eredménye egy háromdimenziós pontfelhő a retinán, amelyből a szerzők egy egészséges szemmodellel és ellentétes irányú sugárkövetéssel állították elő a kimenetet. Megközelítésük fő hátrányait a szimuláció egy objektumsíkra való korlátozottsága, illetve a komplex felületek sugárkövetéséhez szükséges iteratív módszer futási ideje képezik. Wei és mtsai. lényeges sebességnövekedést értek el

a fénytörő felületek háromszögelt poligonhálókkal való modellezésével [68], ami lehetővé tette a sugárkövetés gyorsítására fejlesztett algoritmusok alkalmazását. Vu és mtsai. pedig a sugárrács retinán keletkező képének raszterizációjával nagymértékben növelték a szimuláció sebességét és eliminálták a sugárkövetésből keletkező zajt [69]. Fink és Micol algoritmusához hasonlóan Wei és mtsai. módszere, valamint Vu és mtsai. eljárása is egyetlen objektumsíkra korlátolt, amely szignifikáns megkötést jelent a valós felhasználások számára. Ezenkívül, bár az egyes módszerek lehetővé teszik a szem elemeinek precíz modellezését, a személyreszabhatóság megvalósítása továbbra is csak klinikai mérésekkel lehetséges. Végezetül az elemek komplexitása számottevően megnöveli a számítási időket az egyszerű, analitikus képletekkel kiértékelhető felületekkel szemben.

1.2.3. Valósidejű algoritmusok

Az eddig bemutatott módszerek a futási idejük miatt csak offline rendszerekben alkalmazhatók. A valósidejű szimuláció érdekében Tang és Xiao a Gauss-függvény segítségével közelítette a szem alacsonyrendű aberrációkat tartalmazó pontszórásfüggvényeit [70]. A magfüggvény képpontonkénti méretének meghatározásához kisméretű neurális hálókat alkalmaztak. Módszerük futási ideje a Gauss-magfüggvény szeparábilitásának köszönhetően alacsony, a neurális háló felépítéséből fakadóan pedig a periférikus látás és a változó fókusztávolság is támogatott. Az eljárás fő hátránya azonban szintén a Gauss-magfüggvény tulajdonságaiból ered, ugyanis módszerükkel csak az elmosódás mértéke közelíthető. Mivel a Gauss-függvény nem képes a szem pontszórásfüggvényének pontos jellemzőit helyesen reprodukálni, így az algoritmusukkal szimulálható szemállapotok erősen korlátoltak. Ezenkívül a személyreszabhatóság csak a neurális háló létrehozásához használt szemmodell módosításával lehetséges, amihez költséges klinikai mérések és a hálók időigényes újraépítése szükséges.

Lima és mtsai. szintén létrehoztak egy valósidejű eljárást az emberi szem alacsonyrendű aberrációinak szimulációjára, amely képes a részleges takarásból származó műtermékek elkerülésére is [71]. Módszerük egy előszámító lépésben felépít egy fénygyűjtő fa adatszerkezetet a fizikai pupilla mintáinak egy halmazához. Ez az adatszerkezet azt határozza meg, hogy egy adott kimeneti képpont környezetét az egyes rétegeken milyen súllyal szükséges mintavételezni. Ezt követően algoritmusuk az eredmény látásszimulációt egy bemeneti rétegzett RGB-D képhalmazból a fénygyűjtő fák bejárásával kapott kontribúciók összegyűjtésével

állítja elő. Az eljárás fő előnyeit a sebesség és az egyszerű paraméterezhetőség jelentik, ugyanis módszerük képes a korrekciós szemüveg paramétereiből valós időben szimulálni a képpontonkénti elmosódás mértékét. Algoritmusuk fő hátránya, hogy a Gauss-függvényes megközelítéshez hasonlóan nem lehetséges a szem aberrációit pontosan megjeleníteni, illetve a fa adatszerkezet felépítéséből fakadóan a módszer alkalmatlan a perifériás látás kezelésére.

Xiao és mtsai. a közelmúltban konvolúciós neurális hálókat alkalmaztak a szem mélységélességének valósidejű szimulációjára [32]. Módszerük kedvező értékeket mutat sebesség és pontosság tekintetében a tradicionális mélységélesség-szimulációs eljárásokkal szemben. Eljárásukkal azonban csak az egészséges látás szimulálható, mivel a szerzők elsődleges célja a fejre szerelt kijelzők használata közben fellépő diszkomfortérzet csökkentése volt. További kutatás szükséges tehát annak megítélésére, hogy algoritmusuk hogyan terjeszthető ki tetszőleges vizuális aberráció szimulációjára.

1.2.4. Motiváció

Összességében elmondható, hogy a látásszimulációs eljárások széles palettája ellenére a személyreszabhatóság egy rendkívül elhanyagolt aspektusa a tárgyalt módszereknek. A korábbi munkák zöme sematikus szemmodellt vagy hullámfrontszenzoros méréseket alkalmazott, amelyek a hiányos adatok miatt nem képesek egy adott személy látását hitelesen szimulálni. Továbbá, bár több módszer is megemlíti az egyén fizikai szemstruktúrájának reprezentációját, a mérések nehézségei miatt ezek a munkák is csak sematikus szemmodellekre támaszkodtak. Kutatómunkánk egyik legfőbb célkitűzése ezen probléma megoldása volt.

A látásszimulációt igénylő rendszerek további kritikus aspektusa a szimuláció ideje, ugyanis számos gyakorlati felhasználási területen (például a látásjavító eszközök vagy a fejre szerelt kijelzők által okozott diszkomfort csökkentése) elengedhetetlen a valósidejű látásszimulációs sebesség. Bár több valósidejű algoritmus is elérhető a szakirodalomban, azok csak az alacsonyrendű aberrációk szimulációját teszik lehetővé. Ezenkívül a tárgyalt módszerek zömében a szimuláció is korlátolt az optikai tengelyről érkező pontok aberrációira. Tetszőleges aberráció kezelésére csak az offline megközelítések képesek, amelyek még a modern, nagy számítási kapacitással rendelkező hardverekkel sem kellően gyorsak a valósidejű alkalmazások számára. Munkánk másik fő célja egy tetszőleges vizuális aberrációk hatékony szimulációját lehetővé tevő módszer megalkotása volt.

1.3. Fizikai szemstruktúra becslése

A szakirodalomban a fizikai szemstruktúra becslésére elérhető módszerek számos klinikai mérést igényelnek [72, 73]. A mérésekhez szükséges költséges klinikai eszközök gyakran nem állnak rendelkezésre, ami esetünkben is erősen korlátozó tényező volt. Éppen ezért megalkottunk egy olyan algoritmust, amellyel klinikai mérések nélkül is előállítható egy közelítő becslés a szimulált szem fizikai felépítésére. Mindehhez egyedül a relaxált szem vizuális aberrációra támaszkodunk, amelyek beszerezhetők a szakirodalomban elérhető tanulmányokból, vagy származtathatók a korrekciós szemüveg paramétereiből [47].

Továbbá gyakori probléma a zárt forráskódú eszközök használata is (tipikusan az OpticStudio optikai szoftvercsomag [74] megoldásai). Ezek a rendszerek szintén költségesek, speciális ismereteket igényelnek, és a rekonstrukciós folyamatnak csak korlátolt testreszabhatóságát teszik lehetővé. A folyamat maximális átláthatóságának és finomhangolhatóságának érdekében kiemelt hangsúlyt fektettünk a szemrekonstrukció nyílt forrású eszközökkel való elvégzésére is.

Céljaink eléréséhez először készítettünk egy paraméteres szemmodellt, amely a paraméterek helyes megválasztásával képes a vizuális aberrációk széles spektrumának a reprodukálására. Ezenkívül eljárásokat hoztunk létre a modell fókusztávolságának módosítására és aberrációinak meghatározására. Ezt követően megalkottunk két módszert az egyén szemstruktúráját becslő modellparaméterek meghatározására. Először optimalizációt alkalmaztunk, majd a futási idő csökkentésére saját neurális hálózatokat hoztunk létre. A fent leírtaknak megfelelően mindkét módszerünk csak a szem relaxált állapotához tartozó aberrációs adatokat használja bemenetként. Ezen adatok ismeretében algoritmusaink olyan szemparamétereket keresnek, amelyekkel a szemmodell által generált aberrációk alacsony hibával közelítik a bemenetet. Új eredményeinket a 2021-ben [75] és a 2024-ben [76] közzétett cikkünk alapján ismertetem.

1.3.1. Saját paraméteres szemmodellünk

Szemmodell felépítése

Paraméteres szemmodellünk létrehozásakor több szempontot is figyelembe vettünk. Először is igyekeztünk a paraméterek számát alacsonyan tartani, hiszen annak növekedésével a paraméterbecslés nehézsége is emelkedik. Kiemelten fontos azonban, hogy a szemmodellel szimulálható vizuális aberrációk tere minél szé-

lesebb legyen. Végezetül a paraméterkereső módszereink iteratív jellege miatt igyekeztük a szem aberrációinak meghatározási idejét is minimalizálni.

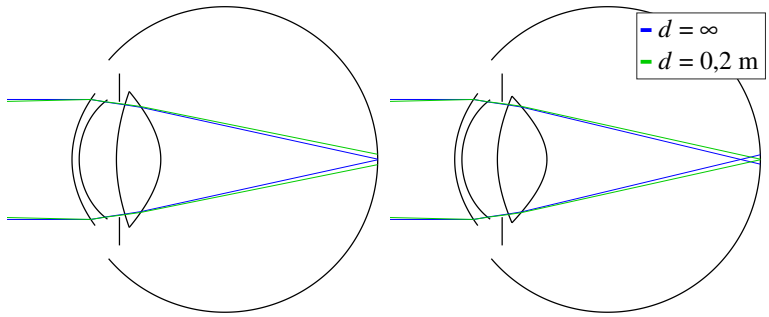
Mindezen szempontok tükrében szemmodellünket Navarro az 1.1.1. alszakaszban ismertetett aszférikus szemmodelljének [43] felhasználásával hoztuk létre. Mivel Navarro modellje az egészséges szem leírására készült, így az aberrációkkal terhelt szem felépítésének szimulációjára a modellt további paraméterekkel egészítettük ki. Először is a modell fénytörő felületeit leíró, az (1.1) egyenlettel definiált felületet kibővítettük egy asztigmia paraméterrel:

$$z_a(x,y) = \frac{\frac{x^2}{R_x} + \frac{y^2}{R_y}}{1 + \sqrt{1 - (1+k) \cdot \left(\frac{x^2}{R_x^2} + \frac{\frac{R_y}{R_x} \cdot y^2}{R_y^2} \right)}}, \tag{1.11}$$

ahol R_x és R_y a felület x és y tengely menti görbületi sugara. Ezután a külső hatásokból fakadó egyenetlenségek modellezéséhez kiterjesztettük a szaruhártya külső felületét egy Zernike-polinomokkal [46] leírt ofszetfelülettel:

$$z_{a,o}(x,y) = z_a(x,y) + \sum_{n,m} \alpha_n^m \cdot U_n^m(x,y), \tag{1.12}$$

ahol U_n^m az 1.1.2. alszakaszban az (1.4) egyenlettel definiált Zernike-polinom, α_n^m pedig a hozzá tartozó Zernike-együttható. Végezetül hozzáadtunk egy optikai tengely körüli forgatást a szaruhártyához, valamint dőlési és decentralizációs együtthatókat a szemlencséhez. Szemmodellünket az 1.5. ábra szemlélteti.



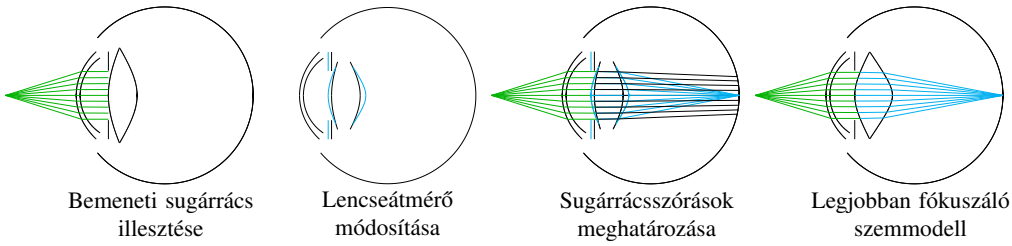
(a) $f = \infty$ **(b)** $f = 0{,}2$ m

1.5. ábra. Szemmodellünk keresztmetszete, egy egészséges szemet szimulálva. Mindkét f fókuszált objektumtávolság esetén a hozzá tartozó d objektumtávolságból érkező sugarak megfelelően fókuszálódnak a retinán.

Szemmodellünk fő korlátját a szemlencse egyszerűsége okozza. Ugyan gradiensindexű lencséket már sikeresen alkalmaztak a pontosság növelésére [77], azok komplexitása és a paramétereinek száma összeegyeztethetetlen a modellünkkel szemben támasztott elvárásokkal (kevés paraméter, aberrációk hatékony meghatározása). Ezenkívül az általunk alkalmazott egyszerű szemlencse átmérőjének módosításakor a lencse többi paramétere hatékonyan, zárt formában kiszámítható, nagymértékben lecsökkentve a fókuszáló folyamat számítási idejét. Végezetül ezen választásunk nem csökkenti számottevően modellünk reprezentációs képességét, ahogyan azt a dolgozat későbbi részeiben demonstrálom.

Fókusztávolság módosítása

Mivel a szemparamétereket becslő módszereink bemeneteként a szem relaxált állapotának vizuális aberrációit használjuk, így az eljárások eredménye is a relaxált szem paraméterei lesznek. A vizsgált szem látásának tetszőleges fókusztávolságú szimulációjához szükségünk van egy módszerre a szem fókusztávolságának módosítására. Ennek az eljárásnak a megalkotásához a valós emberi szem működését vettük alapul, amely a szemlencse átmérőjének módosításával helyezi fókuszba a kívánt objektumot. Fókuszáló módszerünk az ideális szemlencseátmérő kiválasztásához sugárkövetést alkalmaz, ahogyan azt az 1.6. ábra szemlélteti. Algoritmusunk bemenetként a relaxált szemmodellt, a pupillaátmérőt, valamint a kívánt fókuszált objektumtávolságot várja, kimenetként pedig a módosított, a kívánt objektumtávolságra legjobban fókuszáló szemmodellt szolgáltatja.



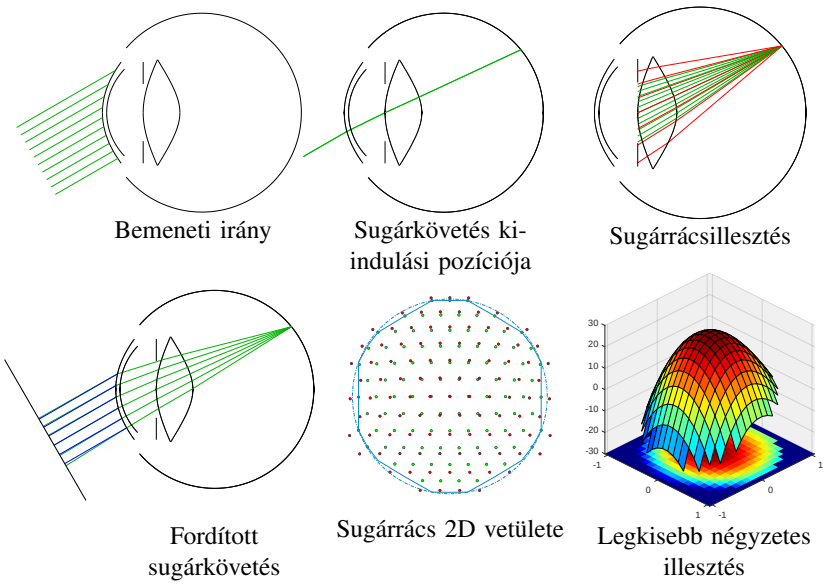
1.6. ábra. Fókusztávolság-módosító algoritmusunk legfőbb lépései.

Első lépésként sugárrácsillesztést hajtunk végre, amely az optikai tengely egy kívánt objektumtávolságú pontjáról induló, a fizikai pupillát lefedő sugarak egy halmazát eredményezi. Ezt követően a relaxált szemlencseátmérőből meghatározzuk a fizikailag lehetséges legkisebb átmérőt [78], majd a modell jelenlegi és legkisebb lencseátmérője közötti tartományt iteratívan többször bejárjuk. Minden

átmérőhöz meghatározzuk a lencse további paramétereit (vastagság, két felületének görbülete stb.) annak konstans térfogata alapján [79], majd az így kapott szemmodellhez kiszámoljuk a sugárrács retinára vetített képének szórását. Algoritmusunk kimenetét a legkisebb szórást eredményező szemmodell adja.

Aberrációk meghatározása

A paraméteres szemmodellünk másik fő eljárása a hullámfront-aberrációk meghatározása. Annak érdekében, hogy szimulációs módszerünk minél pontosabban kövesse a valós klinikai környezetben jellemzően alkalmazott mérések folyamatát, saját eljárásunkat a hullámfrontszenzorok működési elvét modellezve készítettük el. Az így létrehozott algoritmusunk legfőbb lépéseit az 1.7. ábra szemlélteti. Eljárásunk bemenete a szemmodell, illetve a fénysugarak hullámhossza és beesési iránya, kimenete pedig a bemeneti paraméterekkel egy végtelen távoli pontból érkező hullámfront aberrációinak Zernike-együtthatói.



1.7. ábra. Saját paraméteres szemmodellünk hullámfront-aberrációinak meghatározása fordított sugárkövetés segítségével.

Algoritmusunk először megkeresi annak a sugárnak a vetületét a retinán, amely a kívánt beesési irányból érkezik és áthalad a pupilla középpontján. Ezt követően az így kapott retinapontból indítva egy sugárrácsot illesztünk a pupillára, majd a sugárrácsot ellentétes irányban végigkövetjük a szemmodellen. A

sugarak vetületét rögzítjük egy, a beesési irányra merőleges, szemen kívüli síkon. Következő lépésként meghatározzuk a sugarakhoz tartozó referenciapontokat a szaruhártyán kilépő sugarak beesési irányú vetítésével, majd egy ellipszist illesztünk az így kapott referenciapontokra. Ezt követően a tényleges és a referencia vetületi pontok, valamint a pontok optikai úthossza alapján legkisebb négyzetes közelítéssel [48] egy felületet illesztünk az adatokra, amihez a korábban az (1.4) egyenlettel leírt Zernike-polinomokat használjuk bázisfüggvényként. Végezetül az illesztés eredményeként kapott Zernike-együtthatókra skálázást [80] alkalmazunk az ellipszisszerű pupillavetületek kezelésére.

1.3.2. Szemparaméterek becslése optimalizációs módszerrel

A szemparaméterek becslésének kézenfekvő módja valamilyen optimalizációs módszer alkalmazása. Mivel szemmodellünk egy adott paraméterezéshez tartozó aberrációi meghatározhatók, így az optimalizálandó veszteségfüggvényt egyszerűen definiálhatjuk a bemeneti és a generált aberrációs együtthatókból. Ez a megközelítés azonban számos nem triviális kérdést felvet. A továbbiakban ezen problémákat és a megoldásukra alkalmazott módszereinket ismertetem.

Optimalizációs algoritmus választása

Először is fontos megvizsgálnunk az optimalizációs algoritmust. Egy adott probléma tulajdonságaitól függően a szakirodalomban számos optimalizációs módszer elérhető, amelyek lényegesen eltérő tulajdonságokkal rendelkeznek. Az olvasó részletes betekintést kaphat ezen algoritmusok működésébe és tulajdonságaiba Boyd és Vandenberghe [81] könyvéből.

A feladat elvégzésére kezdetben gradiensalapú eljárásokkal kísérleteztünk az ilyen megközelítésekre jellemző gyors konvergencia miatt. A szem felépítéséből és a szaruhártya felületének egyediségéből fakadóan azonban ezek a megközelítések jellemzően hamar megrekedtek egy lokális minimumhelyen. Ezt követően mintakeresést [82] alkalmaztunk, ugyanis a módszer számos kedvező tulajdonsága közül [83] problémánkra nézve hatalmas előnyt jelentett a veszteségfüggvény közvetlen, gradiensek nélküli kiértékelésének képessége.

A mintakereséses eljárás minden iterációban az aktuálisan ismert legjobb pont egy adott méretű környezetében vizsgálja a veszteségfüggvény értékét. Ezen pontok közül a következő iteráció középpontjaként a legkedvezőbbet (legalacsonyabb veszteséggel rendelkezőt) alkalmazza. Az algoritmus a lokális minimum-

helyeket egy kedvezőbb új pont esetén a keresési tartomány növelésével próbálja elkerülni, sikertelen találatkor pedig finomítja a keresést a vizsgált pontok távolságának csökkentésével. Mivel kísérleteink során számos különböző bemenetet megvizsgálva a mintakeresést kellően robusztusnak találtuk, így a továbbiakban ezt az eljárást használtuk.

Az optimalizációs veszteségfüggvény

Az optimalizációs veszteségfüggvény létrehozásakor két fontos tényezőt vettünk figyelembe:

1. Az eredmény szemparaméterek hullámfront-aberrációinak közelsége a bemeneti mérési adatokhoz.
2. A szemmodell elemeinek anatómiai hitelessége, amelyet a korábbi munkákhoz hasonlóan [72, 73] a populációs átlagértékek és szórások alapján határozunk meg.

Mindezek alapján az alábbi veszteségfüggvényt definiáltuk:

$$L = \sum_k \left(w_f f_k |\hat{\alpha}_k - \alpha_k| \right)^2 + \sum_l \left(w_a a_l \max\left(|\bar{x}_l - x_l| - \sigma_l,\, 0 \right) \right)^2, \tag{1.13}$$

ahol α_k és $\hat{\alpha}_k$ a k-adik bemeneti és generált Zernike-együttható, f_k a k-adik együttható veszteséghez való hozzájárulásának súlya, x_l az l-edik modellparaméter, $\bar{x}_l$ és σ_l az l-edik paraméter populációs átlagértéke és szórása (amelyek meghatározásának módját a következő részben ismertetem), a_l pedig az l-edik modellparaméter veszteségének súlya. Ezenkívül w_a és w_f felhasználó által konfigurálható paramétereket jelöl, amelyekkel az anatómiai és a funkcionális tényezők súlya kontrollálható. Kísérleteink során az empirikusan megválasztott $w_a = = 0{,}1$ és $w_f = 200$ értékeket alkalmaztuk.

Paramétertartományok meghatározása

Az anatómiai hitelesség megőrzésének érdekében korlátokat adtunk meg a szemmodellünk paramétereihez. A szükséges értékek meghatározásához megvizsgáltuk a szakirodalomban elérhető populációs eloszlási adatait a szaruhártya [84, 85] és a szemlencse [78, 79, 85, 86] paramétereinek, illetve az egyes elemek optikai tengely menti hosszának [87, 88]. Szemmodellünk paramétereinek ily módon meghatározott korlátait a Függelékben található A.1. táblázat foglalja össze.

Kutatásunk során törekedtünk a korcsoportok és a szembetegségek széles spektrumát figyelembe venni. Bár a korlátolt paraméterek csökkentik az optimalizáció során vizsgált szemmodellek terét, a veszteségfüggvény anatómiai tényezője miatt a korlátokon kívül eső modellek relevanciája egyébként is alacsony. Így tehát úgy ítéljük, hogy módszerünk jó általánosító képességgel rendelkezik, amelyet az 1.3.4. alszakaszban ismertetett, eltérő szemállapotokkal végzett gyakorlati kísérleteink is alátámasztanak. Végezetül módszerünk általánosító képessége szükség esetén tovább növelhető a paraméterkorlátok módosításával.

Fontos megjegyeznem, hogy a szemlencse átmérőjéhez használt adataink eltérnek a szakirodalomban tipikusan fellelhető értékektől. Ennek oka, hogy a szemlencse modellezéséhez használt, alacsony paraméterszámú felületekkel nem írható le pontosan a szem valós alakja. Ennélfogva a felület átmérőjét arányosan megnöveltük, hogy a fókuszáló módszerünkkel generált görbületek megfeleljenek a valós populációs értékeknek.

1.3.3. Interaktív sebesség neurális hálók segítségével

Bár mintakereséses módszerünk fontos eredmény volt a minimális bemeneti adatokkal végzett, teljesen személyre szabott szimuláció eléréséhez, az optimalizáció futási ideje nem tette lehetővé a szemparaméterek interaktív becslését. Ennek oka a költséges sugárkövetést alkalmazó veszteségfüggvény iteratív kiértékelésében rejlik. A probléma kiküszöbölésére a neurális hálók masszív általánosító képességét aknáztuk ki. Célunk egy olyan eljárás megalkotása volt, amellyel a több órás iteratív folyamat eredménye interaktív módon, a másodperc törtrésze alatt közelíthető. Ebben a részben az így létrehozott új eljárásunkat mutatom be.

Általunk alkalmazott neurális hálózatok és feladataik

A látásszimulációhoz szükséges pontszórásfüggvények aberrációs együtthatóinak előállítására az alábbi fő lépéseket végezzük el:

1. Olyan szemparaméterek keresése, amelyekkel a szemmodellünk által generált aberrációs együtthatók közel vannak a szimulált szem relaxált állapotához tartozó aberrációkhoz.
2. A szemlencse paramétereinek módosítása ahhoz, hogy a becsült relaxált szemmodell az egyes bemeneti távolságokra fókuszáljon.
3. Az aberrációs együtthatók meghatározása a kiszámítandó pontszórásfüggvények paramétereihez.

Ezekhez a feladatokhoz először készítettünk egy *szemstruktúrabecslő* hálót, amellyel megbecsülhetjük a relaxált szem paramétereit a bemeneti Z_{2-28} Zernike aberrációs együtthatók (a Z_1 együtthatót nem alkalmazzuk) alapján. Készítettünk továbbá egy *fókuszált szem paramétereit becslő* hálót, amely megadja a bemeneti távolságra fókuszáláshoz szükséges paramétermódosításokat. Ezenkívül létrehoztunk egy *aberrációbecslő* hálót is, amely a pontszórásfüggvények bemeneti paraméterei (d objektumtávolság, f fókuszált objektumtávolság, fény h horizontális és v vertikális beesési szöge, λ hullámhossz, A pupillaátmérő) és a ρ_{1-45} szemparaméterek alapján megadja a szem aberrációit. Végezetül a szemstruktúrabecslő háló tanításához szükségünk volt egy *diszkriminátor* hálóra, amely az optikai tengely pontjain határozza meg az aberrációs együtthatókat. Hálóink pontos bemeneti és kimeneti paramétereit az 1.1. táblázat foglalja össze.

Hálózat	**Bemenetek**	**Kimenetek**
Szemstruktúrabecslő	Z_{2-28}, A, λ	ρ_{1-45}
Fókuszált szem paramétereit becslő	ρ_{1-45}, A, f	Δ_{A_T}, Δ_{L_D}
Aberrációbecslő	ρ_{1-45}, A, λ, h, v	Z_{2-28}
Diszkriminátor	ρ_{1-45}, A, λ	Z_{2-28}

1.1. táblázat. Neurális hálózataink bemeneti és kimeneti paraméterei.

Habár bizonyos hálók feltételezhetően összevonhatók (például a fókuszált szem paramétereinek közvetlen becslésével), az általunk használt megközelítéssel képesek voltunk az egyes neurális hálók méretét csökkenteni. Ily módon a hálók tanítása és kiértékelése hatékonyabb, a futás közbeni memóriaigényük pedig alacsony. Ezenkívül a feladatok szeparációja a becsült szemparaméterek megtekintését is lehetővé teszi, amelyek hasznos információkat hordozhatnak a szimulált szemet illetően.

Tanítóadatok előállítása

Mivel hálózataink az egyedi, paraméteres szemmodellünkre épülnek, így hálóink tanításához saját adathalmazokat kellett létrehoznunk. A hálók eltérő tulajdonságai miatt minden hálóhoz egyedi adathalmazt készítettünk. A tanítás megkönnyítésére a szemmodellben a szaruhártya görbületi sugarait egy asztigmiaparaméterrel adtuk meg, a belső felület sugarát pedig a külső sugár arányaként definiáltuk. Az új paramétereket a Függelékben található A.2. táblázat foglalja össze.

Az aberrációkra épülő hálózataink adathalmazainak létrehozására először az A.2. táblázatban megadott populációs adatokkal egyenletes eloszlású véletlen mintavételezést hajtottunk végre a ρ_{1-45} szemparamétereken és a hálózatok többi bemeneti paraméterén (A, λ, h és v). Ezután a szemmodellünk aberrációszámító eljárásával kiszámoltuk a mintákhoz tartozó Z_{2-28} aberrációs együtthatókat. Az így kapott adatokból a végső adathalmazokat az oszlopok bemeneti és kimeneti halmazokhoz való hálózatfüggő hozzárendelésével állítottuk elő.

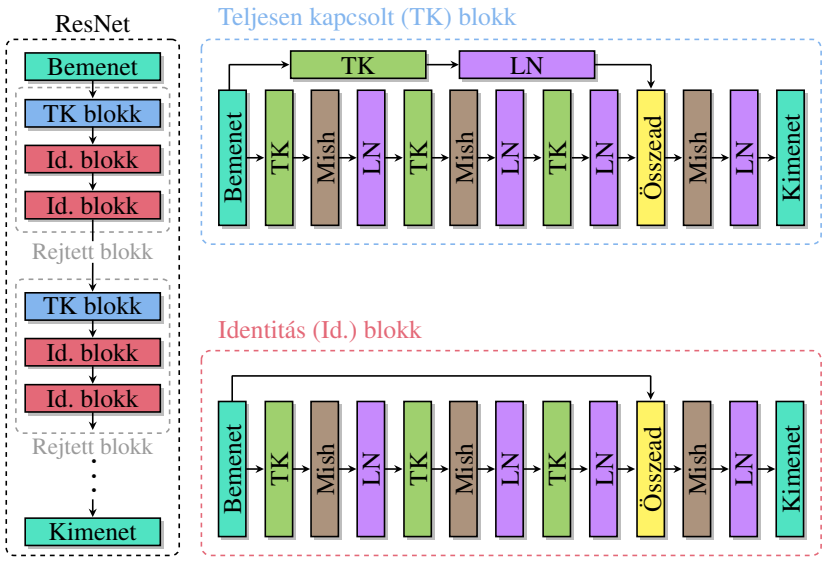
Hasonlóan készítettük el a fókuszált szem paramétereit becslő hálózatunk tanítóadatait is. Először létrehoztuk a mintapontok egy szemparaméterekből, pupillaátmérőből és fókuszált objektumtávolságból álló halmazát. Ezt követően a korábban ismertetett algoritmusunkkal meghatároztuk a mintapontokhoz a szemparaméterek azon módosításait (Δ_{L_D} szemlencseátmérő és Δ_{A_T} csarnokmélység), amelyek az egyes objektumtávolságokra való fókuszáláshoz szükségesek. Végezetül az így kapott adatok bemeneti és kimeneti oszlopokhoz rendelésével megkaptuk a tanítóadatok halmazát.

Fontos kiemelnem, hogy a fókuszált objektumtávolság mintavételezését külön végeztük a többi paraméterétől. Ez a megközelítés garantálja, hogy a mintaszemmodell és a mintatávolság összes egyedi kombinációja megjelenik az adathalmazban. Így tehát a szemparaméterekből és a pupillaátmérőből egyenletes eloszlással véletlenszerűen, a fókuszált objektumtávolságból pedig lineárisan vettünk mintát. Végezetül ezekből a mintákból a fókuszáló lépés bemenetét az adatok Descartes-szorzataként állítottuk elő.

Neurális hálózataink felépítése

A neurális háló struktúrája meghatározó tényezője a hálózat teljesítményének, és kulcsfontosságú szereppel bír a gyakorlati alkalmazáshoz szükséges pontosság elérésében. A terület gyors fejlődése miatt mára számtalan modell elérhető [89]. Mindezen architektúrák közül a szemmodellünk paramétereinek száma miatt a reziduális háló (ResNet) architektúrát vettük alapul [90]. A ResNet architektúra egyik fő, számunkra is kulcsfontosságú célja a mély hálók hatékony tanítása, aminek köszönhetően azt a gyakorlatban is rendszeresen alkalmazzák [89, 91]. A modell legfőbb elemeit a teljesen kapcsolt (TK) rétegek, a rétegek kimenetét transzformáló aktivációs függvény, a tanítás stabilitását és sebességét növelő normalizációs rétegek, és a blokkok kihagyását lehetővé tevő rövidítő útvonalak adják. Bár az eredeti ResNet architektúra konvolúciós hálókhoz készült, Chen

és mtsai. demonstráltak egy regresszióra alkalmazható változatot is [92]. A problémánk sajátosságai miatt további módosításokat végeztünk el a regressziós ResNet modellen, aminek eredményét az 1.8. ábra mutatja be.



1.8. ábra. Az általunk módosított regressziós ResNet architektúra.

Chen és mtsai. regressziós ResNet architektúrája a mély neurális hálók által is gyakran alkalmazott ReLU aktivációs függvényre [93] épült. A ReLU aktiváció azonban a negatív értékek eldobása miatt alacsony gradiensértékekhez vezethet, nagymértékben csökkentve a tanítás sebességét. A probléma elkerülésére a Misra által javasolt Mish aktivációs függvényt [94] alkalmaztuk:

$$f(x) = x\tanh(\ln(1+e^x)). \tag{1.14}$$

Annak érdekében, hogy hálóinkkal tetszőleges érték előállítható legyen, a kimeneti rétegekben lineáris aktivációt használtunk. Ettől eltérően azonban a szemstruktúrabecslő esetén a tanh kimeneti aktivációs függvényt alkalmaztuk, melynek korlátolt értékkészlete garantálja, hogy a becsült szemparaméterek a populációs adatokban megadott korlátok közé esnek.

A mély neurális hálók gyakori eleme a kötegnormalizációs (Batch Normalization, BN) réteg [95], amelyet Chen és mtsai. is alkalmaztak reziduális modelljükben. Gyakorlati kísérleteinkben azonban a rétegnormalizációs (Layer Normalization, LN) [96] réteg lényegesen jobban teljesített, ugyanis a kötegméretek

módosítása egyszerűbbé vált, hálóink pontossága (mely elemzése az 1.3.4. alszakaszban található ablációs vizsgálatban olvasható) pedig nagymértékben megnövekedett. Ezenkívül Chen és mtsai. architektúrájától eltérően a rétegnormalizációs réteget az aktivációs függvény mögé és a rövidítő útvonalak végére helyeztük el, ugyanis ez a konfiguráció szignifikánsan jobban teljesített kísérleteink során.

Neurális hálózatok tanítása

Hálóink tanítására a Rectified Adam (RAdam) optimalizációs eljárást [97] alkalmaztuk, amely a gyakran használt Adam algoritmus [98] egy továbbfejlesztett változata. Az RAdam algoritmus egyik fő erősségét a megnövekedett tanítási stabilitás jelenti, amelyet a tanítás elején kiugró gradiensek korlátozásával ér el. Bár az RAdam algoritmus konvergenciája csak korlátolt feltételek mellett garantált, a szakirodalomban elérhető eredmények alapján az eljárás kiemelkedő gyakorlati teljesítményt mutat [99, 100]. Ezenkívül a tanítás sebességének és stabilitásának növelésére előrenézést [101], regularizációra pedig súlycsökkentést [102] alkalmaztunk. Ezek az eszközök további segítséget jelentenek a lokális minimumhelyek elkerülésében is. Minden hálózatot 30 tanítási egységig tanítottunk 1024 méretű kötegekkel, a tanítás során pedig exponenciálisan csökkenő tanulási és súlycsökkenési sebességet használtunk a konvergencia ütemének növelésére.

Tanítás előtt minden adathalmazt felbontottunk egy tanító és egy teszt (validációs) halmazra. Bár erre a célra a szakirodalomban klasszikusan egy 80:20 arányú felbontást alkalmaznak [103], a közelmúltban demonstrált elméleti eredmények alapján a felbontás arányát célszerű a hálózat bemeneti paraméterei alapján meghatározni [104]. Ennek megfelelően a 85:15 arányú felbontást választottuk, amely a gyakorlati kísérleteink során konzisztensen jobb eredményekhez vezetett, mint a klasszikus 80:20 arányú felbontás. A bemeneti és a kimeneti halmazok oszlopait standardizáltuk az $\hat{x} = (x - \bar{x})/\sigma_x$ formulával, ahol x és $\hat{x}$ az eredeti és a standardizált oszlopokat, $\bar{x}$ és σ_x pedig x átlagértékét és szórását jelöli. Egyedüli kivétel a szemstruktúrabecslő, amely tanh aktivációs függvényének értékkészletéből fakadóan a kimeneti oszlopokat a $[-1, 1]$ tartományba transzformáltuk az $\hat{x} = -1 + 2(x - x_{min})/(x_{max} - x_{min})$ formulával.

A szakirodalomban elérhető eredmények alapján regressziós feladatok esetén az átlagos abszolút hiba (MAE) veszteségfüggvény kedvező tulajdonságokkal rendelkezik a gyakran alkalmazott átlagos négyzetes hiba (MSE) függvénnyel szemben [105]. Ezt a megfigyelést gyakorlati kísérleteink is alátámasztották,

ezért hálózataink tanítása során a MAE veszteségfüggvényt alkalmaztuk:

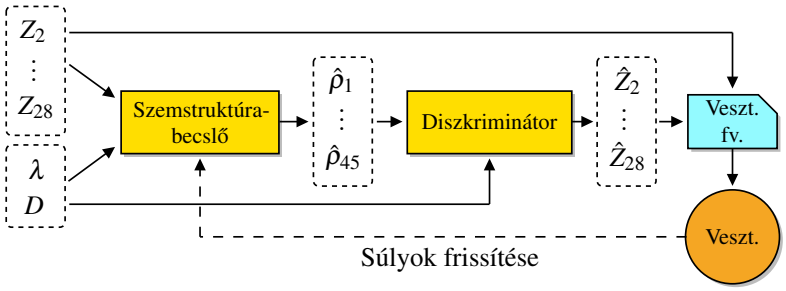
$$MAE(y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|, \tag{1.15}$$

ahol y_i és $\hat{y}_i$ a cél és a becsült kimeneti vektor i-edik oszlopát jelöli.

Diszkriminátoralapú szemstruktúrabecslő tanítás

Kísérleteink alapján a szemstruktúrabecslő fent ismertetett eszközökkel végzett tanítása valós alkalmazások számára nem megfelelő teljesítményt eredményez. Ennek oka, hogy a szemparaméterekből származtatott hiba nem alkalmas a háló pontosságának mérésére, mivel a szemparaméterek és a generált aberrációs együtthatók kapcsolata magasan nemlineáris. Továbbá egy aberrációs vektort számos eltérő szemparaméter-halmaz is generálhat, tovább nehezítve a tanítást.

A probléma megoldásához a diszkriminátor hálónkat alkalmaztuk. A szemstruktúrabecslő háló minden tanítási lépésében vesszük a bemeneti Z_{2-28} aberrációs vektorhoz becsült $\hat{\rho}_{1-45}$ szemparamétereket és a diszkriminátorral kiszámítjuk a $\hat{\rho}_{1-45}$ paraméterezésű szemmodell $\hat{Z}_{2-28}$ aberrációit. A tanítási lépés veszteségét a tanításhoz használt veszteségfüggvény Z és $\hat{Z}$ vektorral való kiértékelésével számítjuk ki. Ez a megközelítés sokkal precízebben méri a becsült szemparaméterek pontosságát, nagymértékben megnövelve a szemstruktúrabecslő hálónk teljesítményét. A folyamatot az 1.9. ábra szemlélteti.



1.9. ábra. A szemstruktúrabecslő hálózatunkhoz megalkotott diszkriminátoralapú tanítási módszerünk egy tanítási lépése.

Fontos kiemelnem, hogy a veszteségfüggvényhez szükséges, szemparaméterekhez tartozó aberrációs együtthatók sugárkövetéssel is meghatározhatók. Bár ez a megközelítés vélhetően növelné a betanított háló pontosságát, a sugárköve-

tés számítási ideje több nagyságrenddel magasabb a neurális hálókénál, ahogyan azt az 1.3.4. alszakaszban bemutatott méréseink demonstrálják. Mivel a veszteségfüggvényt minden tanítási lépésben ki kell értékelni a teljes adathalmazra, így a sugárkövetés költségét nem ítéltük alkalmasnak valós felhasználásra.

Továbbá azt is fontos megjegyeznem, hogy diszkriminátor hálókat a generatív ellenséges hálózatok (GAN) is gyakran alkalmaznak [106]. Ezen a területen a generáló és a diszkriminátor modellek tanítását jellemzően párhuzamosan végzik. A közelmúltban demonstrált új eredmények alapján azonban a diszkriminátor betanítása a generátor előtt számos esetben kedvezően hat a tanítás folyamatára [107, 108]. Ennélfogva mi is a diszkriminátor tanításával kezdjük, és csak azt követően végezzük el a szemstruktúrabecslő tanítását.

1.3.4. Elért eredmények

Tesztkörnyezet

Szemmodellünket, sugárkövetéses aberrációszámító és fókuszáló módszerünket, valamint mintakereséses paraméterkereső algoritmusunkat a MATLAB [109] szoftvercsomaggal és annak `patternsearch` optimalizációs eljárásával implementáltuk. A mintakeresés kiindulási pontjaként a populációs átlagértékeket választottuk ($\overline{x}_l$ oszlop a Függelékben található A.1. táblázatban), a konvergencia sebességének növelésére pedig a paramétereket a $[0,1]$ tartományba normalizáltuk és a 0,01-es kezdeti keresési környezetet alkalmaztuk.

Neurális hálóink tanítóadatainak előállítására, valamint a hálók létrehozására és tanítására a Python programozási nyelvet és a TensorFlow programkönyvtárat [110] használtuk. Az adatok előállítására a MATLAB segítségével implementált szemmodellünket és algoritmusainkat alkalmaztuk, a betanított hálókat a renderelő rendszerünkbe pedig a TensorFlow C++ API segítségével integráltuk. A bemenetek és a kimenetek standardizációjára saját normalizációs rétegeket hoztunk létre, így a normalizációs súlyok hálókba ágyazásával a standardizáció automatikusan végbemegy. Továbbá fontos kiemelnem, hogy a neurális hálók kiértékelését CPU-n hajtottuk végre. Ennek oka, hogy az elérhető videomemória túlnyomó részét a rendereléshez szükséges eszközök használták fel. Ezenkívül a hálókat csak ritkán, új szemállapotok szimulációjának elején szükséges kiértékelnünk, így a GPU-alapú gyorsítás jelentősége esetünkben elhanyagolható volt.

A szakirodalomban elérhető eljárásokat illetően nem végeztünk összehasonlítást létező módszerekkel. Ennek oka, hogy a szükséges eszközökhöz való hoz-

záférés nélkül nem volt módunk ezeket a módszereket végrehajtani. Ahogyan azt korábban kiemeltem, kutatásunk egyik fő célkitűzése (a személyreszabhatóság klinikai eszközök nélküli megvalósíthatósága) is ebből a korlátból eredt. Ezenkívül az értekezés ezen részében demonstrált méréseink kedvező, valós alkalmazások számára megfelelő eredményeket mutattak. Új megközelítésünk összehasonlítása már létező, több bemeneti adatot használó módszerekkel azonban egy fontos jövőbeli kutatási területet jelent.

A méréseink elvégzéséhez egy AMD Ryzen 7 1700 3.00 GHz processzort és egy NVIDIA TITAN Xp videokártyát alkalmaztunk.

Tanítóadatok

Az aberrációkra épülő hálóinkhoz 2 000 000 mintapontot tartalmazó adathalmazokat hoztunk létre, amely során az aberrációk meghatározásához 500×500 méretű sugárrácsokat alkalmaztunk. A szemstruktúrabecslő és a diszkriminátor adathalmazainak előállítása körülbelül öt napig tartott, az aberrációbecslő adathalmazát pedig hét napig generáltuk. A fókuszált szem paramétereit becslő hálózathoz a szemparaméterekből 25 000, a fókusztávolságból pedig 40 mintát vettünk, ami egy 1 000 000 elemű, hat napig generált mintahalmazt eredményezett.

Először megvizsgáltuk a tanítóadatok származtatott értékeihez tartozó leíró statisztikák főbb mutatóit, amelyeket az 1.2. táblázat foglal össze. A diszkriminátor és az aberrációbecslő hálók esetén az aberrációs együtthatókat három csoportra osztottuk azok fokszáma alapján, ugyanis elemzésünk során ezek az értékek hasonlóan viselkedtek. Ezenkívül a szemstruktúrabecslő adathalmazának értékeit nem tüntettem fel. Ennek oka, hogy mivel a szemstruktúrabecslő és a diszkriminátor adathalmazai csak a mintavételezéshez használt véletlenszám-generátor konfigurációjában térnek el, így azok tulajdonságai nagyon hasonlóak.

Az adathalmazok méretének csökkentésével végzett kísérleteink során a betanított hálózatok teljesítménye szignifikánsan romlott. Ennélfogva úgy ítéljük, hogy az adathalmazok méretének további növelése kedvező lehet módszerünk pontosságát illetően. Feltételezésünk gyakorlati ellenőrzésének fő akadályát az adatgeneráláshoz szükséges számítási idő jelentette. Éppen ezért az aberráció- és fókuszszámító eljárások futási idejének csökkentése ígéretes további kutatási területet jelenthet, hiszen ily módon nagyobb adathalmazok hatékony előállítása is lehetséges lenne.

	Diszkriminátor			Aberrációbecslő			Fókuszbecslő	
	$\boldsymbol{Z}_{2-6}$	$\boldsymbol{Z}_{7-15}$	$\boldsymbol{Z}_{16-28}$	$\boldsymbol{Z}_{2-6}$	$\boldsymbol{Z}_{7-15}$	$\boldsymbol{Z}_{16-28}$	$\boldsymbol{\Delta}_{L_D}$	$\boldsymbol{\Delta}_{A_T}$
$\min(x)$	−41,236	−4,601	−0,6095	−114,723	−42,732	−8,2681	−0,878	−0,572
$\max(x)$	26,862	4,744	0,6919	103,282	39,295	8,0194	0,000	0,000
$\overline{x}$	−0,572	−0,018	0,0003	−0,336	0,010	0,0023	−0,622	−0,198
$\overline{\lvert x \rvert}$	2,904	0,379	0,0200	4,231	0,701	0,0556	0,622	0,198
σ_x	4,138	0,568	0,0368	6,689	1,389	0,1192	0,331	0,125
Q_1	−2,373	−0,245	−0,0085	−2,696	−0,253	−0,0150	−0,859	−0,280
Q_2	−0,223	−0,007	0,0001	−0,106	0,001	0,0003	−0,839	−0,200
Q_3	1,626	0,212	0,0089	2,242	0,268	0,0187	−0,380	−0,125

1.2. táblázat. A generált adathalmazaink származtatott értékeihez tartozó leíró statisztikák főbb mutatói. A Zernike-együtthatók fokszáma rendre $n = 1, 2$ (Z_{2-6}), $n = 3, 4$ (Z_{7-15}) és $n = 5, 6$ (Z_{16-28}).

Ablációs vizsgálat

Az általunk javasolt neurális háló architektúra validációjához fontosnak ítéltük összehasonlítani annak teljesítményét a tradicionális előrecsatolt, illetve Chen és mtsai. eredeti regressziós ResNet modelljével. Ennélfogva elvégeztük hálózataink tanítását az alábbi konfigurációkkal:

- *FFN*: Egy tradicionális előrecsatolt háló (*feedforward network*), amelynek rejtett blokkjai egy teljesen kapcsolt és egy kötegnormalizációs réteget, illetve ReLU aktivációs függvényt tartalmaznak.
- *ResNet (Chen és mtsai.)*: A Chen és mtsai. által javasolt ResNet regressziós architektúra [92], ReLU aktivációs függvénnyel és kötegnormalizációval, pontosan követve az eredeti architektúrát.
- *ResNet (saját)*: Az 1.8. ábrán demonstrált módosított ResNet architektúránk, Mish aktivációkkal és rétegnormalizációs rétegekkel.

A hálók rejtett blokkjainak (h), a teljesen kapcsolt rétegek neuronjainak (n), valamint a hálókban összesen keletkező paramétereknek a számát (p) az 1.3. táblázat foglalja össze. Annak érdekében, hogy méréseink az architektúrából fakadó eltéréseket számszerűsítsék, a konfigurációk megválasztásakor törekedtünk a hálózatok teljes paraméterszámának egyezésére az egyes hálótípusok minden konfigurációja esetén.

A neurális hálók pontosságának számszerűsítésére bevett módszer a veszteségfüggvény kiértékelése a tanítás során nem látott adatokat tartalmazó validációs adathalmazra [103, 111]. A tesztelt konfigurációkhoz tartozó MAE értékek

Hálózat	Architektúra	h	n	p	MAE
Szemstruktúrabecslő	FFN	19	3,5 E	221 M	0,123248
	ResNet (Chen és mtsai.)	2	3,5 E	221 M	0,021555
	ResNet (saját)	2	3,5 E	221 M	0,009016
	(diszkriminátor nélkül)	2	3,5 E	221 M	0,075660
Diszkriminátor	FFN	19	2 E	72 M	0,070620
	ResNet (Chen és mtsai.)	2	2 E	72 M	0,013066
	ResNet (saját)	2	2 E	72 M	0,003137
Fókuszált szem paramétereit becslő	FFN	19	2 E	72 M	0,003099
	ResNet (Chen és mtsai.)	2	2 E	72 M	0,001538
	ResNet (saját)	2	2 E	72 M	0,000429
Aberrációbecslő	FFN	19	2 E	72 M	0,087222
	ResNet (Chen és mtsai.)	2	2 E	72 M	0,037234
	ResNet (saját)	2	2 E	72 M	0,015308

1.3. táblázat. A vizsgált hálók tulajdonságai és a betanított hálók pontosságát mérő, validációs halmazokon mért átlagos abszolút hibák (*MAE*). Jól látható, hogy a módosított ResNet architektúránk a többi modellhez képest szignifikánsan alacsonyabb hibákat eredményezett minden vizsgált esetben.

az 1.3. táblázatban olvashatók. Jól látható, hogy az általunk javasolt architektúra a legpontosabb minden vizsgált esetben. Bár Chen és mtsai. architektúrája konzisztensen pontosabb az előrecsatolt hálónál, módosított konfigurációnk további számottevő növekedést biztosított. A hibák mértékét az 1.2. táblázatban feltüntetett leíró statisztikákkal összehasonlítva az is elmondható, hogy a javasolt architektúránkkal épített hálók megfelelőek a szemstruktúra- és az aberrációszámítás pontos elvégzésére, ami nem teljesül minden konfigurációra. Mindezek tükrében úgy találtuk, hogy módosított architektúránk kulcsfontosságú a megfelelő pontosság eléréséhez.

Méréseink alapján a perifériku aberrációk becslése bizonyult a legnehezebb feladatnak. A célértékek 1.2. táblázatban bemutatott leíró statisztikái alapján azonban megfigyelhető, hogy az aberrációs együtthatók tipikusan lényegesen magasabbak a perifériku régióban. Ezzel a megfigyeléssel konzisztens a diszkriminátor és az aberrációbecslő hálózatok hibáinak nagysága is. Mindezek tükrében tehát úgy véljük, hogy a megnövekedett hibaérték fő oka a célértékek nagyságából fakad, az aberrációbecslő relatív pontossága pedig hasonló a többi hálózatéhoz.

Végezetül kiértékeltük a szemstruktúrabecslő tanítására létrehozott, diszkriminátoralapú megközelítésünket is. Erre a célra elvégeztük a hálózat tanítását a naiv, szemparaméterek hibáját alkalmazó célfüggvénnyel, illetve az általunk javasolt, 1.9. ábrán bemutatott eljárással. A két háló pontosságának összehasonlítására alkalmaztuk a diszkriminátort a naiv módon tanított hálózat kimeneteire, aminek eredménye szintén az 1.3. táblázatban olvasható. Ily módon a táblázatban található validációs hibák a megbecsült szemparaméterekhez tartozó aberrációs együtthatók hibáját mérik minden szemstruktúrabecslő esetén. Eredményeinkből egyértelműen látható, hogy a naiv megközelítés átlagos hibája (0,076) közel egy nagyságrenddel nagyobb az általunk javasolt megközelítéssel tanított hálózaténál (0,009). Így tehát úgy ítéltük, hogy a diszkriminátoros tanítási módszerünk esszenciális a szemstruktúrabecslő megfelelő pontosságának eléréséhez.

Vizsgált szemállapotok

Kísérleteink során megvizsgáltunk hat esetet: az emmetrópiát (egészséges látás), a miópiát (rövidlátás), az asztigmiát, a keratokónuszt, a kataraktát (szürke hályog), illetve egy, a LASIK (lézeres) szemműtétet követő állapotot. Az első három eset modellezésére konverziós formulákat [47] alkalmaztunk, amelyekkel a szem aberrációit a korrekciós szemüveg paramétereiből számítottuk ki. A többi szemállapot szimulációjára valós mérési adatokat használtunk.

Az aberrációbecslés vizsgálatára két konfigurációt alkalmaztunk. Az első esetben az objektumpontok mindegyike az optikai tengelyen helyezkedett el (*tengelyen*). Ehhez hat pupillaátmérőt és hét fókuszált objektumtávolságot használtunk a pontszórásfüggvények paramétereinek mintavételezésére. A második esetben az objektumpontok az optikai tengelyen kívülre is estek (*perifférikus*). Ebben az esetben egy 5 mm-es pupillaátmérőjű relaxált szemet alkalmaztunk 19 horizontális és 11 vertikális beesési szöggel. Mindkét konfigurációban három hullámhosszt használtunk.

Pontosság

A neurális hálóink ablációs vizsgálata során mért pontosságok a hálók nagyméretű adathalmazokon vett átlagos teljesítményét számszerűsítették. Következő lépésként szerettük volna a mintakeresésen és a neurális hálókon alapuló módszereink teljesítményét valós esetekre is kiértékelni. Vizsgálatunk elsődleges célja az elvárt kimenetekhez viszonyított relatív pontosságok számszerűsítése volt. Ehhez

a korábban ismertetett hat szemállapot és két aberrációs konfiguráció felhasználásával kiértékeltük a kimenetek átlagos abszolút célértékét és új eljárásaink átlagos abszolút hibáját (MAE). A MAE függvény alkalmazásával a kapott értékek a neurális hálóink 1.3. táblázatban bemutatott pontosságával is összehasonlíthatók. Eredményeinket az 1.4. táblázat foglalja össze.

Érték és módszer		**Emmet.**	**Mióp.**	**Asztig.**	**Kerat.**	**Kata.**	**LASIK**
Szemstruktúra	Cél	0,0000	0,2005	0,3440	0,2894	0,1655	0,0583
• (mintakeresés)	MAE	0,0008	0,0010	0,0011	0,0076	0,0361	0,0029
• (neurális hálózat)	MAE	0,0044	0,0047	0,0038	0,0083	0,0048	0,0024
Fókuszált szem paraméterei	Cél	0,3171	0,0411	0,0122	0,2000	0,2130	0,4124
• (neurális hálózat)	MAE	0,0111	0,0058	0,0020	0,0046	0,0174	0,0033
Aberrációk (tengelyen)	Cél	0,1484	0,2137	0,3376	0,3167	0,4254	0,2506
• (neurális hálózat)	MAE	0,0079	0,0093	0,0093	0,0107	0,0143	0,0086
Aberrációk (periférikus)	Cél	0,7013	0,7009	0,7031	0,7872	0,4156	0,9545
• (neurális hálózat)	MAE	0,0152	0,0152	0,0149	0,0182	0,0208	0,0197

1.4. táblázat. Elvárt kimenetek átlagos abszolút értéke (*cél*) és átlagos abszolút hibák (*MAE*) a mintakeresésen alapuló szemstruktúrabecslő módszerünk, illetve a szemstruktúrát és az aberrációkat becslő neurális hálóink esetén. Látható, hogy módszereink hibája konzisztensen több nagyságrenddel kisebb a célértékeknél.

Jól látható, hogy módszereink rendkívül alacsony relatív hibát eredményeznek ezekben a valószerű tesztesetekben is. Mindkét szemstruktúrabecslő módszerünk esetén elmondható, hogy az előállított szemmodellek képesek a bemeneti szem aberrációinak pontos reprodukciójára, így felhasználhatók a szem látásának szimulációjához. Habár neurális hálóink jellemzően magasabb hibát generálnak, az eltérések a célértékekhez képest továbbra is alacsonyak.

A fókuszált szemparaméterek és aberrációk becslésének esetén a sugárkövetéses módszereink jelentik az összehasonlítások alapját, így csak a neurális hálós módszerünkkel generált értékek hibáját tüntettem fel. Eredményeinkből jól látható, hogy neurális hálóink a költséges sugárkövetés eredményét magas pontossággal közelítik minden tesztkonfigurációban. A legnehezebb feladatnak a fókuszált szemparaméterek becslése bizonyult, aminek oka vélhetően a hálózat célértékeinek eloszlásában keresendő. Ugyanakkor a megnövekedett hiba ellenére is úgy ítéljük, hogy a célértékekhez viszonyított eltérések kellően alacsonyak a neurális hálókra épülő módszereink gyakorlati alkalmazásához.

Sebesség

Módszereink sebességének kiértékelésére megmértük a relaxált és a fókuszált szemstruktúra becsléséhez, illetve az így kapott szemmodellek aberrációinak kiszámításához szükséges futási időket. Méréseink eredményét az 1.5. táblázat foglalja össze.

Feladat	Módszer	Emmet.	Mióp.	Asztig.	Kerat.	Kata.	LASIK
Szemstruktúra becslése	Mintakeresés	3 521 s	6 270 s	7 306 s	10 642 s	4 582 s	4 696 s
	Neurális háló	0,079 s	0,088 s	0,064 s	0,061 s	0,073 s	0,085 s
Fókuszált szem paramétereinek becslése	Sugárkövetés	77,53 s	46,17 s	56,19 s	35,62 s	59,33 s	32,97 s
	Neurális háló	0,046 s	0,056 s	0,063 s	0,051 s	0,065 s	0,059 s
Aberrációk becslése (tengelyen)	Sugárkövetés	25,51 s	24,10 s	21,06 s	30,61 s	28,44 s	31,72 s
	Neurális háló	0,170 s	0,134 s	0,129 s	0,159 s	0,146 s	0,175 s
Aberrációk becslése (periférikus)	Sugárkövetés	213,4 s	250,4 s	208,4 s	207,8 s	201,7 s	194,8 s
	Neurális háló	0,638 s	0,512 s	0,548 s	0,643 s	0,583 s	0,617 s

1.5. táblázat. Saját eljárásaink futási ideje. Sugárkövetéses módszereink esetén a számítási idő zömét a szemstruktúra becslése képezi. Ezzel szemben a neurális hálókat alkalmazó megközelítésünk együttes futási ideje is egy másodperc alatt marad.

Jól látható, hogy a sugárkövetésen és mintakeresésen alapuló szemrekonstrukciós megközelítésünk számítási ideje nagyban függ a bemeneti adatok felépítésétől. A csupán alacsonyrendű aberrációkat tartalmazó szemállapotok esetén az optimalizáció konvergál egy (emmetrópia) vagy két (miópia, asztigmia) óra után, míg a tetszőleges aberrációkat tartalmazó keratokónusz rekonstrukciója három órát is igénybe vesz. Fontos megjegyeznem, hogy a mintakereső algoritmus leállási feltételeként a vizsgálandó környezet méretét használtuk, amihez a pontos becslés érdekében a 0,0005-ös alsó korlátot alkalmaztuk. Kísérleteink alapján azonban az optimalizációs algoritmus a futási idő töredéke alatt közel jut a végeredményhez. Ennélfogva a megállási küszöbérték növelésével a folyamat hossza számottevően csökkenthető a rekonstrukció minőségének szignifikáns romlása nélkül. Végezetül a szemrekonstrukcióhoz képest a fókuszáló és az aberrációszámító folyamat lényegesen gyorsabb, amelyek együtt is néhány perc alatt elvégezhetők minden tesztesetben. Összességében úgy ítéltük, hogy módszereink sebessége megfelelő valós alkalmazások számára.

Neurális hálóink számára a hálózatok nagyszámú kiértékeléséből fakadóan

(*tengelyen*: 126, *perifériku*s: 627 kiértékelés) az aberrációbecslés bizonyult a legköltségesebb feladatnak. Eredményeink alapján azonban egyértelműen látható, hogy a neurális hálókat alkalmazó módszereink kombinált futási ideje egy másodperc alatt maradt minden tesztesetben. Elmondható tehát, hogy sikeresen megvalósítottuk a teljesen interaktív testreszabhatóságot. A sugárkövetéses és a mintakereséses módszerekhez viszonyítva a neurális hálókon alapuló megközelítésünk legalább három nagyságrenddel gyorsabb volt minden fázisban. A legnagyobb előrelépést a szemstruktúrabecslésben tapasztaltuk, amit a mintakeresés által nagy számban elvégzett erőforrásigényes aberrációszámítás okoz. Neurális hálónk esetén ez a költség az adatgenerálásban és a tanításban jelenik meg, a látásszimuláció során viszont a feladat a háló egyetlen kiértékelésével elvégezhető.

1.4. Pontszórásfüggvények előállítása

Ahogyan azt a pontszórásfüggvény ismertetésekor az 1.1.3. alszakaszban kiemeltem, a pontszórásfüggvény nagyban függ olyan tényezőktől, mint az objektum- és a fókusztávolság, a pupillaátmérő, illetve a fény hullámhossza és beesési iránya. A paraméterek számából fakadóan az előállítandó pontszórásfüggvények mennyisége számos felhasználási esetben robbanásszerűen megnő, amelyek kiszámításának ideje számottevően ronthatja a rendszer felhasználhatóságát.

Az 1.1.3. alszakaszban ismertetett ENZ módszer nem képes az emberi szem nagy mennyiségű pontszórásfüggvényét interaktív módon kiszámítani. Ennek oka, hogy a számítások újrafelhasználása és GPU-alapú párhuzamosítása nem triviális feladat. A probléma megoldására kidolgoztunk egy eljárást, amellyel a pontszórásfüggvények nagy elemszámú halmazai hatékonyan kiszámíthatók. Ehhez az ENZ módszer GPU-alapú végrehajtására és a részeredmények újrafelhasználására támaszkodtunk. Algoritmusunk részleteit a 2022-ben publikált cikkünk [112] alapján ismertetem.

1.4.1. Saját, GPU-alapú módszerünk

Eljárásunk főbb lépéseit az 1. algoritmus foglalja össze. Módszerünk feltételezi, hogy a pontszórásfüggvények paraméterei rendelkezésre állnak, és egyedül a pontszórásfüggvények előállítása a cél. Először egy előfeldolgozó lépésben a V_n^m függvényt független komponensekre bontjuk (1–6 sor), amelyekből a pontszórásfüggvények kimeneti halmazát előállítjuk a grafikus kártyán (7–16 sor).

1. algoritmus. A saját, GPU-alapú módszerünk az emberi szem nagyszámú pontszórásfüggvényt tartalmazó halmazainak előállítására.

Bemenet : Defókuszparaméterek $\mathscr{F}$, Zernike-együtthatók β,
mintavételezési paraméterek $\overline{K}$, N_r, μ_r, N_s,
előre kiszámított V_n^m részkomponensek j, J, w.
Kimenet : Pontszórásfüggvények halmaza $\mathscr{U}$.

```
// Komponensek kiszámítása.
for k ← 0 to K̄ do
    parallel foreach f ∈ F do [on CPU]
        ĵ_k(f) ← Compute_jk(k, f, j)
    R ← Create_RadiusGrid(N_r, μ_r)
    parallel foreach r ∈ R, n, m do [on GPU]
        ĵ_k^nm(r) ← Compute_Jk(k, n, m, r, J, w)
// Pontszórásfüggvények kiszámítása.
for u ← 0 to |U| − 1 do
    K, μ_s ← Calculate_Sampling(u, F)
    R, Φ ← Create_PupilGrid(N_s, μ_s)
    parallel foreach r ∈ R, n, m do [on GPU]
        V_n^m(r) ← Calculate_Vnm(u, r, K, ĵ_k^nm, ĵ_k)
    U ← Initialize_ZeroGrid(N_s, N_s)
    parallel foreach r ∈ R, φ ∈ Φ, n, m do [on GPU]
        V(r) ← Interpolate_Vnm(V_n^m, r)
        U(r, φ) ← Accumulate_U(U, V, r, φ, β_n^m)
    U(u) ← ReadBack_Result_From_GPU(U)
```

Független V_n^m komponensek meghatározása

Az 1.1.3. alszakaszban definiált (1.8) és (1.10) egyenletet közelebbről megvizsgálva megállapítható, hogy a pontszórásfüggvények előállításához szükséges számítások legnagyobb részét a V_n^m függvény kiértékelése adja. Célunk tehát a V_n^m függvény kiértékelését hatékonyabbá tenni. Ehhez először az (1.10) egyenlettel definiált V_n^m függvényt a következő két független komponensre bontottuk fel:

$$\hat{j}_k(f) = (2k+1)\exp\left(\frac{if}{2}\right) i^k j_k\left(\frac{f}{2}\right), \tag{1.16}$$

$$\hat{J}_k^{nm}(r) = \sum_{l=\max(0,k-q,p-k)}^{k+p} (-1)^l w_{kl} \frac{J_{m+2l+1}(2\pi r)}{2\pi r}, \tag{1.17}$$

ahol $k \in [0, K]$. Ezekből a komponensekből V_n^m a következő módon áll elő:

$$V_n^m(r, f) = \sum_{k=0}^{K} \hat{j}_k(f) \hat{J}_k^{nm}(r). \tag{1.18}$$

Az általunk javasolt felbontással a V_n^m függvény két paramétere egymástól teljesen függetlenül kezelhető. Ezek közül egyedül f függ a pontszórásfüggvények paramétereitől, hiszen r a pontszórásfüggvény képpontjainak koordinátáiból származtatható. Ennek megfelelően r sűrű mintavételezésével a $\hat{J}_k^{nm}$ komponens értékei előre kiszámíthatók és újrafelhasználhatók minden pontszórásfüggvény előállításához. Ehhez a mintavételezéshez definiáltunk egy konfigurálható μ_r mintaméretet és N_r darabszámot, amelyekkel az r paraméter terét egyenlő méretű tartományokra osztottuk fel ($r \in [0, (N_r - 1)\mu_r]$). Ezenkívül a k paraméter értékét is korlátoztuk egy konfigurálható paraméterrel ($\overline{K}$).

A fent leírtak alapján a V_n^m komponensek előállításához szükséges előfeldolgozási szakaszunk a következő lépésekből áll:

1. j_k, majd $\hat{j}_k$ kiszámítása minden f paraméterhez.
2. w_{kl} kiszámítása n, m, k és l minden kombinációjához.
3. J_{m+2l+1} kiszámítása m, l és r minden kombinációjához.
4. $\hat{J}_k^{nm}$ kiszámítása n, m, k és r minden kombinációjához.

Az 1. és a 3. lépést CPU-n hajtjuk végre. Ennek oka, hogy a Bessel-függvények rekurzivitását [113] felhasználva azok hatékonyan kiértékelhetők CPU-alapú párhuzamosítással. Ezenkívül a számítások átültetése masszívan párhuzamos környezetbe nem triviális, és kész, GPU-alapú eljárások sem érhetők el. Továbbá a w_{kl}-ben fellelhető faktoriálisokat gamma-függvényekkel számítjuk ki. A gamma-függvények GPU-alapú kiértékelése szintén nem triviális, CPU-n viszont hatékonyan elvégezhető az l paraméter végességének köszönhetően. Mindezek alapján az 1–3. lépés CPU-n történő végrehajtása után feltöltjük az eredményeket a videomemóriába és elvégezzük az utolsó, fő lépést a GPU-n.

Ahogyan azt korábban kiemeltem, az (1.16) és az (1.17) egyenletet megvizsgálva belátható, hogy a V_n^m függvény kiszámításához szükséges idő legnagyobb részét a $\hat{J}_k^{nm}$ komponens elemeinek kiértékelése képezi. Mivel ezek az értékek egyedül az r paramétertől függnek, így a 2–4. lépést elég pusztán egyszer, a program inicializációjakor elvégezni. Ily módon a pontszórásfüggvényekből képzett halmaz előállításához szükséges idő jelentősen lerövidíthető.

Pontszórásfüggvények előállítása

A V_n^m részkomponensek kiszámítása után a kiértékeléshez szükséges adatok készen elérhetők GPU pufferekben, így tehát elvégezhetjük a pontszórásfüggvények előállítását. A halmaz elemeit egyesével állítjuk elő a GPU-n, majd azokat a rendszermemóriába visszaolvasva tároljuk el. Bár az elemek párhuzamos kiszámítása is egy lehetséges megközelítés, kísérleteink során a szekvenciális feldolgozás nem okozott számottevő teljesítménybeli csökkenést. Módszerünkkel azonban a szükséges GPU pufferek mérete drasztikusan lecsökkent, így pedig tetszőleges elemszámú halmaz problémamentes előállítása lehetővé vált.

Első lépésként definiálunk egy egyenlő felosztású rácsot. Minden pontszórásfüggvény esetén ugyanannyi mintát veszünk az x és az y tengely mentén (N_s), ám a rácspontok közötti távolságot (μ_s) és a k változó felső korlátját (K) az f paraméter alapján pontszórásfüggvényenként választjuk meg. Ezt követően a mintavételezési paraméterek ismeretében kiértékeljük a rácspontokon a V_n^m függvényt az aktuálisan kiszámított pontszórásfüggvény paramétereivel. Mivel a szükséges $\hat{j}_k$ és $\hat{J}_k^{nm}$ értékek rendelkezésre állnak a videomemóriában, így a feladat leegyszerűsödik egy k paraméter fölötti összegzésre, amelyet az n és az m fokszám minden egyedi kombinációjára szükséges elvégezni. Fontos további tényező, hogy a K érték és a mintavételezésből fakadó legnagyobb r paraméter pontszórásfüggvényenként eltér. Ennek következtében az előre kiszámított komponenseknek túlnyomórészt csak egy lényegesen kisebb részhalmaza szükséges V_n^m kiszámításához, tovább csökkentve az elvégzendő munka mennyiségét.

Végezetül a pontszórásfüggvény mintáit az (1.8) egyenletnek megfelelően a V_n^m függvényértékek β_n^m együtthatókkal súlyozott összegeként számítjuk ki. Ehhez fontos figyelembe vennünk, hogy a pontszórásfüggvényhez az f alapján kiválasztott μ_s rácsméret eltérhet az előszámítási μ_r mintaérettől. A probléma orvoslására az adott pontszórásfüggvényhez előállított V_n^m értékeket egy GPU textúrában tároljuk. Ily módon az előszámított rácspontok interpolációja egy egyszerű, GPU-val gyorsított textúramintavételezési művelettel elvégezhető.

1.4.2. Elért eredmények

Tesztkörnyezet

Az új módszerünkkel elért teljesítménynövekedés mérésére elkészítettük annak referenciaimplementációját C++ programozási nyelven. Ezenkívül implementál-

tunk egy CPU-alapú referenciametódust, amely CPU szálakat és a GNU Scientific Library programkönyvtárat [114] alkalmazza a Bessel-függvények kiértékelésére és a V_n^m minták interpolációjára. A GPU-alapú módszerünket az OpenGL grafikus programkönyvtár [115] által biztosított GLSL árnyalókkal implementáltuk, ugyanis a befoglaló keretrendszerben a szükséges eszközök készen rendelkezésre álltak. Mintavételezésre az $N_r = 4000$, $\mu_r = 1/8$ és $\overline{K} = 800$ értékeket használtuk a V_n^m részkomponensek előállítására, illetve $N_s = 301 \times 301$ mintát alkalmaztunk a pontszórásfüggvény kiértékeléséhez használt rácsban. Végezetül a β együtthatók legnagyobb radiális fokszámaként az $N_\beta = 30$ értéket választottuk, összesen 496 komplex együtthatót eredményezve.

A valós felhasználási módokkal való elemzés céljából két konvolúciós látásszimuláló megközelítés pontszórásfüggvényeinek előállítási idejét mértük meg. Mindkét módszerrel ugyanannak az erősen rövidlátó szemnek a relaxált állapotú, 5 mm-es pupillaátmérőjű látását szimuláltuk. A mérésekhez egy AMD Ryzen 7 1700 3.00 GHz CPU-t és egy NVIDIA TITAN Xp GPU-t alkalmaztunk.

Az első vizsgált módszer ritka halmazokat alkalmaz a szimulációhoz. Ilyen elven működik például Barsky korábban tárgyalt algoritmusa [60] és a dolgozat későbbi részeiben bemutatott módszereink is. Tesztünkhöz 41 objektumtávolságot és három hullámhosszt alkalmaztunk, ami 123 pontszórásfüggvényt eredményezett. A második látásszimulációs megközelítés pedig a képpontok objektumtávolság-szerinti osztályozásával állítja elő a kiszámítandó pontszórásfüggvények sűrűn mintavételezett halmazát. A vizsgált bemeneten a folyamat 5 924 osztályt azonosított, amely három hullámhossz alkalmazásával 17 772 pontszórásfüggvényt eredményezett. Az egyes objektumtávolságok pontszórásfüggvényeit külön-külön számítottuk ki, mivel a halmaz mérete nem tette lehetővé annak egyidejű tárolását a rendszermemóriában.

Teljesítmény

A sebességnövekedés kiértékelésére megmértük minden tesztesetre a V_n^m minták kiszámításához és interpolációjához szükséges időt. Ezenkívül azt is megvizsgáltuk, hogy a $\hat{J}_k^{nm}$ komponens általunk javasolt előszámítása hogyan befolyásolja a halmazok előállítási idejét. Eredményeinket az 1.6. táblázat foglalja össze.

Méréseink egyértelműen demonstrálják, hogy a GPU-alapú eljárásunk több nagyságrenddel gyorsabb a CPU-alapú megközelítésnél. Módszerünk képes a másodperc töredéke alatt előállítani a kis elemszámú halmazokat. Ez az áteresztő-

	ELŐSZÁMÍTÁS NÉLKÜL				ELŐSZÁMÍTÁSSAL			
	RITKA RÁCS		SŰRŰ RÁCS		RITKA RÁCS		SŰRŰ RÁCS	
	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU
$V_n^m(r,f)$	97 s	4,02 s	25 618 s	1402 s	66 s	0,30 s	15 238 s	109 s
$U(r,\phi,f)$	56 s	0,25 s	15 690 s	970 s	62 s	0,17 s	16 158 s	102 s
Teljes idő	153 s	4,27 s	41 308 s	2372 s	128 s	0,47 s	31 396 s	211 s

1.6. táblázat. Futási idők a vizsgált tesztesekben. Jól látható, hogy GPU-alapú módszerünk több nagyságrenddel gyorsabb a naiv, CPU-alapú megközelítésnél. Ezenkívül az általunk javasolt előszámítás hangsúlyos szerepe is megfigyelhető, amely elsősorban a GPU-alapú eljárásunk esetén okozott számottevő gyorsulást.

képesség kiemelten fontos a saját látásszimuláló módszereink számára, hiszen az elengedhetetlen az emberi szem különböző állapotainak interaktív vizsgálatához. Ezenkívül új eljárásunk a nagyméretű, csak több ciklusban előállítható halmazok kiszámításában is jelentős gyorsulást eredményez.

Az előszámítás nélkül és az előszámítással készített méréseinket összehasonlítva az is egyértelműen látható, hogy az elért teljesítménynövekedés nem csak a számítás masszívan párhuzamos környezetbe való átültetéséből ered. Mindezt csak a V_n^m függvény általunk javasolt felbontásával és előszámításával tudtuk elérni. Továbbá az is jól látható, hogy az előszámítás csak a GPU-alapú módszerünk esetén okoz számottevő gyorsulást. Ennek oka, hogy a CPU-alapú végrehajtás során a legfőbb korlátot a rendszermemória elérési ideje jelenti.

Memóriaigény

Mivel új módszerünk előszámításra épül, így fontos megvizsgálnunk algoritmusunk memóriaigényét is. Az eljárásunk által használt memória mennyiségének legmeghatározóbb tényezőjét az előszámítási paraméterek jelentik. A méréseinkhez használt tesztkonfigurációval a w_{kl}, a J_{m+2l+1}, a $\hat{J}_k^{nm}$ és a V_n^m komponens memóriaigénye rendre 96,8 MB, 24,9 MB, 3,1 GB és 7,8 MB. Jól látható, hogy a memóriaigény túlnyomó részét a $\hat{J}_k^{nm}$ komponens képezi, ami az (n,m) kombinációk nagy számából és az r paraméter sűrű mintavételezéséből ered. Módszerünk memóriaigénye azonban teljesen megfelelő a jelenleg elérhető GPU-k számára.

Fontos megjegyeznem, hogy az interpolációs műtermékek elkerüléséhez rendkívül magas felbontású, így szükségtelenül nagy memóriaigényű mintavételezést használtunk. Ahogyan azt azonban a dolgozat ezt követő részeiben ismerte-

tem, a látásszimulációs módszereink a pontszórásfüggvények képtérre vetítésére épülnek, nagyban redukálva az interpolációs hibák jelentőségét. Ilyen esetekben N_r csökkentésével és μ_r arányos növelésével az előszámított értékek memóriaigénye lényegesen redukálható számottevő hibák nélkül.

Végezetül azt is fontos kiemelnem, hogy szükség esetén az előszámított adatok a rendszermemóriában is tárolhatóak. Ebben az esetben az adatokat csak a halmazok számításának idejére szükséges a rendszermemóriából a videomemóriába másolni. Az általunk vizsgált tesztesetek ugyanakkor nem igényelték az előszámított értékek ily módon való mozgatását.

1.5. Látás szimulációja valósidejű rendszerekben

Értekezésem ezen részében ismertetem az emberi látást szimuláló képek előállítására megalkotott eljárásainkat. Ehhez a dolgozat korábbi részeiben bemutatott, a fizikai szemstruktúra becslését és a szem pontszórásfüggvényeinek hatékony előállítását lehetővé tevő eszközökre támaszkodtunk. Először létrehoztunk egy komplex fázorokat alkalmazó módszert az alacsonyrendű aberrációkat tartalmazó látás hatékony szimulációjára. Ez az algoritmus a sebességet helyezi előtérbe és az optikai tengelyen elhelyezkedő pontok távolságfüggő pontszórásfüggvényeinek szeparábilis approximációján alapszik. Ezután megalkottunk egy, az egzakt pontszórásfüggvényeket alkalmazó módszert is, amely képes tetszőleges aberráció és a periférikus látás valósidejű szimulációjára.

1.5.1. Alacsonyrendű aberrációk szimulációja fázorokkal

Ahogyan azt az 1.1.3. alszakaszban kiemeltem, a pontszórásfüggvény erősen függ olyan térben változó paraméterektől, mint a beesési szög és az objektumtávolság. Éppen ezért a háromdimenziós jelenetek konvolúciója a szem pontszórásfüggvényével nem lehetséges frekvenciatérben, ami egy jelentős gyorsítási lehetőség kizárását eredményezi. Ezenkívül a pontszórásfüggvények méretéből fakadóan a konvolúció közvetlen kiértékelése rendkívül számításigényes.

A probléma kiküszöbölésére a kamerarendszerek mélységélességét szimuláló módszerek tipikusan úgynevezett *szeparábilis* magfüggvényeket használnak. A módszer lényege, hogy az eredeti magfüggvénnyel végzett konvolúció eredménye alacsony hibával közelíthető egydimenziós magfüggvények egymást követő alkalmazásával. Kétdimenziós magfüggvények esetén ez jellemzően egy hori-

zontális és egy vertikális komponens alkalmazását jelenti, amellyel a konvolúció idejének skálázódása négyzetesről lineárisra redukálható. Mindez már közepes méretű magfüggvények esetén is nagyban csökkenti a szimuláció idejét.

A szeparábilis magfüggvény megválasztásának több módja is létezik. A Gauss-függvényt a számítógépes grafikában már régóta alkalmazzák a mélységélesség reprodukciójára [116, 117] és az emberi látás szimulációjára [70]. A kamerarendszerek rekesznyílása által generált úgynevezett *bokeh* mintázat pontosabb közelítésére geometriaalapú szűrők [118], szinguláris érték felbontás [119], valamint optimalizációval meghatározott ritka magfüggvények [120] is megtalálhatók a szakirodalomban. Ezenkívül a kör alakú mintázatok esetén komplex fázorok is alkalmazhatók [121, 122].

Munkánk célja egy olyan algoritmus létrehozása volt, amellyel az emberi szem alacsonyrendű aberrációi gyorsan, a létező módszerekre jellemző hosszadalmas előfeldolgozási lépés nélkül szimulálhatók. Mivel az alacsonyrendű aberrációk jellemzően kör- és ellipszisszerű diffrakciós mintákhoz vezetnek [14, 15, 31, 70, 71], így a konvolúciós magfüggvényeket komplex fázorokkal közelítjük. Módszerünket a 2018-as publikációnk [123] alapján mutatom be.

Előfeldolgozási lépések

Minden szimulálandó szemállapot esetén végrehajtunk egy előfeldolgozási fázist az első renderelés előtt. Ennek során előállítjuk a pontszórásfüggvények egy mélységfüggő rácsát, amelyből a renderelési szakaszban a háromdimenziós jelenet képpontonkénti elmosási sugarát határozzuk meg. Ezután ellipszisillesztést hajtunk végre, melynek kimenete egy elforgatási és egy zsugorodási érték lesz. Ezeket a paramétereket az asztigmiás pontszórásfüggvények approximációjához alkalmazzuk. Ezt követően előállítjuk a magfüggvény mintákhoz tartozó súlyait. Végezetül az előfeldolgozás során meghatározott összes adatot feltöltjük a videomemóriába a renderelési szakaszban való gyors hozzáférés érdekében.

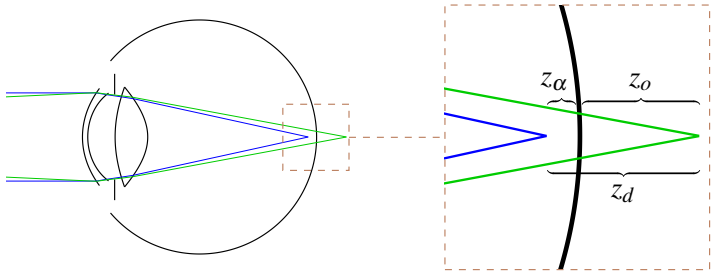
Pontszórásfüggvények kiszámítása

Módszerünk bemenetét a szimulált szem korrekciós szemüvegének paraméterei képezik. A pontszórásfüggvényekhez szükséges aberrációs együtthatók meghatározását az 1.3. szakaszban ismertetett eszközökkel végezzük el, amihez a szemüveg paramétereire alkalmazott konverziós formulákat [47] és a szemstruktúra becslését lehetővé tevő eljárásainkat használjuk.

A pontszórásfüggvény (1.8) egyenletű definíciójához szükség van a defókuszparaméterekre. Erre a célra a defókuszparaméter és a gyújtótávolság eltolódása közötti összefüggést alkalmaztuk. Egy adott d_o objektumtávolsághoz tartozó z_o gyújtótávolság-eltolódást a szem aberrációiból (z_α), illetve a fókusz- és objektumsík közötti távolságból (z_d) fakadó részre bontottuk. Az egyes eltolódások viszonyát az 1.10. ábra szemlélteti. z_α és z_d értékét, valamint a d_o objektumtávolsághoz tartozó f_o defókuszparamétert a következő módon kapjuk meg:

$$z_\alpha = d_{pr} - f, \quad z_d = f - \frac{1}{\frac{1}{f} + \frac{1}{d_o}}, \quad f_o = (z_\alpha + z_d)\frac{-2\pi u_0}{\lambda}, \tag{1.19}$$

ahol f a szem gyújtótávolsága, d_{pr} a pupilla és a retina távolsága, λ a fény hullámhossza, $u_0 = 1 - \sqrt{1 - s_0^2}$, $s_0 = NA/n$, n és NA pedig rendre a törésmutató és a numerikus rekesz a szem képterében.



1.10. ábra. Gyújtótávolság-eltolódás felosztása a szem aberrációiból, valamint a fókusz- és objektumsík távolságából eredő részekre.

Konvolúció komplex fázorokkal

A körszerű mintázatokkal rendelkező konvolúciós magfüggvények komplex fázoros approximációját Niemitalo javasolta [121]. Módszerének fő gondolata, hogy az $Ae^{i(\omega x+\theta)}$ fázort az e^{-x^2} Gauss-függvényhez hasonlóan x^2-tel paraméterezve és a Gauss-függvénnyel összeszorozva az alábbi konvolúciós magfüggvény definiálható:

$$K(x) = e^{-ax^2+ibx^2}, \tag{1.20}$$

ahol a és b a magfüggvény térbeli skálázását lehetővé tevő, konfigurálható paraméterek. A konvolúció a K magfüggvénnyel az alábbi módon formalizálható:

$$F(x,y) = \sum_{r_1=-R}^{R} \sum_{r_2=-R}^{R} K\left(\frac{r_1}{R}\right) K\left(\frac{r_2}{R}\right) I(x+r_1, y+r_2), \tag{1.21}$$

$$I'(x,y) = A \cdot \mathrm{Re}\left(F(x,y)\right) + B \cdot \mathrm{Im}\left(F(x,y)\right), \tag{1.22}$$

ahol I és I' a bemeneti és a kimeneti kép, R a magfüggvény sugara, $\mathrm{Re}(x)$ és $\mathrm{Im}(x)$ az x komplex szám valós és képzetes része, A és B pedig konfigurálható paraméterek, amelyek a valós és képzetes tagok súlyozását teszik lehetővé.

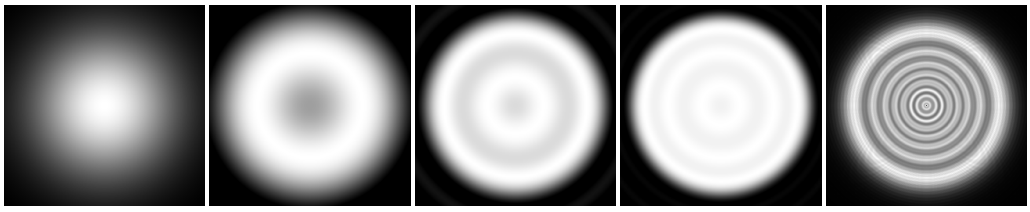
A módszer további gondolata, hogy egy adott konvolúciós magfüggvény pontosabban közelíthető több, eltérő súlyozású komplex fázoros magfüggvény összegeként. Az összegzés tagjaira a továbbiakban a *komponens* elnevezést alkalmazom. A többkomponensű fázoros konvolúció az alábbi módon formalizálható:

$$K_c(x) = e^{-a_c x^2 + i b_c x^2}, \tag{1.23}$$

$$F_c(x,y) = \sum_{r_1=-R}^{R} \sum_{r_2=-R}^{R} K_c\left(\frac{r_1}{R}\right) K_c\left(\frac{r_2}{R}\right) I(x+r_1, y+r_2), \tag{1.24}$$

$$I'(x,y) = \sum_{c=1}^{C} A_c \cdot \mathrm{Re}\left(F_c(x,y)\right) + B_c \cdot \mathrm{Im}\left(F_c(x,y)\right), \tag{1.25}$$

ahol C a komponensek száma, A_c, B_c, a_c és b_c pedig a magfüggvény súlyainak eloszlását vezérlő paraméterek. Az 1.11. ábra szemléltet néhány fázorokkal létrehozott magfüggvényt, illetve összehasonlítja azokat a Gauss-magfüggvénnyel és egy, az emberi szemhez kiszámított pontszórásfüggvénnyel.



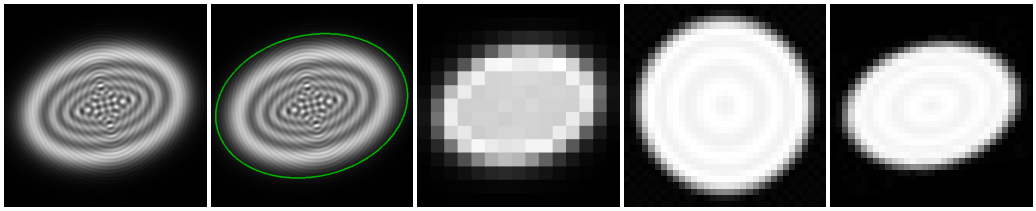
Gauss-függvény | Komplex fázor ($C=1$) | Komplex fázor ($C=2$) | Komplex fázor ($C=3$) | Valódi pontszórásfüggvény

1.11. ábra. Az emberi szem egy pontszórásfüggvényének approximációja a Gauss-függvénnyel és komplex fázorokkal.

Az (1.24) egyenletből látható, hogy a komplex fázoros konvolúció szeparábilis, hiszen egy adott bemeneti kép esetén a két összegzés szekvenciálisan is végrehajtható. Ennek következtében a mintavételek számának skálázódása kvadratikusról lineárisra csökken, hiszen egy adott kétdimenziós, R sugarú magfüggvény esetén a minták száma $(2R+1)^2$, míg szeparábilis esetben csak $(C+1)\cdot(2R+$ $+1)$. A módszer fő hátrányát a megnövekedett memóriaigény jelenti, ugyanis a horizontális és vertikális fázis között képpontonként $3C$ komplex számot kell eltárolnunk. Mindez azonban a modern videokártyák esetén nem jelent különösebb hátrányt, ahogyan azt Garcia a gyakorlatban is demonstrálta [122].

Pontszórásfüggvények approximációja fázorokkal

Az emberi látás komplex fázoros szimulációjához a pontszórásfüggvényes magfüggvény képpontonkénti méretét és súlyainak eloszlását szükséges közelítenünk. Algoritmusunk erre a célra egy többlépcsős folyamatot alkalmaz, amely folyamat releváns részeredményeit az 1.12. ábra szemlélteti.



A szem pontszórásfüggvénye | Illesztett ellipszis | Képtérre vetített pontszórásfüggvény | Fázoros magfüggvény (körszerű) | Transzformált fázoros magfüggvény

1.12. ábra. A pontszórásfüggvény komplex fázorokkal való approximációja.

Első lépésként szükség van a komponensek C darabszámára. C megválasztása a felhasználó feladata, hiszen az a memóriaigényt, a renderelési időt és az illesztés pontosságát is befolyásolja. Ezt követően adott C értékhez az A_c, B_c, a_c és b_c paramétereket kell úgy megválasztani, hogy a fázoros és a valós magfüggvény eltérése alacsony legyen. A közelítés pontosságának maximalizációjára Niemitalo optimalizációt alkalmazott [121]. Ugyanakkor Garcia demonstrálta, hogy empirikus módon is megválaszthatók olyan paraméterek, amelyek valós alkalmazások számára megfelelően jól közelítik a valós magfüggvényt [122]. Gyakorlati kísérleteink során az optimalizációval meghatározott paraméterek nem növelték számottevően a szimuláció pontosságát a Garcia által javasolt empirikus paraméterekhez képest. Ezzel szemben az optimalizációs megközelítés jelentősen meg-

hosszabbítja az előfeldolgozást, valamint számos olyan kérdést felvet, ami nagyban megnöveli az algoritmus komplexitását. Éppen ezért munkánk során Garcia empirikusan megválasztott paramétereit használtuk, aminek eredménye a korábban az 1.11. ábrán prezentált magfüggvényeken megtekinthető.

Szükséges továbbá az ellipszisszerű mintázatokat tartalmazó pontszórásfüggvények közelítését is megvalósítanunk. Ehhez az (1.24) egyenletben az I kép mintavételezését a következő módosított koordinátákkal végezzük:

$$I_x = x + r_1 \frac{R_p}{R} \cos(\varphi) - r_2 \sigma_e \frac{R_p}{R} \sin(\varphi), \tag{1.26}$$

$$I_y = y + r_1 \frac{R_p}{R} \sin(\varphi) + r_2 \sigma_e \frac{R_p}{R} \cos(\varphi), \tag{1.27}$$

ahol R_p a képpont elmosási sugara, σ_e az előfeldolgozáskor meghatározott ellipszis kis- és nagytengelyének aránya, φ pedig a tengelyek által bezárt szög. Módosított mintavételezésünk nem egészértékű mintakoordinátákat is generál, aminek kezelését a négy szomszédos képpont bilineáris interpolációjával oldottuk meg. A feladat a modern videokártyák eszközkészletével triviálisan elvégezhető hardveresen gyorsított módon, így mintavételezésünk nem okoz számottevő teljesítménybeli csökkentést, sem pedig komplexitásbeli növekedést.

Végezetül szükséges az (1.26) és az (1.27) egyenletben megjelenő R_p mintavételezési sugarat is megadnunk. Erre a célra az előfeldolgozás során előállított ritka pontszórásfüggvényrácsot alkalmazzuk, amelyből először egy adott pontszórásfüggvény képtérbeli méretét definiáljuk:

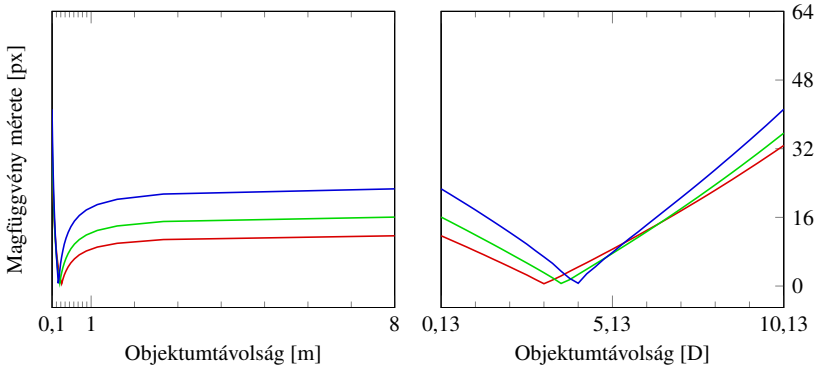
$$D_p = N\sigma, \qquad \sigma = \frac{H\mu_s}{fov_y \mu_\theta}, \tag{1.28}$$

ahol N és μ_s a pontszórásfüggvény mintáinak száma és egy minta fizikai mérete, H és fov_y a bemeneti kép magassága és az előállításához használt vertikális látószög (fokokban), μ_θ a retinán egy foknyi terület mérete (megközelítőleg 288 μm [124]), σ pedig a pontszórásfüggvény zsugorodási tényezője. Egy adott képpont elmosási sugarát a két szomszédos pontszórásfüggvény méretének lineáris interpolációjával határozzuk meg:

$$R_p = \frac{D_p^{(1)}}{2} \cdot (1 - \lambda_p) + \frac{D_p^{(2)}}{2} \cdot \lambda_p, \quad \lambda_p = \frac{d_p - d_p^{(1)}}{d_p^{(2)} - d_p^{(1)}}, \tag{1.29}$$

ahol d_p, $d_p^{(1)}$ és $d_p^{(2)}$ a képpont és a szomszédos pontszórásfüggvények mélysége, $D_p^{(1)}$ és $D_p^{(2)}$ pedig a szomszédos pontszórásfüggvények képtérbeli mérete.

Fontos kiemelnem, hogy az (1.29) egyenlet kiértékelése során az objektumtávolságot dioptriában mérjük. Ahogyan azt Barsky megemlítette munkájában [60], a szem pontszórásfüggvényének mérete jellemzően közel egyenesen arányos a dioptriában mért objektumtávolsággal, amely viszony nem áll fenn a méterben megadott mélység esetén. Ezt a megfigyelést az 1.13. ábra szemlélteti egy egészséges szem esetén. Ennek megfelelően a képpontokhoz tartozó elmosás mértéke jól becsülhető a dioptriában megadott mélység lineáris interpolációjával.

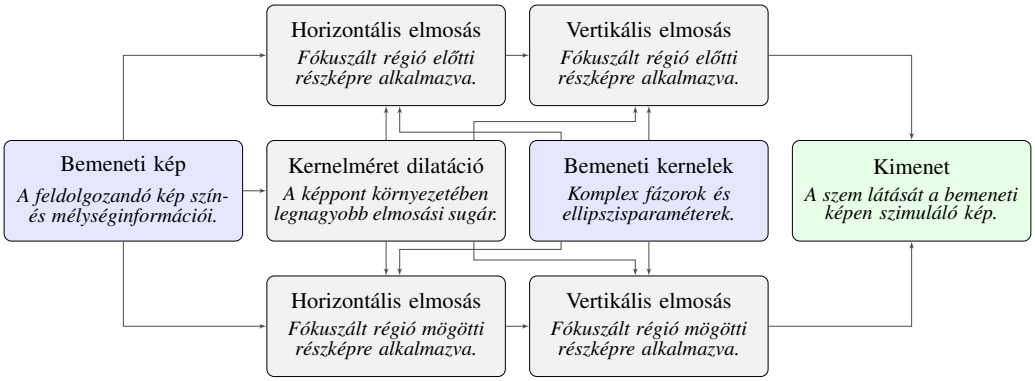


1.13. ábra. A magfüggvény D_p méretének változása a d_p objektumtávolság függvényében, három eltérő hullámhosszon. Jól látható, hogy a magfüggvény mérete közel lineárisan függ a dioptriában mért objektumtávolságtól, amely viszony nem áll fent a méterben megadott mélység esetén.

A renderelési fázis

A fent ismertetett eszközökkel elvégezhető a látást szimuláló renderelési lépés, amelyet minden képkockában végrehajtunk. Ebben a fázisban az előfeldolgozás során létrehozott adatokat, illetve a feldolgozandó kép képpontonkénti szín- és mélységinformációit használjuk a látásszimulációhoz. A szakasz legfontosabb lépéseit az 1.14. ábra szemlélteti.

Kezdésként létrehozunk egy dilatált elmosásisugár-puffert, amely minden képponthoz tartalmazza a képpont egy konfigurálható méretű környezetében fellelhető, legnagyobb R_p elmosási sugarat. Ezt követően a bemeneti képet felbontjuk egy előtérre és egy háttérre. Választópontként azt a mélységértéket használjuk, amely az előfeldolgozás során meghatározott pontszórásfüggvények közül a



1.14. ábra. A renderelési szakasz legfontosabb lépései.

legkisebb elmosási sugárral rendelkezik. A felbontás után először elvégezzük a horizontális elmosást az előtérre és a háttérre az (1.24) egyenlet alapján, amely az előtérhez és a háttérhez is komponensenként egy ideiglenes képet eredményez. Ezt követően végrehajtjuk a vertikális elmosást az ideiglenes részképre. Végezetül az (1.25) egyenletnek megfelelően összegezzük az előtér és a háttér összes részeredményét egyetlen képpé, ami az algoritmus kimenetét szolgáltatja.

1.5.2. Továbbfejlesztett, csempealapú módszerünk

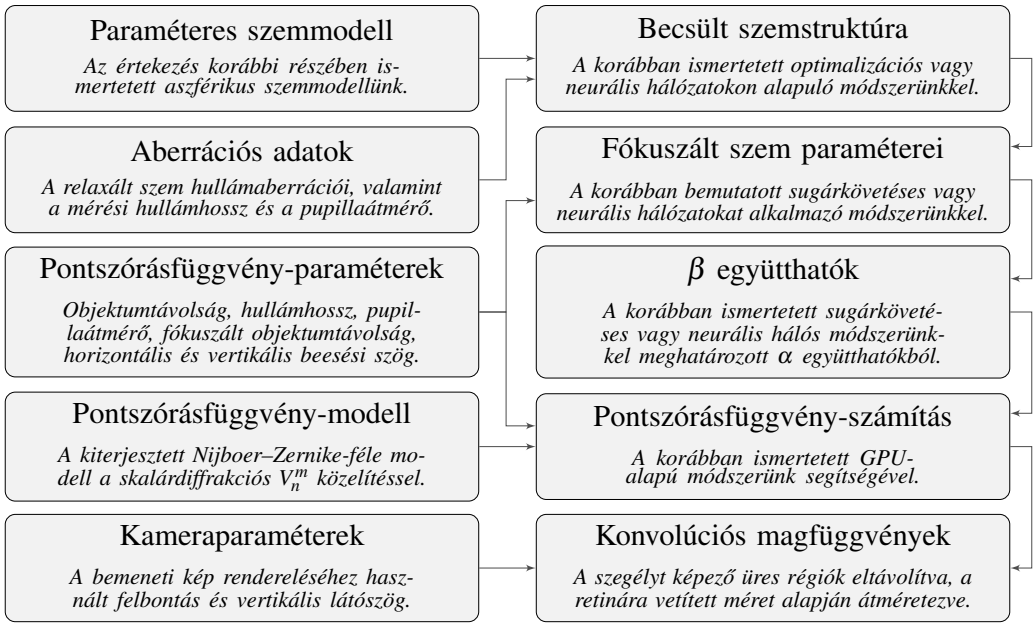
Az emberi látáshoz hű renderelésre megalkotott másik algoritmusunk fő célja a konvolúció hatékonyságának ötvözése a perifériás látás és a kromatikus aberráció szimulációjának képességével. Ezek a jelenségek értékes információkat hordoznak az agy számára és nagyban hozzájárulnak a mindennapos tevékenységeink hatékony elvégzéséhez [30, 31, 125, 126]. Ezenkívül a vizuális aberrációk gyakran váratlan módon viselkednek a perifériás régióban. Ezt jól szemlélteti a *relatív perifériás hiperópia* jelensége, amely során a rövidlátó szem által érzékelt kép éles a perifériás régió bizonyos távoli részeiben [62]. További célunk volt a tetszőleges összetételű hullámaberrációk támogatása a szimulálható állapotok bővítésének érdekében. Végezetül a szimulált szem pupillaátmérőjének és fókusztávolságának dinamikus módosíthatóságát is meg akartuk valósítani, ily módon növelve a szimulált szem interaktív vizsgálatának képességét.

Céljaink eléréséhez egy olyan konvolúciós eljárást alkottunk meg, amely képes a szem valós pontszórásfüggvényeinek képpontonkénti precíz becslésére. Barsky módszeréhez [60] hasonlóan egy ritka pontszórásfüggvényrácsot alkalmaztunk, amelyhez megalkottunk egy GPU-alapú interpolációs stratégiát a mag-

függvény képpontonkénti approximációjára. A hatékony konvolúcióhoz Franke és mtsai. csempealapú eljárását [127] terjesztettük ki a képpontonként változó magfüggvényű konvolúcióval. Létrehozott módszerünk egy egyszeri előfeldolgozási szakaszra és egy képkockánkénti renderelési fázisra bontható. Ezek részleteit a 2021-ben [75] és a 2024-ben [76] közzétett cikkünk alapján ismertetem.

Az előfeldolgozási szakasz

Az előfeldolgozási lépés fő célja a konvolúciós magfüggvények előállítása. A szakasz legfontosabb lépéseit az 1.15. ábra mutatja be.



1.15. ábra. Az előfeldolgozási szakasz legfőbb lépései.

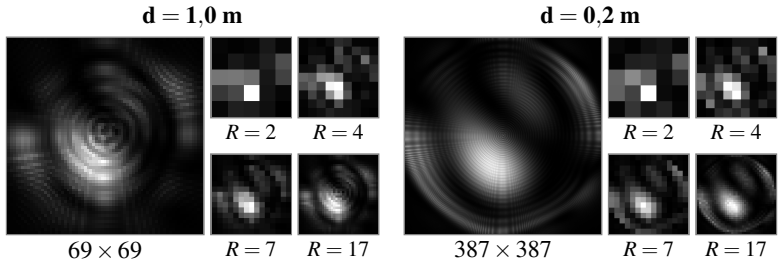
Csempealapú algoritmusunkhoz az objektumtávolságon kívül a pontszórásfüggvény egyéb paramétereit is szükséges mintavételeznünk a dinamikus pupillaátmérő és fókusztávolság, a kromatikus aberráció, illetve a perifériás látás szimulációja céljából. Ennek következtében a ritka pontszórásfüggvényrácsunk tengelyeit a d objektumtávolság, a λ hullámhossz, az A pupillaátmérő, az f fókuszált objektumtávolság, illetve a h horizontális és a v vertikális beesési szög képezik.

A pontszórásfüggvényrács elemeinek paramétereit illetően, az objektumtávolsághoz és a fókuszált objektumtávolsághoz tartozó paramétereket dioptriában

mérve lineárisan mintavételeztük. Ennek oka, hogy Barsky észrevétele [60] szerint a pontszórásfüggvény mérete közel arányosan változik a dioptriában mért távolsággal, ahogyan azt korábban az 1.13. ábra segítségével demonstráltam. A pupillaátmérőkhöz a lineárisan mintavételezett 2–7 mm tartományt használtuk, amelyet Watson és Yellott [128] eredményei alapján alkalmasnak ítéltünk a hétköznapi szituációk lefedésére. A beesési szögek paramétereit szintén lineárisan mintavételeztük. Ehhez egy 50°-os vertikális látószögű kameramodellt és a modern kijelzőkre jellemző téglalap alakot feltételezve a $[-45°, 45°]$ horizontális és a $[-25°, 25°]$ vertikális tartományt alkalmaztuk. Végezetül a hullámhosszakat kézzel adtuk meg, egy-egy egyedi hullámhosszt rendelve az RGB színtér három csatornájához. A mintavételezési darabszámokat a felhasználás módjától függően választottuk meg, amelyekre a továbbiakban N_d, N_f, N_A, N_h, N_v és N_λ néven hivatkozok, értéküket pedig az eredményeink tárgyalásakor ismertetem.

Mivel módszerünk célja tetszőleges aberráció szimulációja, így bemenetként a szimulált szem relaxált állapotának aberrációs együtthatóit alkalmaztuk. Erre támaszkodva először elvégezzük a fizikai szemstruktúra becslését, majd az így kapott szemmodellből elkészítjük az egyes fókusztávolságokhoz tartozó fókuszált szemmodelleket. Ezután a pontszórásfüggvényrács minden eleméhez meghatározzuk az aberrációs együtthatókat. Mindezen lépéseket az 1.3. szakaszban bemutatott eszközeinkkel végezzük el. Végezetül előállítjuk a ritka pontszórásfüggvényrács elemeit az 1.4. szakaszban demonstrált módszerünkkel.

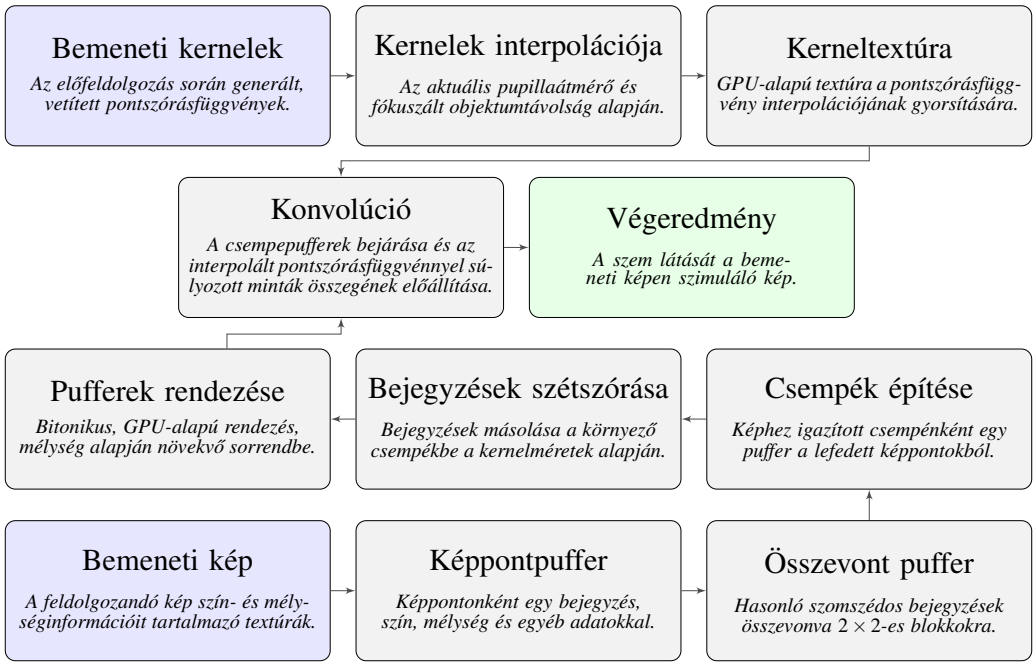
A pontszórásfüggvényrács birtokában előállítjuk a szimulációhoz szükséges konvolúciós magfüggvényeket. Ehhez először kiszámítjuk az egyes magfüggvények méretét, amelyet a pontszórásfüggvény az 1.5.1. alszakaszban az (1.28) egyenlettel megadott képtérbeli méreteként határozunk meg. Ezt követően kiszámítjuk minden egyes magfüggvényhez, hogy az interpolációs folyamat során arra milyen elmosási sugarakban lesz szükség. Ehhez a rács szomszédos elemei között fellelhető legkisebb és legnagyobb képtérbeli magfüggvényméreteket használjuk. A magfüggvények pontos méretének birtokában átméretezzük a pontszórásfüggvényeket az összes szükséges elmosási sugárnak megfelelően, ami az interpoláció alapjául szolgáló ritka magfüggvényrácsot eredményezi. Végezetül az így kapott magfüggvényeket feltöltjük a videomemóriába, hogy a renderelési fázis során felhasználhassuk azokat a GPU-alapú interpolációs technikánkkal. A folyamatot az 1.16. ábra szemlélteti két példa pontszórásfüggvényre.



1.16. ábra. Eredeti pontszórásfüggvények és néhány belőlük generált, eltérő elmosási sugarú (r) magfüggvény, két objektumtávolság (d) esetén.

A renderelési szakasz

A renderelési fázis Franke és mtsai. csempealapú eljárására épül [127], amelyet kibővítettünk a valós pontszórásfüggvényt alkalmazó konvolúcióval. A szakasz bemenetét a feldolgozandó kép szín- és mélységinformációit tartalmazó textúrák, a rendereléshez használt kameraparaméterek, illetve az előfeldolgozás során kiszámított magfüggvényadatok képezik. A szakasz kimenete pedig a bemenethez tartozó, látást szimuláló kép. A fázis főbb lépéseit az 1.17. ábra foglalja össze.



1.17. ábra. A renderelési fázis legfontosabb lépései.

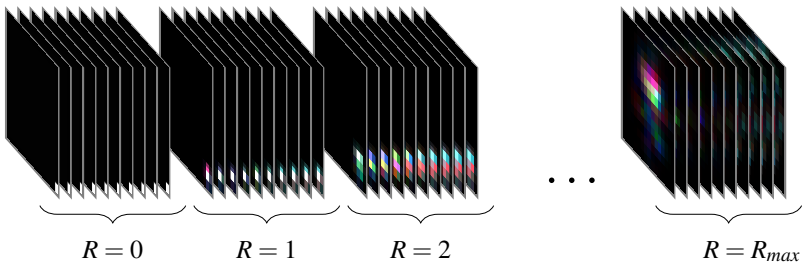
A ritka magfüggvényrács felhasználásával először feltöltünk egy GPU-alapú gyorsítótextúrát, amely az adott képkockához tartozó pupillaátmérőhöz és fókuszált objektumtávolsághoz tartozó konvolúciós magfüggvényeket tartalmazza. Ez a textúra lehetővé teszi a valós, folytonos pontszórásfüggvény hardveresen gyorsított közelítését. Ezt követően összegyűjtjük a bemeneti kép képpontjainak szín- és mélységinformációit egy GPU pufferbe, majd a feldolgozandó adatmennyiség csökkentéséhez a kellően hasonló szomszédos képpontok bejegyzéseit összevonjuk. Következő lépésként a képet csempékre osztjuk és minden csempéhez építünk egy puffert az előző lépésben generált bejegyzésekből. A későbbi gyors hozzáférhetőség érdekében a pufferekben található bejegyzéseket a magfüggvényeik elmosási sugara alapján átmásoljuk a környező csempék puffereibe. Ennek eredményeként minden puffer tartalmazza az összes olyan mintát, amely a csempe által lefedett képpontok számára releváns lehet. Ezt követően a csempepufferek tartalmát mélység szerint növekvő sorrendbe rendezzük a helyes takarási viszonyok kezeléséhez. Végezetül a kimeneti kép minden képpontjához bejárjuk a hozzá tartozó csempepuffert és előállítjuk a releváns minták súlyozott összegét. Ehhez azon mintákat tekintjük, amelyek magfüggvénye lefedi az adott kimeneti pixelt, a magfüggvény mintához tartozó súlyának előállításához pedig a szakasz elején létrehozott textúrát alkalmazzuk.

Magfüggvény mélységfüggő interpolációja

A magfüggvény interpolációjának első lépése a folyamat hardveres gyorsításához használt háromdimenziós GPU textúra előállítása. A textúrát az előfeldolgozás során statikusan allokáljuk, a méretét oly módon megválasztva, hogy a textúrában elférjen minden magfüggvény az összes releváns elmosási sugárra méretezve:

$$(2R_{max}+1,\ 2R_{max}+1,\ N_d \cdot (R_{max}+1)), \tag{1.30}$$

ahol R_{max} a legnagyobb elmosási sugár. Ezt követően a jelenlegi képkockához tartozó A_c pupillaátmérő és f_c fókuszált objektumtávolság alapján kiszámítjuk az elmosási sugarat a ritka magfüggvényrács mélységértékeinek és hullámhosszainak minden kombinációjához, majd egy GPU pufferben eltároljuk az eredményt. Végezetül feltöltjük a textúra rétegeit, amely során a képtérre vetített magfüggvényeket az elmosási sugár és az objektumtávolság szerint rendezve elhelyezzük a rétegeken. A hullámhossztól függő magfüggvényeket az RGB textúra csatornáiban a bal alsó sarokba igazítva tároljuk. Az elrendezést az 1.18. ábra szemlélteti.



1.18. ábra. A háromdimenziós gyorsítótextúránkban tárolt magfüggvények elrendezése. Az egyes pontszórásfüggvényekhez tartozó magfüggvényeket elmosási sugár, azon belül pedig objektumtávolság szerint rendezve tároljuk el.

Textúraelrendezésünk fő előnye, hogy a két szomszédos objektumtávolság közötti interpoláció a textúra egyetlen mintavételezésével elvégezhető. Továbbá, mivel minden magfüggvény eltérő rétegen, a síkban ugyanazon koordinátához igazítva helyezkedik el, így komplex textúrakoordináta-számítások sem szükségesek, és interpolációs áthatások sem jelentkeznek.

A kromatikus magfüggvény egy képponthoz tartozó mintájának előállítására csatornánként két textúra-mintavételezést hajtunk végre. Ehhez először meghatározzuk a legközelebbi magfüggvény indexét:

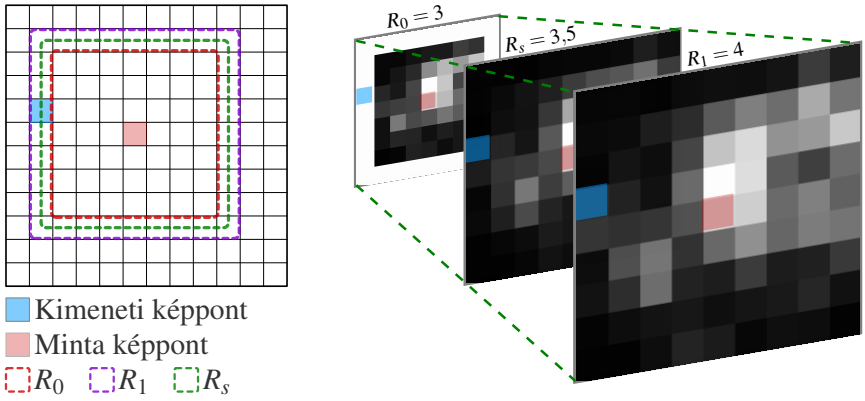
$$I_s = \frac{1}{D_\Delta} \cdot \left(\frac{1}{d_s} - D_{min} \right), \tag{1.31}$$

ahol d_s a képpont objektumtávolsága méterben, D_{min} a ritka rács legkisebb objektumtávolsága dioptriában, D_Δ pedig a ritka rács objektumtávolság-mintái közötti különbség dioptriában. I_s egyértelműen beazonosítja a két interpolálandó magfüggvényt és biztosítja a keverési tényezőt a törtrészben. Ezután meghatározzuk a minta R_s elmosási sugarát a megtalált magfüggvényekhez tartozó elmosási sugarak interpolációjával (I_s törtrészét alkalmazva keverési tényezőként), majd R_s-t lefelé és felfelé kerekítve két mintát veszünk a magfüggvénytextúrából. A mintavételezéshez használt textúrakoordinátákat az alábbi módon határozzuk meg:

$$(x_c - x_s + r,\ y_c - y_s + r,\ I_s + rN_d), \tag{1.32}$$

ahol r a (felfelé vagy lefelé) kerekített sugár, (x_c, y_c) és (x_s, y_s) pedig a kimeneti és a minta képpont képtérbeli koordinátái. A textúra ily módon való mintavételezése garantálja a mélység szerinti interpoláció hardveresen gyorsított végrehajtását.

Végezetül a csatornához tartozó súlyértéket a két minta lineáris interpolációjával kapjuk meg, R_s törtrészét alkalmazva keverési tényezőként. Az 1.19. ábra szemlélteti a folyamatot egy mintára.



1.19. ábra. A magfüggvény egy csatornához tartozó súlyának meghatározása. Az elmosási sugár két egészértékű szomszédjával mintavételezzük a gyorsítótextúrát, majd a kapott mintákat manuálisan interpoláljuk a kívánt magfüggvényméret törtrésze alapján.

Hatékonyabb textúraelrendezés és mintavételezés

Az előző részben bemutatott gyorsítótextúra és mintavételezés hátrányait a rétegek nagy száma, illetve az RGB csatornák és az egészértékű elmosási sugarak manuális mintavételezése jelentik. Ennek eredményeként egy kromatikus magfüggvényminta meghatározásához hat darab textúraminta szükséges, amelyek kiértékelési költségét a szignifikáns textúraméret nagyban megnöveli. Továbbá a textúra méretéből fakadóan a gyorsító kiterjesztése is körülményes a perifériás látás szimulációjára.

A fent leírt problémák megoldására kidolgoztunk egy hatékonyabb textúraelrendezést, amely a gyorsítótextúra rétegeit az objektumtávolság mintavételezésével választja meg. A renderelési szakasz elején kiszámítjuk minden réteg interpolált elmosási sugarát a réteg objektumtávolsága alapján. Ezután az egészértékű sugarak és az objektumtávolság szerint a ritka rács szomszédos magfüggvényeiből előállítjuk a réteghez tartozó magfüggvényt, amelyet a réteg közepéhez igazítva eltárolunk. Ily módon a kromatikus minták egyetlen textúramintavétellel kiszámíthatók, jelentősen lecsökkentve az interpoláció költségét. A rétegek darabszámát és a hozzájuk tartozó objektumtávolságokat képkockánként határozzuk

meg. A magfüggvényméretek eltéréséből fakadó műtermékek elkerülése érdekében kritikus, hogy minden szomszédos rétegpár elmosási sugarának különbsége egynél kisebb legyen. Továbbá a memóriaigény minimalizálásához a dinamikus mintavételezést oly módon végezzük el, hogy az ezen feltétel teljesítéséhez szükséges legkevesebb rétegszámot alkalmazzuk.

Céljainkhoz a magfüggvényrács minden objektumtávolságához egy egyedi réteget rendelünk, amelyek közé eltérő számú átmeneti réteget helyezünk el. Az átmeneti rétegek számát az alábbi módon határozzuk meg:

$$L_d^k = \lceil |R_{k+1} - R_k| \rceil - 1, \tag{1.33}$$

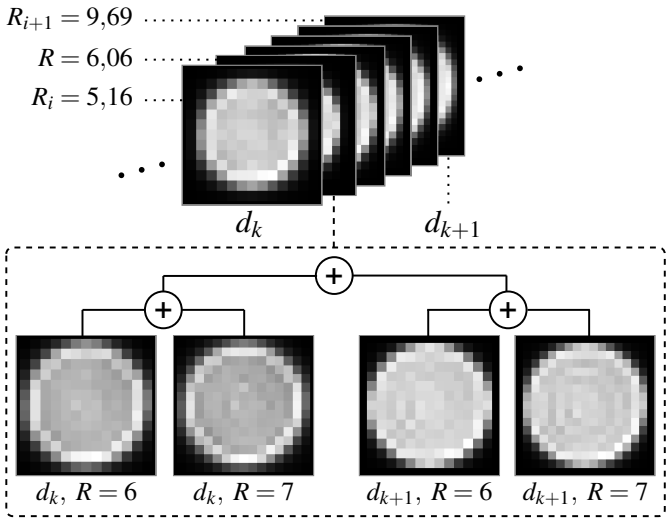
ahol d_k a ritka rács k-adik objektumtávolsága, L_d^k a d_k-hoz tartozó réteg után beszúrt átmeneti rétegek száma, R_k a d_k távolság három csatornája közötti legnagyobb elmosási sugár a jelenlegi képkockában, $\lceil \cdot \rceil$ pedig a felfelé kerekítést jelöli. A gyorsítótextúra mérete a következő:

$$\left(2R_{max}+1,\ 2R_{max}+1,\ N_d + \sum_{k=1}^{N_d-1} \overline{L}_d^k\right), \tag{1.34}$$

ahol R_{max} a legnagyobb elmosási sugár, $\overline{L}_d^k$ pedig a ritka magfüggvényrács k-adik objektumtávolságához tartozó hullámhossz, pupillaátmérő és fókusztávolság kombinációk közötti legnagyobb L_d^k értéket jelöli. A mintavételezési textúrakoordinátákat az alábbi módon számítjuk ki:

$$\left(x_c - x_s + R_{max},\ y_c - y_s + R_{max},\ \mathrm{frac}\,(I_s) \cdot \left(L_d^{\lfloor I_s \rfloor} + 1\right) + \sum_{k=1}^{\lfloor I_s \rfloor - 1} \left(L_d^k + 1\right)\right), \tag{1.35}$$

ahol $\mathrm{frac}\,(x)$ az x törtrészét, $\lfloor \cdot \rfloor$ pedig a lefelé kerekítést jelöli. A törtértékű I_s magfüggvényindex garantálja, hogy a mélység szerinti interpoláció hardveresen gyorsítva, egyetlen mintával megtörténik, a rétegek létrehozásából fakadóan a folytonos elmosási sugarat is figyelembe véve. Továbbá a textúra középre rendezéséből eredően a magfüggvényen kívül eső minták értéke zérus, ily módon kezelve a kromatikus aberrációból származó, csatornánként változó elmosási sugarakat. A textúraelrendezésünket és egy adott réteg egy hullámhosszának kiszámítását az 1.20. ábra szemlélteti.



1.20. ábra. Objektumtávolság-alapú textúraelrendezésünk ($L_d^k = 4$). A ritka rács minden d objektumtávolságához egy egyedi réteget helyezünk el, amelyek közé dinamikusan megválasztott számú átmeneti réteget szúrunk be.

Periférikus magfüggvények kezelése

A dinamikus, objektumtávolság-alapú textúraelrendezésünk memóriaigénye kellően alacsony ahhoz, hogy azt kiegészíthessük a periférikus látás szimulációjához szükséges magfüggvényekkel. Ehhez az 1.20. ábrán demonstrált, mélységfüggő magfüggvényeket tartalmazó textúrát egy blokknak tekintjük, majd horizontálisan és vertikálisan is elhelyezünk egymás mellett több, a beesési szögekhez tartozó blokkot. Ennek eredményeként a magfüggvényekből egy háromdimenziós rácsot képezünk a gyorsítótextúrában, melynek elemeire a fenti jelöléssel való konzisztencia érdekében a továbbiakban is csak *réteg*ként hivatkozok.

A korábbiakhoz hasonlóan a gyorsítótextúrát minden képkockában frissítjük, a rácspontok számát pedig dinamikusan állapítjuk meg. A rácspontok meghatározásához a fenti megfontolások alapján először a ritka magfüggvényrács minden h_i horizontális beesési szög, v_j vertikális beesési szög, és d_k objektumtávolság kombinációjához hozzárendelünk egy réteget. Ezt követően a folytonos átmenet érdekében a rétegek közé a rács minden tengelyén átmeneti rétegeket helyezünk el. A beszúrt rétegek darabszámát a következő módon számítjuk ki:

$$L_h^i = \max_{\substack{j=1,\ldots,N_v \\ k=1,\ldots,N_d}} L(R_{i,j,k}, R_{i+1,j,k}), \tag{1.36}$$

$$L_v^j = \max_{\substack{i=1,\dots,N_h \\ k=1,\dots,N_d}} L(R_{i,j,k}, R_{i,j+1,k}), \tag{1.37}$$

$$L_d^k = \max_{\substack{i=1,\dots,N_h \\ j=1,\dots,N_v}} L(R_{i,j,k}, R_{i,j,k+1}), \tag{1.38}$$

$$L(R_1, R_2) = \lceil |R_1 - R_2| \rceil - 1, \tag{1.39}$$

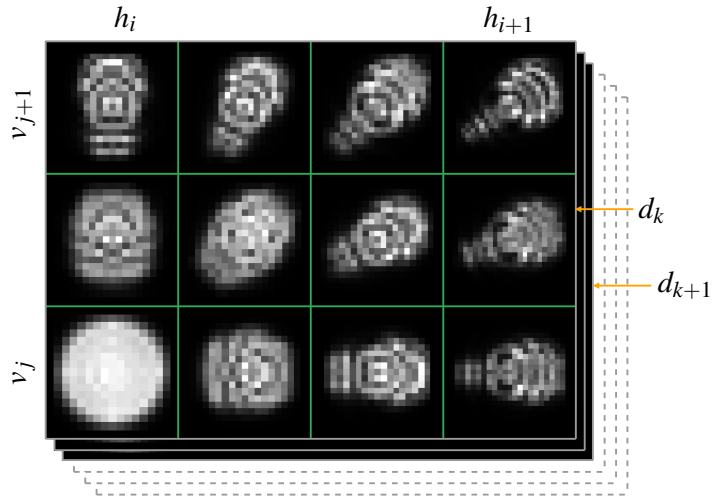
ahol $R_{i,j,k}$ a $\left(h_i, v_j, d_k\right)$-hoz tartozó magfüggvény három csatornája közötti legnagyobb elmosási sugár a jelenlegi képkockában, L_h^i, L_v^j és L_d^k pedig rendre a beszúrt rétegek száma a horizontális, a vertikális és az objektumtávolság tengelyen a $\left(h_i, v_j, d_k\right)$-hoz tartozó réteg után. A gyorsítótextúra mérete az alábbi:

$$H = (2R_{max} + 1) \cdot \left(N_h + \sum_{i=1}^{N_h - 1} \overline{L}_h^i\right), \tag{1.40}$$

$$W = (2R_{max} + 1) \cdot \left(N_v + \sum_{j=1}^{N_v - 1} \overline{L}_v^j\right), \tag{1.41}$$

$$D = N_d + \sum_{k=1}^{N_d - 1} \overline{L}_d^k, \tag{1.42}$$

ahol $H \times W \times D$ a textúra mérete, R_{max} a legnagyobb elmosási sugár, $\overline{L}_h^i$, $\overline{L}_v^j$ és $\overline{L}_d^k$ pedig rendre a hullámhossz, pupillaátmérő és fókusztávolság kombinációk közötti legnagyobb L_h^i, L_v^j és L_d^k érték. Elrendezésünket az 1.21. ábra demonstrálja.



1.21. ábra. Periférikus magfüggvényeket támogató textúraelrendezésünk ($L_h^i = 2$, $L_v^j = 1$, $L_d^k = 1$), amelyet az 1.20. ábrán szemléltetett módszer kiterjesztésével hoztunk létre.

A kromatikus magfüggvényminták meghatározását az optikai tengely pontjaihoz megalkotott textúránk mintavételezésére vezetjük vissza. Egy adott képpont esetén először annak horizontális és vertikális beesési szöge alapján meghatározzuk a mintavételezendő blokkok indexeit (I_h^s és I_v^s). Ezt követően az azok lefelé és felfelé kerekítésével keletkező négy blokkot mintavételezzük az (1.35) egyenlettel. Végezetül az eredményt a négy minta bilineáris interpolációjával számítjuk ki, az I_h^s és az I_v^s index törtrészét alkalmazva interpolációs tényezőként.

Az aberrációk mértéke tipikusan lényegesen magasabb a perifériás régióban, ami a perifériás magfüggvények méretének hatalmas megnövekedéséhez vezet. A pusztán az optikai tengely pontjaihoz tartozó gyorsítótextúrához viszonyított lényeges rétegszámbeli ugrással kombinálva mindez a szükséges videomemória exponenciális növekedését eredményezi. Ennek következtében a gyakorlatban a fenti megközelítéssel létrehozott textúrák mérete jellemzően nem alkalmas a jelenleg elérhető GPU-k számára. A probléma megoldására az (1.39) egyenletben található, átmeneti rétegek számát megadó L függvényt egy redukciós komponenssel egészítettük ki:

$$L'(R_1,R_2) = (sL(R_1,R_2))^{-p} \cdot L(R_1,R_2), \tag{1.43}$$

ahol s és p konfigurálható paraméter. Megjegyzem, hogy az átmeneti rétegek számának fix értékként való kezelése ($L'(R_1,R_2) = C$, ahol C egy konfigurálható paraméter) egy további lehetséges megoldást jelent, viszont az (1.43) egyenlettel definiált dinamikus megközelítésünk sokkal robusztusabb a ritka rács alacsony mintavételezésére és az elmosási sugarak ebből fakadó nagymértékű eltéréseire.

Végezetül fontos kiemelnem, hogy a perifériás elrendezés két további tengellyel bővíti az interpolálandó magfüggvények számát. Ennek következtében a képkockánkénti gyorsítótextúra egyetlen értékének előállításához a ritka rács 192 mintája szükséges (csatornánként 64 minta). Gyakorlati kísérleteink során a textúra megnövekedett mérete miatt a textúrafeltöltő lépés hossza tipikusan egy nagyságrenddel nagyobb volt, mint a teljes konvolúciós fázis. A probléma kiküszöbölésére az előfeldolgozás során a ritka rács minden pupillaátmérő és fókusztávolság kombinációjához elkészítünk egy GPU-n tárolt textúrát a fent leírt perifériás elrendezéssel. Ezután képkockánként egyszerűen A_c és f_c alapján megkeressük a két-két releváns textúrát és bilineáris interpolációval kiszámítjuk a gyorsítótextúra képpontjait. Továbbá a videomemória ily módon megnövekedett igényének csökkentéséhez nem alkalmazunk átmeneti rétegeket ($L'(R_1,R_2) = 0$).

Bár döntésünk csökkenti a szimuláció pontosságát, ahogyan azt eredményeink kiértékelésekor demonstrálom, a csökkenés mértéke nem számottevő. Fontos további tényező azonban, hogy az átmeneti rétegek elhagyásával az előfeldolgozás során készített textúrák mérete megegyezik a képkockánként újraépített gyorsítótextúra méretével. Ennek köszönhetően a gyorsítótextúra minden képpontja meghatározható négy hardveresen gyorsított textúra-mintavételezéssel.

1.5.3. Elért eredmények

Tesztkörnyezet

Új módszereink kiértékelésére létrehoztuk azok referenciaimplementációját a C++ programozási nyelv és az OpenGL grafikus programkönyvtár [115] segítségével. Az előfeldolgozási fázisokban az értekezés korábbi részeiben tárgyalt algoritmusaink referenciaimplementációját alkalmaztuk. A látásszimulációt képkockánként végrehajtott szűrőként implementáltuk, amelynek bemenetét raszterizációval, 1280×720-as felbontással és egy 50°-os vertikális látószögű kamerával állítottuk elő. Megjegyzem, hogy módszereink más forrásból származó bemenetekkel (például sugárkövetés vagy RGB-D kamerás felvételek) is kompatibilisek, amennyiben a képpontonkénti szín- és mélységinformációk elérhetőek.

Vizsgálatainkhoz két eltérő komplexitású jelenetet választottunk: egy egyszerű, primitívekből álló kompozíciót (*primitívek*), illetve egy lényegesen összetettebb, valósághű jelenetet (*San Miguel*). A komplex fázoros módszerünk elemzéséhez a miópia és asztigmia szimulációját vizsgáltuk, mivel ezen eljárásunk célja az alacsonyrendű aberrációkkal terhelt látás szimulációja volt. A valós pontszórásfüggvény közelítésén alapuló csempézett konvolúciós algoritmusunk kiértékeléséhez pedig az 1.3.4. alszakaszban vizsgált hat állapotot (emmetrópia, miópia, asztigmia, keratokónusz, katarakta, LASIK műtét) tekintettük. Minden szemállapothoz 5 mm-es pupillaátmérőt és 8 m-es fókuszált objektumtávolságot alkalmaztunk a futási idők és a pontosságok mérésekor.

A konvolúcióval elérhető legmagasabb minőségű referenciaszimulációk előállítására implementáltunk egy CPU-alapú eljárást, amely a képpontok valós pontszórásfüggvényével végzett konvolúcióval szimulálja a látást. Ehhez a képpontokat először objektumtávolság és opcionálisan beesési szögek alapján osztályoztuk egy alacsony küszöbérték segítségével, majd kiszámítottuk az osztályokhoz tartozó pontszórásfüggvényeket és elvégeztük a konvolúciót. A csempézett konvolúciós módszerünk optikai tengely pontjainak aberrációit szimuláló

változatának létező módszerrel való összehasonlításához implementáltuk Barsky mélységfüggő szeleteket alkalmazó konvolúciós eljárását [60]. Módszerünk periférikus látást szimuláló változatát is összehasonlítottuk létező, konvolúcióra épülő módszerekkel, amihez Rodríguez Celaya és mtsai. [61] képpontonkénti magfüggvény-interpolációs, valamint Gonzalez Utrera [23] mélységfüggő szeletekre épülő algoritmusát is implementáltuk. A rendelkezésre álló hardveres erőforrások megfelelő kihasználásához CPU-alapú párhuzamosítást alkalmaztunk minden eljárás implementációja során, a szükséges pontszórásfüggvényeket pedig GPU-alapú gyorsítással állítottuk elő.

Az összehasonlításhoz használt módszerek kiválasztásakor törekedtünk a szakirodalomban elérhető legújabb, tetszőleges aberráció szimulációjára alkalmas algoritmusokat alkalmazni. Ezen módszerek közül azonban csak konvolúciós megközelítéseket vettünk számításba. Ennek fő oka, hogy a sugárkövetéses eljárások számottevő, a módszerre jellemző sajátossággal rendelkeznek [66–68]. Ilyen sajátosság például a véletlen mintavételezésből fakadó zaj és a pupilla alakjából fakadó görbült és korlátos kimenet, amelyek nagymértékben csökkentik az eredmények összehasonlíthatóságát. Az ilyen eltérések minimalizációjára az összehasonlításhoz használt módszerek esetén is a saját algoritmusaink alapjául szolgáló ENZ modellt alkalmaztuk. Továbbá a szemmodell és a szimuláció jellemzői miatt a sugárkövetéses módszerek futási ideje messzemenően meghaladja az interaktív környezetekhez elfogadható korlátot. Mindezen okok miatt úgy ítéltük, hogy a sugárkövetéses módszerek összehasonlítása az általunk javasolt interaktív eljárásokkal helytelen és nagyban félrevezető eredményekhez vezetne.

A szimuláció pontosságának kiértékelésére a referencia kimenethez viszonyított maximális jel-zaj viszonyt (PSNR) használtuk. Ezt a metrikát a szakirodalomban gyakran alkalmazzák a szimulált képek hitelességének mérésére [129], amelyet számos releváns renderelő eljárás tesztelésére is felhasználtak [14, 32, 71, 130]. A PSNR az átlagos négyzetes hibából (MSE) származtatható, kiszámítása pedig többcsatornás, csatornánként nyolcbites képek esetén a következő [129]:

$$PSNR = 10\log_{10}\frac{255^2}{MSE}, \tag{1.44}$$

$$MSE = \frac{1}{M\cdot N\cdot C}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\sum_{c=0}^{C-1}\left[f(n,m,c)-g(n,m,c)\right]^2, \tag{1.45}$$

ahol f és g a két összehasonlítandó, $M\times N$ méretű, C csatornás képet jelöli.

A futási idők mérésére egy AMD Ryzen 7 1700 3.00 GHz processzort és egy NVIDIA TITAN Xp videokártyát alkalmaztunk.

Alacsonyrendű aberrációk szimulációja fázoros módszerünkkel

Módszerünk kiértékeléséhez a fázoros magfüggvény 17×17 mintát, illetve $C = 1$ és $C = 2$ komponenst tartalmazott. Ezenkívül szimuláltuk a korábbi módszerek által gyakran használt Gauss-függvényes konvolúciót is, amelyhez $C = 1$ komponenst és a megfelelő komponenssúlyokat alkalmaztuk. A magfüggvény méretének approximációjára 81 objektumtávolság-függő pontszórásfüggvényt számítottunk ki, ami körülbelül három másodpercig tartott. Így tehát elmondható, hogy a rövid előfeldolgozást érintő célunkat sikeresen teljesítettük. A folytonos pontszórásfüggvény kiértékelésén alapuló referenciamódszer 595 (*primitívek*) és 419 (*San Miguel*) mélységfüggő osztályt azonosított és megközelítőleg 11 perc számítást igényelt képenként. A generált kimeneteket a Függelékben található B.1. ábra szemlélteti.

Először megmértük módszerünk futási idejét a vizsgált tesztesetekre, aminek eredményét az 1.7. táblázat foglalja össze. Méréseink alapján egyértelműen megállapítható, hogy módszerünk kiváltképp alkalmas valósidejű rendszerek számára. Továbbá a futási idő konzisztens és a bemenet összetételétől teljesen független, ami szintén kulcsfontosságú a gyakorlati alkalmazhatóság számára. Fontos azonban megjegyeznem, hogy a Gauss-függvényes konvolúció futási idejét nem tüntettem fel. Ennek oka, hogy az eljárás komplex fázorokkal való közelítése számottevően megnöveli a szimuláció futási idejét, ily módon pedig az alkalmatlan egzakt összehasonlítások alapjául szolgálni.

		Primitívek			**San Miguel**		
Állapot	**Metrika**	**Gauss**	**C = 1**	**C = 2**	**Gauss**	**C = 1**	**C = 2**
Miópia	Futási idő	-	1,45 ms	2,30 ms	-	1,79 ms	2,73 ms
	PSNR	32,53 dB	32,66 dB	32,39 dB	31,73 dB	33,02 dB	33,13 dB
Asztigmia	Futási idő	-	1,41 ms	2,27 ms	-	1,80 ms	2,77 ms
	PSNR	32,76 dB	32,87 dB	33,05 dB	31,34 dB	32,37 dB	32,64 dB

1.7. táblázat. Futási idők és a referenciához viszonyított PSNR pontossági értékek a Gauss-függvénnyel és a fázorokon alapuló eljárásunkkal végzett szimulációkhoz. Jól látható, hogy fázoros magfüggvényünk konzisztensen alacsonyabb hibákat generál, mint a tradicionális Gauss-függvény.

A módszerünk pontosságát mérő, referenciához viszonyított PSNR értékek szintén az 1.7. táblázatban olvashatók. Fontos kiemelnem, hogy a valós pontszórásfüggvények reprodukciója rendkívüli kihívást jelent a szeparábilis magfüggvényeken alapuló, teljesítményt előtérbe helyező módszerek számára. Eredményeink alapján azonban úgy ítéljük, hogy komplex fázoros megközelítésünk valós alkalmazások számára megfelelő pontosságot ért el. Méréseinkből az is látható, hogy fázoros módszerünk még a Garcia által javasolt [122] empirikus súlyokkal is pontosabban közelíti a szem valós pontszórásfüggvényeit, mint az egyszerű Gauss-függvény. Szükség esetén a pontosság a súlyok Niemitalo által javasolt illesztésével tovább fokozható [121], ám az elért pontosság és az optimalizáció korábban kiemelt hátrányai miatt annak alkalmazását szükségtelennek ítéltük.

A referenciához képest legjelentősebb eltéréseket mutató régiók közül néhány megtekinthető a Függelékben található B.1. ábra kiemelt részein. Megfigyelhető, hogy módszerünk jól közelíti az elmosódás mértékét és a pontszórásfüggvény alakját minden vizsgált esetben. Ezenkívül fázoros magfüggvényünk kevesebb hibát generál a Gauss-függvényes módszerhez képest. Így tehát elmondható, hogy módszerünk alkalmas az alacsonyrendű aberrációk szimulációjára és növeli a pontosságot a korábban használt Gauss-függvénnyel szemben.

Tengelyen elhelyezkedő pontok szimulációja csempézett konvolúcióval

Módszerünk kiértékeléséhez a pontszórásfüggvényrácsot az $N_d = 33$, $N_A = 6$, $N_f = 7$ és $N_\lambda = 3$ paraméterekkel mintavételeztük, 4158 egyedi mintapontot eredményezve. GPU-alapú módszerünkkel a rács előállítása körülbelül 15 másodpercig tartott minden szemállapot esetén. Barsky módszerével [60] $N_d = 41$ mélységet és $N_\lambda = 3$ hullámhosszt alkalmaztunk. A folytonos pontszórásfüggvényű konvolúcióra épülő referenciamódszer 595 (*primitívek*) és 419 (*San Miguel*) mélységfüggő csoportot generált és körülbelül 11 perc számítást igényelt képenként. A kimeneteket a Függelékben található B.2. és B.3. ábra szemlélteti.

Először a képkockánkénti gyorsítótextúra méretét és feltöltési idejét vizsgáltuk meg az elmosási sugáron (*sugár*) és az objektumtávolságon (*mélység*) alapuló elrendezésünkkel. Eredményeinket az 1.8. táblázat foglalja össze. Jól látható, hogy mindkét megközelítés memóriaigénye és számítási ideje megfelelő a modern videokártyákon való alkalmazáshoz. Ezenkívül az is megállapítható, hogy az objektumtávolságon alapuló módszerünk nagyságrendekkel lecsökkenti a gyorsítótextúra méretét és a képkockánkénti újragenerálás idejét.

	Rétegek		Textúra		Interpoláció	
Állapot	**Sugár**	**Mélység**	**Sugár**	**Mélység**	**Sugár**	**Mélység**
Emmetrópia	1419	65	39,11 MB	1,79 MB	0,63 ms	0,26 ms
Miópia	957	65	11,86 MB	0,82 MB	0,26 ms	0,21 ms
Asztigmia	891	65	9,55 MB	0,71 MB	0,26 ms	0,17 ms
Keratokónusz	1320	65	31,43 MB	1,55 MB	0,45 ms	0,24 ms
Katarakta	957	65	11,86 MB	0,81 MB	0,38 ms	0,23 ms
LASIK műtét	1221	65	24,82 MB	1,32 MB	0,49 ms	0,19 ms

1.8. táblázat. Gyorsítótextúránk memóriaigénye és képkockánkénti előállítási költsége a két gyorsítótextúra-elrendezésünkkel.

Ezt követően a pontosság mérésére kiszámítottuk a Barsky módszerével és a csempézett konvolúciós eljárásunkkal generált kimenetekre a referenciához számított PSNR értékeket. Eredményeink az 1.9. táblázatban olvashatók.

		Primitívek			San Miguel		
Állapot	**Metrika**	**Korábbi [60]**	**Saját (Sugár)**	**Saját (Mélység)**	**Korábbi [60]**	**Saját (Sugár)**	**Saját (Mélység)**
Emmetrópia	Futási idő	21,93 s	24,68 ms	10,57 ms	13,41 s	16,13 ms	7,43 ms
	PSNR	31,92 dB	46,52 dB	46,43 dB	33,70 dB	46,29 dB	46,14 dB
Miópia	Futási idő	12,33 s	21,73 ms	9,53 ms	13,51 s	23,06 ms	8,66 ms
	PSNR	33,39 dB	46,80 dB	46,77 dB	33,52 dB	50,75 dB	50,60 dB
Asztigmia	Futási idő	11,98 s	26,51 ms	7,51 ms	13,29 s	30,62 ms	10,70 ms
	PSNR	33,61 dB	47,07 dB	47,09 dB	33,89 dB	49,35 dB	49,33 dB
Keratokónusz	Futási idő	17,10 s	27,08 ms	12,93 ms	14,62 s	24,66 ms	9,87 ms
	PSNR	33,64 dB	45,74 dB	45,77 dB	32,61 dB	48,91 dB	48,83 dB
Katarakta	Futási idő	15,53 s	24,50 ms	9,38 ms	15,22 s	17,09 ms	6,81 ms
	PSNR	31,53 dB	45,55 dB	45,53 dB	30,81 dB	45,51 dB	45,45 dB
LASIK műtét	Futási idő	16,46 s	21,29 ms	8,67 ms	11,86 s	20,60 ms	8,73 ms
	PSNR	31,07 dB	45,76 dB	45,73 dB	30,89 dB	48,10 dB	47,95 dB

1.9. táblázat. Futási idők és a referenciához viszonyított PSNR pontossági értékek a Barsky algoritmusával [60] és az új módszerünk két gyorsítótextúra-elrendezésével készített szimulációkhoz. Jól látható, hogy saját algoritmusunk szignifikánsan pontosabb a korábbi módszernél. Ezenkívül megközelítésünk rendelkezik a valósidejű rendszerek számára szükséges sebességgel, ami nem teljesül a korábbi módszerre.

Jól látható, hogy a módszerünk által generált látásszimulációk rendkívül magas pontosságot értek el a referenciaképekhez viszonyítva. Ennek tükrében elmondható, hogy a magfüggvény-approximációs eljárásunk hűen reprodukálja a folytonos, valós pontszórásfüggvényt. Gyakorlati kísérleteink alapján az eltérések többsége a bemeneti képpontok összevonásából fakad. Továbbá az is egyértelműen látható a generált hibákból, hogy a mélységalapú textúraelrendezés szignifikánsan redukált rétegszámai nem okoznak számottevő hibanövekedést. Ennek fő oka, hogy ez a megközelítésünk lényegében a magfüggvény interpolációjához nem használt adatok kiszűrésén alapszik.

Ezzel szemben Barsky módszere nem képes a képpontok közötti részleges takarási viszonyok helyes kezelésére és lényegesen kevesebb pontszórásfüggvény-információt alkalmaz. Mindezen okokból kifolyólag a módszer jelentős eltéréseket produkál, ahogyan azt a hibametrikák is tükrözik. A pontatlanságok egy része az elmosás mértékének eltéréseiben fedezhető fel, amelyek a helytelen magfüggvények alkalmazásából erednek. Ezenkívül a részleges takarási viszonyok figyelmen kívül hagyásából fakadóan az objektumok határainál szignifikáns hibák találhatók. A legnagyobb eltérést generáló területek közül néhány megtekinthető a Függelékben található B.2. és B.3. ábra kiemelt részein. Ezeken a területeken jól láthatók a Barsky módszerével készített látásszimulációk korlátai, illetve megfigyelhető a műtermékek hiánya az általunk bemutatott módszerrel generált kimeneteken. Végezetül a komplex fázoros kimenetekkel összevetve eredményeinket megfigyelhető a kromatikus aberráció szimulációjának jelentősége is, amit Cholewiak és mtsai. is kihangsúlyoztak [30, 31].

A pontosság számszerűsítése után megmértük a futási időket is, aminek eredményét szintén az 1.9. táblázatban tüntettem fel. Jól látható, hogy új módszerünk sikeresen eléri a valósidejű rendszerekhez szükséges szimulációs időt. Mindez nem mondható el a korábbi módszerről, amely minden esetben másodperceket igényelt egyetlen szimuláció előállításához. Továbbá az is egyértelműen megállapítható, hogy az objektumtávolság-alapú textúraelrendezés konzisztensen gyorsabb az elmosási sugárra épülő módszernél. Ehhez az alacsonyabb memóriaigény és a drasztikusan lecsökkent mintaszám is számottevően hozzájárul.

Periférikus látás szimulációja csempézett konvolúcióval

A periférikus látás szimulációjához szükséges magfüggvények száma miatt két eltérő felhasználási módot is megvizsgáltunk:

– *Precíz* mód: az $N_h = 31$, $N_v = 21$, $N_d = 9$, $N_A = 1$, $N_f = 1$ és $N_\lambda = 3$ paraméterekkel mintavételezett rács 17 577 magfüggvényt tartalmazott és körülbelül két perc számítást igényelt. A gyorsítótextúra átmeneti rétegeinek számát az (1.43) egyenlettel és az empirikusan megválasztott $s = 1$ és $p = 1/2$ paraméterekkel határoztuk meg.
– *Dinamikus* mód: az $N_h = 23$, $N_v = 13$, $N_d = 9$, $N_A = 5$, $N_f = 5$ és $N_\lambda = 3$ paraméterekkel mintavételezett pontszórásfüggvényrács 201 825 pontszórásfüggvényt tartalmazott és körülbelül 20 perces számítást eredményezett. A módszer ismertetésekor tárgyalt okok miatt nem alkalmaztunk átmeneti rétegeket a gyorsítótextúrában.

A korábbi módszerek esetén Rodríguez Celaya és mtsai. [61] algoritmusához az $N_h = 5$, $N_v = 3$, $N_d = 2$ és $N_\lambda = 3$ értékeket, Gonzalez Utrera módszeréhez [23] pedig az $N_h = 11$, $N_v = 11$, $N_d = 3$ és $N_\lambda = 3$ értékeket használtuk. A folytonos pontszórásfüggvényeket alkalmazó referenciamódszer összesen 194 243 (*primitívek*) és 127 548 (*San Miguel*) osztályt azonosított, a teljes látásszimuláció pedig több órát vett igénybe. A szimulált kimeneteket a Függelékben található B.4. és B.5. ábra szemlélteti.

Kiértékelésünket ismét a képkockánkénti gyorsítótextúra vizsgálatával kezdtük, aminek eredményét az 1.10. táblázat foglalja össze.

	Precíz		**Dinamikus**		
Állapot	**Rétegek**	**Textúra**	**Gyorsítótár**	**Textúra**	**Interpoláció**
Emmetrópia	$51 \times 27 \times 17$	1,54 GB	5,71 GB	234,06 MB	1,94 ms
Miópia	$50 \times 26 \times 17$	1,13 GB	4,70 GB	192,67 MB	1,90 ms
Asztigmia	$50 \times 26 \times 17$	1,13 GB	4,43 GB	181,58 MB	1,93 ms
Keratokónusz	$55 \times 27 \times 17$	2,14 GB	7,68 GB	314,38 MB	2,43 ms
Katarakta	$37 \times 30 \times 17$	0,46 GB	3,09 GB	85,01 MB	1,36 ms
LASIK műtét	$56 \times 32 \times 17$	2,32 GB	6,50 GB	266,08 MB	2,12 ms

1.10. táblázat. Gyorsítótextúránk memóriaigénye és képkockánkénti előállítási költsége a két vizsgált konfigurációval.

A *precíz* mód egyedüli költsége a gyorsítótextúra memóriaigénye, amely a *dinamikus* módban az átmeneti rétegek hiányánál fogva lényegesen alacsonyabb. Továbbá, mivel a szemparaméterek nem változnak képkockánként, így a *precíz* konfigurációval a gyorsítótextúra az előfeldolgozás során előállítható. Ezzel szemben a *dinamikus* módban az előfeldolgozáskor a pupillaátmérő és a fókusztá-

volság kombinációihoz létrehozott gyorsítótár is memóriát igényel, a gyorsítótextúrát pedig képkockánként szükséges újragenerálni. Jól látható azonban, hogy a rétegcsökkentő stratégiánknak köszönhetően a memóriaigény és a gyorsítótextúra előállítási ideje is tökéletesen alkalmas módszerünk modern grafikus kártyákon való végrehajtására.

Ezt követően megmértük a vizsgált módszerek futási idejét minden teszteset-re, aminek eredményét az 1.11. táblázat foglalja össze.

		Primitívek				San Miguel			
Állapot	**Metrika**	**Korábbi [61]**	**Korábbi [23]**	**Saját (Prec.)**	**Saját (Dinam.)**	**Korábbi [61]**	**Korábbi [23]**	**Saját (Prec.)**	**Saját (Dinam.)**
Emmetrópia	Futási idő	1,70 h	4,70 h	20,92 ms	22,81 ms	1,70 h	4,60 h	14,34 ms	14,48 ms
	PSNR	33,12 dB	31,37 dB	45,59 dB	45,01 dB	30,28 dB	31,31 dB	46,35 dB	45,06 dB
Miópia	Futási idő	1,70 h	3,50 h	14,71 ms	16,18 ms	1,70 h	3,60 h	11,74 ms	11,92 ms
	PSNR	32,62 dB	31,11 dB	45,06 dB	43,88 dB	30,26 dB	30,81 dB	43,77 dB	41,96 dB
Asztigmia	Futási idő	1,80 h	3,70 h	13,48 ms	14,85 ms	1,90 h	3,60 h	11,77 ms	11,65 ms
	PSNR	31,80 dB	29,65 dB	45,39 dB	44,54 dB	29,67 dB	31,49 dB	43,90 dB	42,03 dB
Keratokónusz	Futási idő	1,80 h	5,90 h	19,56 ms	20,14 ms	1,80 h	5,70 h	14,32 ms	14,02 ms
	PSNR	36,18 dB	32,13 dB	44,25 dB	43,64 dB	29,78 dB	32,22 dB	44,23 dB	42,19 dB
Katarakta	Futási idő	1,00 h	1,80 h	13,37 ms	12,81 ms	1,00 h	1,80 h	10,67 ms	11,31 ms
	PSNR	35,91 dB	33,82 dB	44,81 dB	44,35 dB	32,02 dB	34,48 dB	44,76 dB	44,17 dB
LASIK műtét	Futási idő	1,60 h	4,80 h	24,44 ms	24,85 ms	1,60 h	5,70 h	20,56 ms	19,13 ms
	PSNR	36,32 dB	31,23 dB	44,32 dB	43,16 dB	29,21 dB	32,62 dB	44,85 dB	42,16 dB

1.11. táblázat. Futási idők és a referenciához viszonyított PSNR pontossági értékek a Rodríguez Celaya és mtsai. eljárásával [61], a Gonzalez Utrera algoritmusával [23], valamint a saját módszerünk két konfigurációjával készített szimulációkhoz. Jól látható, hogy a csempézett konvolúcióra épülő eljárásunk jelentősen pontosabb mindkét korábbi módszernél. Továbbá a létező eljárások több órás futási idejével ellentétben algoritmusunk teljesítménye megfelelő a valósidejű alkalmazások számára.

Jól látható, hogy a korábbi módszerek több órás számítási ideje messze meghaladja az interaktív rendszerek számára alkalmas áteresztőképességet minden vizsgált tesztesetben. Ezzel szemben a csempézett konvolúción alapuló módszerünk a perifériás látás szimulációja esetén is konzisztensen megfelelő valós felhasználások számára. Ezenkívül módszerünk a pupillaátmérő és a fókusztávolság dinamikus változtathatóságát is lehetővé teszi. Az algoritmusunk sebességét érintő legfőbb eltérés a két vizsgált tesztjelenet között észlelhető. A látásszimuláció jellemzően tovább tartott a *primitívek* esetén, aminek oka vélhetően a magfüggvény interpolációjához szükséges memóriatranzakciókban keresendő.

A sebesség elemzését követően megvizsgáltuk a szimuláció pontosságát is. A pontosságot számszerűsítő, referenciához viszonyított PSNR értékek az 1.11. táblázatban olvashatók. Jól látható, hogy új módszerünk mindkét konfigurációja rendkívül magas pontosságot ért el minden vizsgált tesztesetben és kiválóan teljesíti a valós rendszerek számára szükséges szimulációs minőséget. Ezenkívül, ahogyan az elvárható, a sűrűbben mintavételezett magfüggvényrács és az átmeneti rétegek alkalmazása miatt a *precíz* mód konzisztensen alacsonyabb hibamértéket eredményez, mint a *dinamikus* konfiguráció. A korábbi két konvolúciós algoritmushoz képest módszerünk szignifikánsan magasabb pontosságot mutat minden vizsgált tesztesetben. Mindez elsősorban a módszerünk által használt pontszórásfüggvények számottevően magasabb darabszámának, illetve a képpontok közötti részleges takarási viszonyok helyes kezelésének köszönhető.

A legnagyobb eltérést mutató részek közül néhány régió megtekinthető a Függelékben található B.4. és B.5. ábra kiemelt részein. Jól látható, hogy módszerünk hűen reprodukálja a mérési alapként szolgáló folytonos pontszórásfüggvényt ezekben a régiókban is. Ezenkívül algoritmusunk pontosan közelíti a valós pontszórásfüggvény dinamikusan változó alakját és méretét is. Ily módon módszerünk az olyan jelenségeket is hitelesen szimulálja, mint a kromatikus aberráció és a relatív perifériás hiperópia. Ezzel szemben a korábbi módszerek esetén a pontszórásfüggvény rendkívül ritka mintavételezéséből fakadóan a mélységtől és a beesési iránytól függő magfüggvény kiterjedése és alakja is jelentősen sérül. Ezáltal a korábbi módszerek alkalmatlanok a szimulált szem látásának pontos jellemzésére. Elmondható tehát, hogy az általunk javasolt interpolációs stratégia tökéletesen alkalmas a perifériás magfüggvények approximációjára és nagymértékben növeli a szimuláció minőségét a létező eljárásokhoz képest.

2. Lencsefényfoltok tervezését és renderelését támogató eszközök fejlesztése

A kamerák lencsefényfoltjai alapvető eszköznek számítanak a fotográfia és a cinematográfia területén, így azok gyors és valósághű szimulációja kardinális szempont a számítógéppel generált szintetikus képek esetén [37–39]. Az értekezés ezen részében az optikai rendszerek által generált lencsefényfoltok hatékony tervezésére és valósidejű megjelenítésére megalkotott eszközeinket ismertetem.

2.1. Szakmai előzmények, motiváció

2.1.1. Algoritmusok lencsefényfoltok renderelésére

A lencsefényfoltok keletkezésükből fakadóan két komponensből tevődnek össze, amelyek pontos szimulációja merőben eltérő megközelítést igényel. Mindkét elem közös jellemzője, hogy megjelenésüket a magas intenzitású fényforrások okozzák. A lencsefényfoltok egyik fő komponense a *becsillanási fényfolt* (angolul tipikusan *glare* vagy *starburst pattern* néven hivatkoznak rá), amelyet az optikai rendszer rekesznyílásán áthaladó fény diffrakciója okoz, és a fényforrás képe körül jelenik meg. A jelenség másik fő elemét pedig a *szellemek* (angolul *ghosts*) képezik, amelyek a beérkező fény többszöri visszaverődése által generált, jellemzően rekesznyílás-formájú színes foltok a képen.

Empirikus módszerek

A lencsefényfoltok renderelésének egyik legkorábbi megközelítése textúraalapú kompozíción alapszik, amely során előre generált textúrákat helyeznek el a bemeneti képen. A textúrák létrehozása és azok képpel való kompozíciója is empirikus módon történik, a tényleges kamerarendszer optikai felépítésének és tulajdonságainak figyelembevétele nélkül. Ilyen megoldást először Kilgard javasolt [131], aki egy, a képernyő közepén áthaladó egyenes mentén helyezte el a szellemeket. Ezt követően King a szellemfoltok intenzitását és áttetszőségét a folt képcentrumtól való távolsága alapján határozta meg [132]. Maughan az intenzitásértékek pontosabb kiszámítását a képen a fényforráshoz tartozó, látható és takart képpontok számának aránya alapján végezte el [133], amihez később Sekulic egy

GPU-alapú megoldást is demonstrált [134]. Ezeknek a módszereknek a fő előnye az egyszerűségben és a futási időben rejlik, ugyanis a szimulációhoz használt áttetsző négyzetek megjelenítése triviális és alacsony számítási költséggel jár. Mivel azonban a szimuláció semmilyen fizikai alappal nem rendelkezik, így ezek a módszerek csak erősen korlátolt feladatokra használhatók.

Textúrák helyett Oat egy utófeldolgozó szűrővel szimulálta a becsillanási fényfoltokat a kép magas intenzitású pontjai körül [135], Alspach pedig egy vektoros reprezentációt alkalmazott az egyes lencsefényfolt-elemek testreszabható modellezésére [136]. A két eljárás fő előnye, hogy a textúraalapú kompozícióval szemben nagyobb tervezői szabadságot tesz lehetővé. Az algoritmusok által használt megközelítéssel azonban továbbra is csak a fényfoltok empirikus szimulációja lehetséges, ami erősen korlátozza az eljárások felhasználási területeit.

Fizikai alapú eljárások

Az optikai rendszer fizikai struktúrájának felhasználásával végzett képszintézis a számítógépes grafika számos területén képes a hitelesség fokozására. Ennélfogva a tudományos közösség nagy energiát fektet a hatékony és pontos, fizikai alapú eljárások megalkotásába [130, 137–140].

A lencsefényfoltok fizikai alapú szimulációját először Chaumond demonstrálta [141], aki a lencsefényfoltok megjelenítésére fénykövetést alkalmazott egy valós kameramodell fénytörő felületeivel. Ezt követően Keshmirian fotontérképeket [142], Steinert és mtsai. pedig Monte Carlo integrálást használtak lencsefényfoltok szimulációjára [143]. Lendermann és mtsai. a diffrakció pontos kezelésével lehetővé tette a tetszőleges rekesznyíláshoz és fókusztávolsághoz tartozó becsillanási fényfoltok fizikailag helyes szimulációját [144]. Ezeknek a módszereknek a fő előnye a fizikai alapú szimulációban rejlik, ugyanis a generált kimenetek lényegesen pontosabban modellezik a jelenséget, mint az empirikusan elhelyezett textúrák. Mindez a megjelenített lencsefényfoltok számában, elhelyezkedésében és alakjában is megfigyelhető. A módszerek fő hátránya a számítási idő, ami a nagyméretű optikai rendszerek költséges sugárkövetéséből és a felhasznált sugarak jellemzően nagy számából fakad. Ezenkívül a sugarak tipikusan véletlen mintavételezésen alapulnak, jól látható zajt generálva.

Az optikai módszerek helyett Walch és mtsai. digitális felvételek feldolgozását javasolták [145]. Módszerük a szimulált kamerával készített képek lencsefényfoltjainak megkeresésén alapszik, amelyekre Bézier-görbék darabjaiból

felépített felületeket illesztenek. Ezt követően a görbékből készített poligonrácsok raszterizációjával képesek a kamera összes lencsefényfoltját szimulálni a fényforrás tetszőleges elhelyezkedése esetén. A módszer nagy előnye, hogy a kamera birtokában annak fizikai paraméterei nélkül is szimulálhatók az optikai rendszer fényfoltjai. Ezenkívül a módszer számítási költsége is alacsonyabb, és a raszterizációs megközelítés miatt a mintavételezési zaj is elkerülhető. Algoritmusuk azonban számottevően csökkenti a szimuláció pontosságát, ugyanis a Bézier-görbék simasága és egyszerűsége alkalmatlan a valós fényfoltokra jellemző komplex alakzatok modellezésére. Ezenkívül eljárásuk egy költséges előfeldolgozási lépést is igényel.

Valósidejű módszerek

Habár a fent leírt algoritmusok képesek a fényfoltok valósághű megjelenítésére, a szimulációhoz használt technikák rendkívül számításigényesek. Hullin és mtsai. ritka sugárrácsokat alkalmaztak a szellemfoltok valósidejű, fizikai alapú renderelésére [41]. Módszerük szellemenként egy kis elemszámú, hézagos sugárrácsot követ végig a rendszeren a szenzorig, amelyből GPU-alapú raszterizációval állítja elő a kimenetet. A sugárrács kis méretének köszönhetően a módszer drasztikusan csökkenti a renderelési időt, ami az alacsony komplexitású kamerák esetén akár valósidejű szimulációt is lehetővé tesz. Az analitikus sugárkövetés és a szellemek raszterizációjának költsége azonban erősen korlátozza a módszer sebességét nagyméretű optikai rendszerek esetén. Ezenkívül a megfelelő minőség és teljesítmény eléréséhez elengedhetetlen egy költséges előfeldolgozási lépés és a szimulációs paraméterek helyes megválasztása.

A sugárkövetéshez szükséges metszéspontok gyorsabb kiszámítására Hullin és mtsai. Taylor-polinomok használatát javasolták [146]. Bár ezt a megközelítést Pekkarinen és mtsai. a gyakorlatban is alkalmazta [39], a lencsefényfoltszimuláció ily módon való elvégzésének pontos jellemzőit (pontosság, futási idő stb.) a szerzők nem értékelték ki. Ennélfogva a polinomokra épülő megközelítés valósidejű, interaktív rendszerekben való felhasználhatósága kérdéses. A szellemfoltok raszterizációs költségének csökkentésére pedig a közelmúltban Bodonyi és Kunkli [147] egy csempealapú megoldást javasolt. Módszerükkel nagyszámú szellemfoltok is hatékonyan megjeleníthetők, ily módon komplex optikai rendszerek és nagy mennyiségű fényforrások is szimulálhatók.

A szimuláció sebességének további drasztikus növelésére Lee és Eisemann

paraxiális sugárkövetést alkalmazott [148]. Módszerük fő gondolata, hogy transzfermátrixok segítségével gyorsan meghatározható a szellemfoltok helye, mérete és intenzitása. Ezekből az értékekből a szellemek képe textúraalapú módon gyorsan közelíthető. Az eljárás fő hátránya, hogy bonyolultabb fényfoltalakzatok ily módon nem jeleníthetők meg. Ahogyan azt azonban munkájukban demonstrálták, a módszer ennek ellenére is nagy pontossággal és lényegesen rövidebb számítási idővel közelíti a sugárkövetéses algoritmusok kimenetét.

Textúrák fizikailag helyes előállítása

Az előző részben ismertetett módszerek jellemzően textúrákat alkalmaznak a diffrakció kezelésére, amely a becsillanási fényfoltok és szellemfoltok alakzatát is befolyásolja. A textúrák fizikai alapú előállítása nem csak a szóban forgó algoritmusok számára kulcsfontosságú, de ily módon az empirikus megközelítések pontossága is szignifikánsan növelhető.

A diffrakció szellemfoltokra gyakorolt hatását Hullin és mtsai. a rekesznyílás képének frakcionális Fourier-transzformációjával [41] modellezte. Bár ily módon a szellemfoltok széleinek mintázata jól közelíthető, a lencsék egyenetlenségeiből fakadó jellemzők nem kezelhetők. A szellemfoltok képtérbeli mérete számos esetben kellően nagy ahhoz, hogy ezek az egyenetlenségek jól látható mintázatokat eredményezzenek. A probléma megoldására Joo és mtsai. sugárkövetést és az egyenetlenségek textúraalapú modellezését alkalmazta [149], nagymértékben növelve a szellemfoltok mintázatának hitelességét.

A kamerarendszerek becsillanási fényfoltjai az emberi szemben erős fény hatására fellépő fényfoltokhoz hasonlóan keletkeznek. Mindkét esetben az optikai rendszer rekesznyílása (az emberi szem esetén a pupilla) által generált fényelhajlás okozza a mintázatokat, éppen ezért szimulációjuk is tipikusan hasonló eszközök segítségével végezhető el. Az emberi szem becsillanási fényfoltjait szimuláló textúrák létrehozására Kakimoto és mtsai. Fraunhofer-diffrakciót alkalmaztak [150], amellyel a pupilla által okozott fényelhajlás modellezhető. A pontosság növelésére van den Berg és mtsai. a szemben lévő apró molekulák hullámhossztól függő fényszórását modellezte [151]. Mindkét megközelítés esetén fontos korlátozó tényező, hogy a fényfolt képtérbeli méretét figyelmen kívül hagyja. A probléma megoldására Ritschel és mtsai. megvalósították a fényfoltok dinamikus, pupillaátmérőtől függő méretezését, illetve a szemben található molekulák fényszórását is pontosabban modellezték, mint a korábbi módszerek [152]. Vége-

zetül az eddig felsorolt eljárások költsége nem teszi lehetővé a textúrák dinamikus, valósidejű előállítását, aminek megoldására Scandolo és mtsai. egy hierarchikus megoldást javasoltak a Fraunhofer-diffrakció gyors kiszámításához [153]. Módszerükkel a becsillanási fényfoltok textúrája dinamikusan, valós időben is előállítható, ily módon az optikai rendszer felépítésének változása, illetve a fényforrások dinamikus takarási viszonya is modellezhető.

2.1.2. Motiváció

A speciális effektek renderelését támogató programkönyvtáraknak hangsúlyos szerepe van a valós alkalmazásokban [154, 155]. Bár léteznek a gyakorlatban is használt optikai szimulációs szoftverek [74, 156], azok nem biztosítják a lencsefényfoltok hatékony rendereléséhez szükséges eszközöket. A jelenség fizikai alapú szimulációjának nehézségei miatt célom egy használatra kész programkönyvtár elkészítése volt a fényfoltok valósidejű szimulációjára. Ehhez létrehoztam egy nyílt forráskódú keretrendszert *OpenLensFlare* néven [157], amely egy tervező- és egy renderelőmodul segítségével valósítja meg a kitűzött célokat.

Ezenkívül Hullin és mtsai. lencsefényfolt-renderelő algoritmusát [41] annak sebessége és pontossága miatt a gyakorlatban is rendszeresen alkalmazzák [39]. Éppen ezért az algoritmusnak kardinális szerepe van a keretrendszerem által célzott valósidejű applikációk számára. Az eljárás egyik fő negatívuma a minőség drámai romlásának elkerüléséhez szükséges költséges előfeldolgozó lépés. Az előszámító lépés kritikus a kamerarendszer lencsefényfoltjainak elemzéséhez, ugyanakkor az algoritmus részét képező sugárkövetéses megközelítés nem alkalmas a módosítások hatásának interaktív kiértékelésére, nagymértékben nehezítve a jelenség tervezésének folyamatát. A probléma orvoslására megalkottunk egy gyors, a folyamat eredményét interaktív módon approximáló eljárást [158].

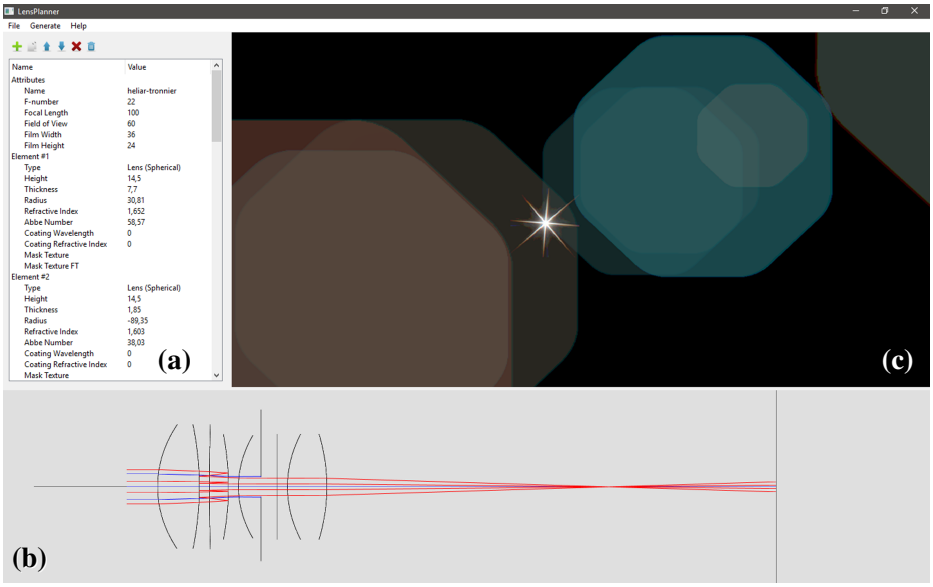
2.2. Saját, nyílt forráskódú programkönyvtár

2.2.1. A programkönyvtár tervezési részletei

A lencsefényfoltok fizikai alapú szimulációjának fő integrációs nehézsége az optikai rendszer modellezésében és a modellel történő hatékony sugárkövetésben rejlik. A kameraobjektív fizikai paramétereinek beszerzése, megértése és szerkesztése mind nehéz feladat, amelyek távol állnak a lencsefényfoltok szimulációját implementáló és megjelenését tervező programozók és művészek feladatkö-

rétől. Programkönyvtáram elkészítése során ezen problémák megoldására törekedtem. Rendszerem egy olyan integrált megoldást nyújt, amellyel a szükséges adatok könnyen kezelhetők, a lencsefényfoltok pedig hatékonyan szimulálhatók.

A kitűzött célok eléréséhez a rendszert két fő modulból építettem fel. Egyrészt a programkönyvtáram tartalmaz egy futásidejű modult, amely a felhasználó rendelkezésére bocsájtja az optikai rendszerek modellezéséhez szükséges elemeket és a fényfoltokat megjelenítő algoritmusok implementációját. A rendszerem másik fő komponense a futásidejű modulra épülő szerkesztőfelület, amellyel a kamerarendszer paraméterei szerkeszthetők, a modellezett rendszer sematikus rajza vizualizálható, a rendszer által generált lencsefényfoltok pedig megjeleníthetők. A szerkesztőmodulról egy pillanatképet a 2.1. ábra szemléltet. Eredményeimet a továbbiakban a 2017-ben megjelent publikációm [157] alapján ismertetem.



2.1. ábra. Pillanatkép az OpenLensFlare szerkesztőmoduljáról, amellyel a kamerarendszer szerkeszthető (a) és vizualizálható (b), a lencsefényfoltok pedig szimulálhatók (c).

Optikai rendszer modellezése

Ahogyan a 2.1.1. alszakaszban kiemeltem, a lencsefényfoltok fizikai alapú szimulációjához szükség van az optikai rendszer paramétereire. Ezek a paraméterek jellemzően az egyes fénytörő felületek tulajdonságaiból (törésmutató, felület típusa, fizikai magassága, görbületi sugara stb.), valamint a teljes optikai rendszer

jellemzőiből (a rendszer gyújtótávolsága és a rekesznyílásának átmérője, a szenzor fizikai mérete stb.) tevődnek össze. Mindezen értékek beszerzése történhet például könyvekből [159, 160] és az objektívek szabadalmi dokumentumaiból. A paraméterek módosítása hasznos lehet a generált lencsefényfoltok kinézetének megváltoztatásához, ám azok kimenetre gyakorolt hatása nehezen kiszámítható. Éppen ezért a tervezői modul dedikált komponenseket tartalmaz a lencseparaméterek módosítására, az optikai rendszer sematikus rajzának megtekintésére, valamint a rendszer által generált lencsefényfoltok valósidejű megjelenítésére.

A szükséges adatok kezelésére a futásidejű modul egy speciális objektumot tartalmaz, amelyben az optikai rendszerhez és az annak fénytörő felületeihez tartozó adatok hatékonyan tárolhatók. Az objektumban lévő értékek módosítása a szerkesztőmodul dedikált felületén végezhető, ahol a rendszer globális és lencseszintű paraméterei is szerkeszthetők. A szóban forgó felület a 2.1. ábra (a)-val jelölt komponensén tekinthető meg futás közben. A tervezői modul segítségével a felhasználó képes azonnali visszacsatolást szerezni a módosítások lencsefényfoltokra gyakorolt hatásáról, drasztikusan megkönnyítve a kívánt megjelenésű fényfoltokat generáló kameraparaméterek előállítását.

Fényfoltok renderelése

Az optikai modellparaméterek birtokában elvégezhető a lencsefényfoltok szimulációja. Erre a célra a keretrendszerem futásidejű modulja biztosítja a szükséges algoritmusokat. A modulban koncepcionális szinten is elkülönülnek a becsillanási fényfoltokhoz tartozó textúrák előállítását és megjelenítését, illetve a szellemek renderelését végző eszközök. Ily módon a felhasználó az eltérő tulajdonságokkal rendelkező eljárások kombinációjával pontosabban megválaszthatja a szimulációhoz alkalmazott technikákat.

Az implementált algoritmusokat illetően rendszerem az értekezés írásakor a becsillanási fényfoltok szimulációjára a Fraunhofer-diffrakción alapuló eljárást tartalmazza, amelyet Kakimoto és mtsai. [150], Ritschel és mtsai. [152], valamint Hullin és mtsai. [41] is alkalmaztak munkájuk során. A szellemfoltok szimulációjára pedig rendszerem Hullin és mtsai. [41] sugárkövetéses módszerét, illetve Lee és Eisemann [148] transzfermátrixokra épülő algoritmusát biztosítja. A szimuláció finomhangolhatóságának érdekében az implementációk elkészítése során nagy hangsúlyt kapott az algoritmusok megfelelő paraméterezhetősége.

Adatok generálása és tárolása

Az OpenLensFlare-ben implementált algoritmusok többségéhez esszenciális egy előfeldolgozó szakasz. Ennek célja lehet például a diffrakciós textúrák előállítása, a transzfermátrixok kiszámítása, illetve a sugárkövetéses eljárás szellemenkénti szimulációs paramétereinek meghatározása. Bár ezek a lépések gyakran kellően rövidek az alkalmazás inicializációja során való elvégzéshez, bizonyos esetekben az előfeldolgozás órákat is igénybe vehet, szükségessé téve a számított adatok tárolását és újrafelhasználását.

Keretrendszerem tervezése során meghatározó szempont volt az adatgenerálási lépés szeparációjának és az adatok újrafelhasználhatóságának támogatása. Minden implementációs elem oly módon készült el, hogy a generált adatok teljesen transzparensen hozzáférhetők legyenek. Ezáltal a szimulációhoz szükséges adatok (optikai rendszer paraméterei, becsillanási fényfolt textúrája, szellemek szimulációs paraméterei stb.) előre elkészíthetők és eltárolhatók, majd futás előtt a befoglaló szoftver által egyszerűen betölthetők és felhasználhatók. Továbbá az adatok szerializációja is elvégezhető tetszőleges módon.

Ezenkívül a tervezőmodul tartalmazza a szóban forgó adatok mentésének és betöltésének referenciaimplementációját, amellyel az adatok elmenthetők képek és XML fájlok halmazaként. Ezen XML fájlok egyikének struktúráját a 2.1. kódrészlet demonstrálja. A keretrendszerben található referenciaimplementáció használatra kész megoldást biztosít a feladat elvégzésére és kiindulási alapként szolgál az alkalmazás számára megfelelő saját megoldás elkészítésére.

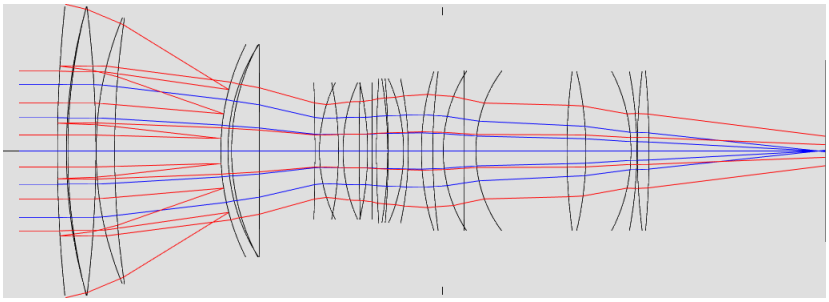
```
<?xml version="1.0" encoding="UTF-8"?>
<opticalSystem>
  <name>Példa kamerarendszer</name>
  <fnumber>11</fnumber>
  <!-- További kameraparaméterek... -->
  <elements>
    <element>
      <type>lensSpherical</type>
      <radius>65.220001</radius>
      <!-- További lencseparaméterek... -->
    </element>
    <!-- További lencsék... -->
  </elements>
</opticalSystem>
```

2.1. kódrészlet. Példa az optikai rendszer paramétereit tartalmazó, a keretrendszerben található referenciaimplementációval generált XML fájl struktúrájára.

Vizualizációs eszközök

Az adatok vizualizációja kulcsfontosságú a tudomány számos területén. Segítségével egy probléma megértésének ideje és nehézsége drasztikusan csökkenthető, és az adatok vizuális reprezentációja nélkül nehezen hozzáférhető tudás kinyerése is lehetségessé válhat. Éppen ezért elengedhetetlennek ítéltem a megfelelő adatvizualizációs eszközök elérhetővé tételét az OpenLensFlare szerkesztői modulja segítségével.

Egy kamerarendszer viselkedésének megértése pusztán annak paraméterei alapján még szakértő lencsetervezők számára is nehéz feladat. Az ehhez szükséges szakterület-specifikus tudással az OpenLensFlare potenciális felhasználóinak zöme (művészek és programozók) feltehetően nem rendelkezik. A probléma megoldására a tervezői modul egy valós időben frissülő, kétdimenziós sematikus rajzot biztosít a szerkesztett optikai rendszerről. Mivel az optikai rendszer leírásából hiányoznak az elemek összekötéséhez szükséges adatok, így a diagram elkészítéséhez csak körívek és vonalak alkalmazhatók. Ahogyan azonban azt a 2.2. ábra szemlélteti, a rendszer így is képes hasznos és jól értelmezhető rajzok előállítására.



2.2. ábra. Példa az OpenLensFlare által készített vizualizációra. A lencserendszer sematikus rajzán a visszaverődés nélküli sugarak a képhez tartozó útvonalat (kék), a két visszaverődést tartalmazó sugarak pedig egy potenciális szellem útját jelenítik meg (piros).

A kamerarendszer által generált szellemek létrejöttének megértéséhez az OpenLensFlare képes a sematikus rajzon szellemútvonalak megjelenítésére is. Ehhez az adott számú visszaverődéssel rendelkező útvonalakból (amely visszaverődések száma akár nulla is lehet a képhez tartozó természetes útvonal esetén) bármelyik megjeleníthető a felhasználó által konfigurálható paraméterekkel (sugarak száma, beesési szög és magasság, szín stb.). Ezenkívül az eltérő útvonalak

összehasonlításához akár több ilyen vizualizáció is elhelyezhető egyidejűleg. Ily módon a tervező mélyebb betekintést nyerhet a szellemútvonalak természetébe, és az így szerzett tudás birtokában módosíthatja az optikai rendszer paramétereit. Változtatásainak lencsefényfoltokra gyakorolt hatását pedig a vizualizációs eszközöknek hála azonnal meg is tekintheti, ezáltal megteremtve a kívánt eredményhez vezető visszacsatolási hurkot. A 2.2. ábra ilyen útvonalakat is tartalmaz, ezáltal szemléltetve az eszközt működés közben.

Egy további fontos vizualizációs eszköz a lencsefényfoltok azonnali megtekintésének lehetősége, hiszen a tervezés során kiemelt cél a kívánt fényfoltkinézetek elérése. Erre a célra a szerkesztői modul a futásidejű komponenst alkalmazza a szerkesztett kamerarendszerrel szimulált lencsefényfoltok folyamatos megjelenítésére. Ennek köszönhetően a felhasználó számára garantálható, hogy a tervezés során látott vizualizáció teljes mértékben megegyezik a befoglaló alkalmazásban a futásidejű modul segítségével készített szimulációval. Ezenkívül a rendszer képes a szimuláció során generált láthatatlan adatok (mint például a rekesznyíláson áthaladás koordinátái és a sugár koordinátái az első lencsén) megjelenítésére is. Ezeknek a vizualizációknak a létrehozásáért szintén a futásidejű komponens felel. Mindez lehetővé teszi egy dedikált hibakereső mód elhelyezését a befoglaló rendszerben, amely módban a szerkesztővel előállított vizualizációkkal konzisztens kimenetek tekinthetők meg.

2.2.2. Implementációs megfontolások

Programozási interfész tervezése

A programkönyvtár tervezésekor hangsúlyos szempont volt a kiterjeszthetőség. A legfőbb koncepciókat (felületi elem, optikai rendszer, fényforrás, szellemútvonal, fényfoltszimuláló algoritmus stb.) saját osztályok reprezentálnak a futásidejű modulban. Ily módon a szóban forgó osztályok specializációjával és kompozíciójával a kívánt modularitás és bővíthetőség hatékonyan megvalósítható.

A könyvtár által biztosított szimulációs algoritmusok implementációja szintén a futásidejű modulban található. Ennek köszönhetően a tervezői modul és a befoglaló szoftver is könnyen és konzisztensen képes alkalmazni ezeket az algoritmusokat. Ezenkívül a rendszer saját renderelő algoritmusokkal is egyszerűen kiterjeszthető, amelyek implementációjához a könyvtárban található kódbázis hasznos kiindulási pontként szolgál. A programkönyvtár interfészének használatát a 2.2. kódrészlet szemlélteti.

```
// Lencserendszer betöltése (az applikáció implementálja)
OpticalSystem* system = loadOpticalSystem();
// Diffrakciós algoritmus a becsillanási fényfoltokhoz
auto starburstRenderer = new DiffractionStarburstAlgorithm(system);
// 512x512 méretű becsillanási textúra előállítása, amely a
// [380 nm, 780 nm] intervallumot egy 5 nm-es lépésközzel járja be
starburstRenderer->generateTexture({ 512, 512, 380f, 780f, 5f });
// Sugárkövetéses algoritmus a szellemek megjelenítésére
auto ghostRenderer = new RayTraceGhostAlgorithm(system);
// Optimális renderelési paraméterek kiszámítása a kamera szellemeihez
ghostRenderer->computeGhostAttributes(GhostList(system));
```

2.2. kódrészlet. Példa C++ kódrészlet, amely a rendereléshez használt objektumok létrehozását és inicializációját demonstrálja.

Grafikus programkönyvtár megválasztása

A grafikus programozási interfész megválasztása kardinális a futásidejű komponens számára, hiszen az nagymértékben befolyásolhatja a könyvtár befoglaló szoftverekben való alkalmazhatóságát. Mivel az OpenGL grafikus könyvtár [115] széles körben támogatott és interoperábilis más grafikus programkönyvtárakkal, így kézenfekvő választás volt a rendszer renderelő eszközeinek megvalósítására. Ennek következtében minden algoritmus OpenGL objektumokat alkalmaz bemenetként, és egy megfelelően konfigurált OpenGL kontextus meglétét feltételezi a helyes működéshez.

A kitűzött célok elérésének egy másik módja egy dedikált absztrakciós réteg alkalmazása a grafikus programozási interfész eszközeinek elrejtésére. Ez a megközelítés nem ritka grafikus könyvtárakat alkalmazó nagyméretű környezetekben. Rendszeremhez hasonlóan azonban a speciális effektek renderelését célzó, létező programkönyvtárak [161, 162] is tipikusan egy adott grafikus könyvtárra épülnek. Ennek legfőbb okait az absztrakciós réteg futásidejű felára, a megtervezésének komplexitása, illetve az egyes grafikus könyvtárakhoz tartozó modulok elkészítésének költsége jelentik. Ezenkívül harmadik fél által készített absztrakciós programkönyvtárak is rendelkezésre állnak (például a *bgfx* csomag [163]). Az OpenLensFlare tervezésekor elérhető modulok mindegyike tartalmazott azonban olyan korlátozó tényezőt, amely drasztikusan megnehezítette annak keretrendszeremben való felhasználását. Ezenkívül egy absztrakciós programkönyvtár a befoglaló rendszer számára is egy lényeges extra függőséget jelent és rendelkezik az absztrakciós rétegek fent ismertetett problémáival.

Grafikus árnyalók

Habár az OpenLensFlare által használt külső erőforrások túlnyomó része konfigurálható, a grafikus árnyalók kezelése speciális figyelmet igényel. Mivel az árnyalók kódját a grafikus programkönyvtár egyetlen karaktersorozatként várja, így azokat gyakran beágyazzák szöveges literálként a program forráskódjába. A keretrendszeremben implementált renderelő algoritmusok hangsúlyos része azonban árnyalókban található. Ennélfogva az árnyalók kódja kellően hosszú és összetett ahhoz, hogy hatékony és jól kezelhető tárolásuk csak önálló fájlok segítségével legyen lehetséges. Ebben az esetben azonban azok futásidejű elérését biztosítani kell, lehetőleg a felhasználó számára láthatatlan módon.

A probléma megoldására a két fent ismertetett módszert ötvöztem. Először is a grafikus árnyalók kódját különálló fájlokban tároltam a keretrendszer C++ forráskódja mellett. Ezt követően a fordítás során a programkönyvtár fordítási szkriptje megkeresi az árnyalók kódját tartalmazó fájlokat, és mindegyikhez generál egy C++ forrásfájlt egy, az árnyaló programkódját tartalmazó szöveges literállal. Ily módon a renderelő algoritmusok implementációi hozzáférnek a generált C++ forrásfájlokhoz, elkerülve a futás közbeni fájlrendszerelérés szükségességét.

Renderelés

A lencsefényfoltok szimulációja a rendszerem által biztosított objektumok esetén csak igény szerint megy végbe, amikor a felhasználó a programkönyvtárban található objektumok interfészein keresztül meghívja a megfelelő függvényeket. A befoglaló szoftver részéről tehát elvárt, hogy elvégezze ezen objektumok létrehozását és konfigurációját a korábban ismertetett módon. Ennek eredményeként az algoritmusokat implementáló osztályok elvégzik a szükséges GPU erőforrások előkészítését (pufferek allokációját, grafikus árnyalók fordítását stb.). Ezt követően a szoftver az applikációs ciklusban szükség szerint meghívhatja az objektumok renderelésért felelős metódusait, annak függvényében, hogy milyen gyakorisággal és paraméterekkel kívánja a szimulációt elvégezni.

Felhasználói interfész

Egy további meghatározó szempont a szerkesztőmodul felhasználói felületének megvalósításához használt programkönyvtár volt. Erre a célra elsősorban az operációs rendszerek natív eszközei és a harmadik féltől származó megoldások je-

lentették a főbb lehetőségeket. Ezen utóbbi kategória esetén is számos eltérő programkönyvtár érhető el, az interfészelemek renderelését az alkalmazásra bízó megoldásoktól [164] egészen a mindenre kiterjedő, eszközök széles tárházával rendelkező csomagokig [165].

Az egyes megközelítések hátrányait tekintve a natív eszközök esetén szükséges az egyes platformokhoz tartozó implementációk manuális létrehozása. A harmadik féltől származó egyszerűbb programkönyvtárak gyakran nem képesek a natív felhasználói felületek kinézetének imitációjára, nem teszteltek megfelelően a gyakorlatban, a fejlesztőik általi hosszútávú támogatás kétséges, az alkalmazásuk pedig tipikusan erősen komplikált. Mindezeket figyelembe véve választásom a Qt [165] keretrendszerre esett, amellyel a fent említett problémák összessége orvosolható. Ily módon a szerkesztői modul platformfüggetlensége és az operációs rendszer saját kinézetének megfelelő közelítése is elérhetővé vált, alacsony implementációs költségek és jó karbantarthatóság mellett.

2.3. Saját paraméterkereső algoritmus

Hullin és mtsai. lencsefényfolt-renderelő algoritmusa hatékonyan kombinálja a fizikai pontosságot és az alacsony futási időket. Ennélfogva a módszert a gyakorlatban is rendszeresen alkalmazzák [39], és az OpenLensFlare keretrendszer kulcsfontosságú algoritmusát képezi. A megközelítés lényeges hátránya egy költséges előfeldolgozási lépés, amely az összes szellemfolthoz megkeresi azt a tartományt, ahonnan a beérkező sugarak képesek eljutni a szenzorig. Az előfeldolgozás kihagyásával a szimuláció minősége (alulmintavételezés) vagy sebessége (megnövekedett sugárszám) drámaian romlik. Ezenkívül módszerük a szellemfoltok intenzitását is a szenzorig eljutó sugarak területe alapján határozza meg, amely előfeldolgozás nélkül számottevő eltéréseket produkál.

Az előfeldolgozó lépés nagyobb kamerarendszerek esetén akár órákat is igénybe vehet, ugyanis a szerzők által javasolt megközelítés sugárkövetést alkalmaz. A folyamat idejéből fakadóan viszont a lencsefényfoltok interaktív tervezése lehetetlen oly módon, hogy a felhasználó megfelelő képet kapjon a szimuláció várható tulajdonságairól.

Mindezen okokból fakadóan célunk egy olyan módszer megalkotása volt, amellyel a belépő sugárrácsot korlátoló tartomány interaktívan megbecsülhető. Az értekezés ezen részében új módszerünk részleteit ismertetem a 2019-ben prezentált [158] eredményeink alapján.

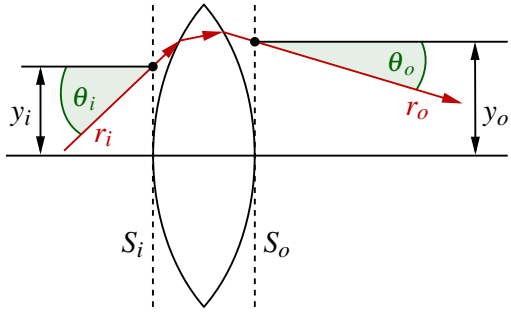
2.3.1. Matematikai háttér

Az általunk megalkotott módszer fő ötlete, hogy költséges sugárkövetés helyett lényegesen gyorsabb *transzfermátrixokat* [166] alkalmazzunk. Ily módon *paraxiális* (az optikai tengellyel kis szöget bezáró) sugarak és a *paraxiális approximáció* (a trigonometrikus függvények közelítése azok Taylor-sorfejtését egy együtthatóra korlátolva) segítségével a sugarak optikai rendszeren keresztüli terjedése mátrixokkal leírható. Módszerünk létrehozása során azzal a gyakran alkalmazott feltételezéssel éltünk, hogy az optikai rendszer fénytörő elemei sík- és gömbdarabok segítségével leírhatók [41, 148], ugyanis ily módon a szükséges paraxiális transzfermátrixok egyszerűen megadhatók. Bár a valós kamerarendszerek gyakran rendelkeznek ennél bonyolultabb elemekkel is, az azok gömbdarabként való kezeléséből fakadó hiba mértéke nem számottevő, viszont az elvégzendő számításokat nagymértékben leegyszerűsíti. Ezenkívül szükség esetén bővített modellek is alkalmazhatók a nem forgásszimmetrikus felületek transzfermátrixos sugárkövetésére [167].

Tekintsük most a transzfermátrix definícióját. Ehhez jelöljünk ki két, az optikai tengelyre merőleges síkot, S_i-t és S_o-t. Legyen továbbá $r_i = \begin{pmatrix} y_i & \theta_i \end{pmatrix}^T$ egy sugár, amely S_i-t az optikai tengelytől y_i távolságra metszi és az optikai tengellyel θ_i szöget zár be, $r_o = \begin{pmatrix} y_o & \theta_o \end{pmatrix}^T$ pedig az S_o-n keresett eredmény sugár. r_o kiszámításának egy lehetséges módja a következő [166]:

$$\begin{pmatrix} y_o \\ \theta_o \end{pmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{pmatrix} y_i \\ \theta_i \end{pmatrix}, \tag{2.1}$$

ahol az A, B, C és D értékekből képzett mátrix a *transzfermátrix*. Az egyes elemek viszonyát a 2.3. ábra szemlélteti.



2.3. ábra. A (2.1) egyenletben használt elemek viszonya.

Habár a transzfermátrixok konzekvens előállításával S_i és S_o megválasztása tetszőleges, a gyakorlatban tipikusan elemi mátrixok sorozatát alkalmazzuk egy folyamat modellezésére. Ebben az esetben az említett elemi mátrixok a fényvisszaverődés, a fénytörés és a fénysugár adott közegen való áthaladásának felelnek meg, a következő módon:

$$T = \begin{bmatrix} 1 & d \\ 0 & 1 \end{bmatrix}, \quad R_s = \begin{bmatrix} 1 & 0 \\ \frac{n_1 - n_2}{r n_2} & \frac{n_1}{n_2} \end{bmatrix}, \quad R_p = \begin{bmatrix} 1 & 0 \\ 0 & \frac{n_1}{n_2} \end{bmatrix}, \\ L_s = \begin{bmatrix} 1 & 0 \\ \frac{2}{r} & 1 \end{bmatrix}, \quad L_p = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{2.2}$$

ahol T, R és L rendre a fénysugár továbbítását, törését és visszaverődését jelöli, $\cdot_s$ és $\cdot_p$ a gömbi és síkbeli felületekre utal, n_1 és n_2 a közeghatár előtti és utáni törésmutató, r a görbületi sugár, d pedig a továbbítás távolsága.

2.3.2. Saját módszerünk

Módszerünk koncepcionálisan két fő lépésre tagolható: a sugárkövetéshez használt sugárrács korlátainak megkeresése a bemenő felületen és az egyéb renderelési paraméterek meghatározása.

Határoló négyszög meghatározása

Lee és Eisemann fényfoltrenderelő megközelítéséhez [148] hasonlóan a határoló régiók megkeresésére módszerünk transzfermátrixokat alkalmaz a sugárkövetés költségének elkerülésére. Módszerüktől eltérően azonban eljárásunk nem csak a rekesznyílás képét veszi figyelembe az eredmény meghatározásához, hanem a szellem útvonala során érintett összes fénytörő felületet megvizsgálja.

Algoritmusunk tervezése során kulcsfontosságú szempont volt a felhasznált elemek forgásszimmetriájának kiaknázása. Ily módon számításaink során a fényforrás és az optikai rendszer által bezárt horizontális és vertikális szögek közül elég csak a horizontális szöget figyelembe vennünk. Ezt követően a számításaink eredményeként kapott négyszög csúcsait egyszerűen elforgatjuk az optikai tengely körül a vertikális beesési szögnek megfelelően. Végezetül a kimenetet az elforgatott négyszöget befoglaló, tengelyekkel párhuzamos oldalakkal rendelkező négyszögként állítjuk elő. A horizontális szöghöz tartozó korlátok meghatározására létrehozott módszerünket a 2. algoritmus foglalja össze.

2. algoritmus. Saját módszerünk a belépő sugarakat korlátoló tartomány meghatározására.

Bemenet : Szellem útvonalához tartozó felületelem-magasságok és transzfermátrixok $\mathscr{P}$, beesési szög θ, optikai rendszer első felületelemének magassága h_0

Kimenet : Korlátoló négyszög középpontja c és oldalhosszai l

$b_{0^\circ}, b_\theta, B_{0^\circ}, B_\theta \leftarrow -\infty, -\infty, \infty, \infty$

foreach $(h, M) \in \mathscr{P}$ **do**

`// Határok kiszámítása az aktuális felülethez`

$y_{0^\circ}, Y_{0^\circ} \leftarrow \text{sort}\left(\frac{-h}{M_{1,1}}, \frac{h}{M_{1,1}}\right)$

$y_\theta, Y_\theta \leftarrow \text{sort}\left(\frac{-h-\theta M_{1,2}}{M_{1,1}}, \frac{h-\theta M_{1,2}}{M_{1,1}}\right)$

`// Szenzort elkerülő szellemek`

if $Y_\theta < -h_0$ **or** $y_\theta > h_0$ **then**

$c \leftarrow \{\, 0,0 \,\}, l \leftarrow \{\, 0,0 \,\}$

return

`// Határok frissítése a jelenlegi felület határaival`

$b_{0^\circ}, B_{0^\circ} \leftarrow \max(b_{0^\circ}, y_{0^\circ}), \min(B_{0^\circ}, Y_{0^\circ})$

$b_\theta, B_\theta \leftarrow \max(b_\theta, y_\theta), \min(B_\theta, Y_\theta)$

`// Kimeneti eredmény meghatározása`

$B_c \leftarrow \frac{b_\theta + B_\theta}{2}$ $B_e \leftarrow \max(B_\theta - b_\theta, B_{0^\circ} - b_{0^\circ})$

$c \leftarrow \{\, B_c, 0 \,\}, l \leftarrow \{\, B_e, B_e \,\}$

Egy adott szellem és horizontális beesési szög esetén először előállítjuk a szellem útvonalán található minden felülethez azt a transzfermátrixot, amely az optikai rendszerbe kívülről belépő sugarakat eljuttatja az adott felületig. Ehhez a sugár által elszenvedett jelenségeknek megfelelően a (2.2) egyenletben definiált transzfermátrixok szorzataként egy 2×2-es mátrixot állítunk elő minden felülethez. Bár ezek a mátrixok gyorsan kiszámíthatók, kis méretüknél fogva azok egyszeri előállítása és eltárolása elhanyagolható memóriaigénnyel rendelkezik, a számítási időre viszont kedvező hatással van. Ezt követően sorra megvizsgáljuk a szellem által érintett felületeket. A transzfermátrix és a felület magassága alapján a (2.1) egyenlettel minden elemhez meghatározzuk a külső felület azon tartományát, ahonnan a sugarakat a szellem útvonalán végigkövetve azok pontosan lefedik a vizsgált felületet. A rekesznyílás esetén ez a magasság a nyílás méretének felel meg. Ezt a lépést végrehajtjuk az összes érintett felületre, majd az így kapott régiók metszeteként előállítjuk az eljárás eredményét.

A paraxiális approximáció esetén a közelítés pontossága a beesési szög növekedésével arányosan csökken. A számítások költsége azonban kellően alacsony azok többszöri problémamentes végrehajtásához. Ennélfogva a pontosság növelésére a θ bemeneti beesési szögön túl a merőleges sugarakra ($\theta = 0^\circ$) is elvégezzük a fent ismertetett műveletsort minden felületelem vizsgálatakor. Az eredmény előállításához pedig egyszerűen mindkét számítást felhasználjuk a metszet meghatározásakor. Ezzel a módszerrel nagymértékben csökkenthetjük az olyan esetek előfordulását, ahol a kiszámított négyszög nem fedi le a teljes valós határoló tartományt.

Egyéb paraméterek megválasztása

A határoló régiók ismeretében Lee és Eisemann módszeréhez [148] hasonlóan egy adott szellem szenzoros képének határoló négyszögét is egyszerűen kiszámíthatjuk. Ehhez egyszerűen a szellem teljes útvonalához tartozó transzfermátrixot kell a szellem határoló négyszögére alkalmaznunk a (2.1) egyenlettel. Ezeket a négyszögeket több csatornára kiszámítva a szimulálandó csatornák száma és az eredmények simaságát garantáló spektrális szűrő elmosási iránya is meghatározható a Hullin és mtsai. által közzétett módon [41].

Hullin és mtsai. a sugárrácsméretek meghatározásához a paraméterkeresés során a sugárrácspontok képének eloszlását vették alapul. Ezt a megközelítést azonban nem alkalmazhatjuk módszerünkkel, hiszen csak a szellem szenzoros képét határoló négyszöghöz van hozzáférésünk. Gyakorlati tapasztalataink alapján azonban kisméretű sugárrács csak az alacsony intenzitású vagy a teljes optikai rendszert lefedő szellemfoltok esetén alkalmazható. Ellenkező esetben a rácspontok szélénél jól látható interpolációs műtermékek jelennek meg. Ezen szellemek megtalálása triviális a fent leírt módokon. Ezenkívül a szellemfoltot a szenzoron határoló négyszög területe, illetve annak a beeső sugárrács területével való viszonya is felhasználható heurisztikák megválasztására. Gyakorlati kísérleteink alapján azonban a megfelelően megválasztott határoló négyszöggel elvégzett sugárkövetéssel a legtöbb szellemhez jól használható egy adott, előre kijelölt rácsméret. Éppen ezért különösebb nehézséget csak az igazán komplex szellemalakzatok jelentenek. A problémát okozó komplex szellemfoltok száma azonban többnyire kellőképpen alacsony azok manuális azonosításához, így sugárrácsaik mérete manuálisan megnövelhető.

2.3.3. Elért eredmények

Az új módszerünkkel elért eredményeink kiértékelésére elkészítettük saját eljárásunknak és Hullin és mtsai. sugárkövetésen alapuló előfeldolgozási megközelítésének [41] referenciaimplementációját a C++ programozási nyelv és az OpenGL grafikus programkönyvtár [115] segítségével. Méréseinkhez egy Heliar Tronnier kamerát (USP 2645156, nyolc felület és 13 szellem), egy Nikon teleobjektívet (USP 4223981, 21 felület és 142 szellem), valamint egy Canon teleobjektívet (USP 5537259, 33 felület és 312 szellem) használtunk. Ezenkívül Hullin és mtsai. sugárkövetéses referenciamódszeréhez 256×256 méretű sugárrácsokat alkalmaztunk, aminek implementációja során a megfelelő teljesítményhez GPU-alapú sugárkövetést és CPU-alapú párhuzamos adatfeldolgozást alkalmaztunk.

Mivel módszerünk legfőbb célja az előfeldolgozási lépés számítási idejének csökkentése volt, így kiértékelésünket a futási idők összehasonlításával kezdtük. Ehhez megmértük a sugárkövetéses eljárás számítási idejét, illetve módszerünknek a transzfermátrix előállításához és a paraméterek kiszámításához szükséges lépéseinek futási idejét. Mindkét eljárás sebességmérése során az egy adott kameraálláshoz tartozó paraméterek meghatározási idejét vizsgáltuk. Módszerünk megalkotásakor kulcsfontosságú szempont volt a számítás képkockánként való végrehajthatósága az előfeldolgozási lépés teljes elkerülésének érdekében. Emiatt elvégeztük a paraméterkeresést a $\theta \in [0^\circ, 90^\circ]$ tartomány $0{,}5^\circ$-os lépésközzel meghatározott 181 pontján, majd az így kapott méréseket átlagoltuk. Eredményeinket a 2.1. táblázat foglalja össze.

Feladat	**Heliar Tronnier**	**Nikon**	**Canon**
Transzfermátrixok előállítása	0,234 ms	0,821 ms	0,969 ms
Paraméterkeresés	0,006 ms	0,058 ms	0,176 ms
Sugárkövetéses referencia	0,274 s	2,611 s	6,680 s

2.1. táblázat. Az általunk javasolt módszer és a referenciaeljárás futási ideje. Jól látható, hogy saját módszerünk interaktív módon, a másodperc törtrésze alatt képes megbecsülni a sugárkövetéses referencia eredményét.

Méréseinkből jól látható, hogy az általunk javasolt eljárás számítási költsége nagyságrendekkel kisebb, mint a sugárkövetéses referenciamódszeré. Míg az előző eljárás a legkisebb optikai rendszerrel sem alkalmas a paraméterkeresés valós időben történő elvégzésére, addig módszerünk a legnehezebb tesztesetben

is sokkal gyorsabb, mint amit egy valósidejű rendszer megkövetel. Ez a tulajdonság még akkor sem sérül, ha a transzfermátrixok kiszámítását is hozzáadjuk módszerünk futási idejéhez, azt feltételezve, hogy a kamerarendszer paraméterei képkockánként megváltoznak.

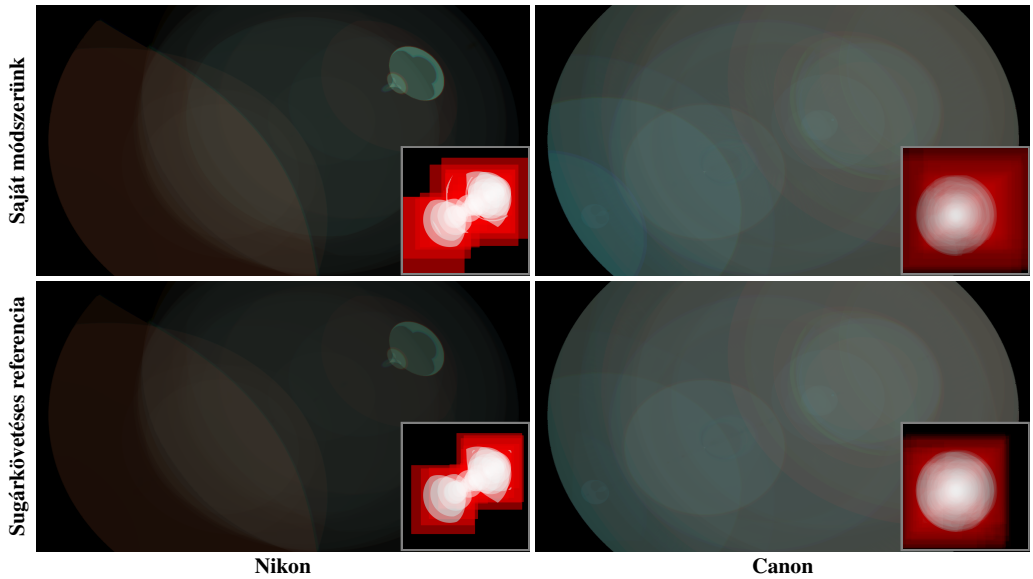
Ezt követően megvizsgáltuk eljárásunk pontosságát is. Ehhez azt vettük figyelembe, hogy a határoló négyszögek elhelyezkedése és mérete is nagyban befolyásolja a szimuláció minőségét. Ennélfogva kiszámítottuk kamerarendszerenként a generált határoló négyszögek középpontjának és oldalhosszainak a sugárkövetéses referenciához viszonyított átlagos abszolút hibáját a teljes paramétertartomány ($[0°, 90°]$) fölött. Továbbá annak érdekében, hogy megvizsgáljuk a paraxiális approximáció beesési szögtől függő hatását, kiszámítottuk a fent említett hibákat három egyenlő távolságra elhelyezett konkrét beesési szögre ($0°, 22{,}5°, 45°$) is. Végezetül a kapott eredményeket minden esetben normalizáltuk a kamerarendszer első felületelemének magasságával a hiba mértékének egyszerűbb megállapíthatóságához. Méréseink eredményét a 2.2. táblázat foglalja össze.

	Heliar Tronnier		**Nikon**		**Canon**	
	Középpont	**Oldalhossz**	**Középpont**	**Oldalhossz**	**Középpont**	**Oldalhossz**
$[0°, 90°]$	0,029814	0,084673	0,069531	0,114250	0,046572	0,071695
$0°$	0,000000	0,018264	0,000000	0,052451	0,000000	0,034406
$22{,}5°$	0,000045	0,018232	0,000251	0,052671	0,000066	0,034418
$45°$	0,000091	0,018200	0,001843	0,053058	0,000318	0,034379

2.2. táblázat. Határoló négyszögek középpontjának és oldalhosszainak sugárkövetéses referenciához viszonyított átlagos abszolút hibája, a kamerarendszer első felületelemének magasságával normalizálva. A relatív hibák mértéke alapján látható, hogy módszerünk alacsony hibával képes közelíteni a hosszadalmas sugárkövetéses folyamattal előállított referenciákat.

Eredményeink alapján elmondható, hogy bár módszerünk számítási ideje lényegesen alacsonyabb, annak pontosságát illetően az általunk javasolt megközelítés nem okoz számottevő eltérést a referenciaeljáráshoz viszonyítva. A határok középpontjai és oldalhosszai is jól közelítik a sugárkövetéssel számított értékeket. Mindezek alapján úgy ítéljük, hogy az általunk bemutatott megoldással a lencsefényfolt-renderelő algoritmus paraméterei teljesen valósidejű módon és magas pontossággal megbecsülhetők, így tehát kitűzött célunkat sikeresen elértük.

Végezetül az eltérések súlyának felmérésére megvizsgáltuk az algoritmusunkkal és a sugárkövetéses referenciamódszerrel generált paraméterek felhasználásával szimulált lencsefényfoltokat is. Ennek eredményét a 2.4. ábra szemlélteti a két magasabb komplexitású, Nikon és Canon optikai rendszerre. A bemutatott képek sarkaiban a szimulációhoz használt határoló négyszögeknek az optikai rendszer első felületére vetített képe is megtalálható. Jól látható, hogy a két paraméterező módszerrel létrehozott szimuláció közötti eltérés minimális, ami elsősorban a meghatározott határoló régiók nagymértékű hasonlóságának köszönhető. A leghangsúlyosabb különbséget a Nikon kamerarendszerrel generált kimenet jobb felső sarkában található szellemalakzat intenzitásában fedeztük fel. Meglátásunk szerint azonban az eltérés ebben az esetben sem számottevő, a referenciakimenet ismerete nélkül pedig nem sérti a szimuláció valószerűségét. Összességében tehát úgy ítéljük, hogy módszerünk megfelelően felhasználható a gyakorlatban, és a kitűzött céljainkat sikeresen megvalósítottuk.



2.4. ábra. Példa lencsefényfolt-szimulációk és a szimulációhoz használt sugárrácsokat határoló négyszögek az általunk javasolt paraméterkereső módszerrel és a sugárkövetés referenciaeljárással. Jól látható, hogy az interaktív módszerünkkel paraméterezett szimuláció nagy pontossággal közelíti a költséges előfeldolgozással előállított kimeneteket.

Összefoglalás

Az értekezésben bemutattam a képalkotó rendszerek sajátosságainak szimulációját érintő kutatásaink új eredményeit. Munkánk két fő területét a vizuális aberrációkkal terhelt emberi látás személyre szabott szimulációja (1. fejezet), valamint a kamerarendszerek lencsefényfoltjainak tervezését és valósidejű megjelenítését támogató szoftverek és módszerek fejlesztése (2. fejezet) képezik.

Az emberi látás személyreszabható szimulációjához először létrehoztunk egy saját paraméteres szemmodellt. Továbbá eljárásokat készítettünk a szemmodell vizuális aberrációinak meghatározására, a modell fókusztávolságának módosítására, illetve a szimulálni kívánt szem fizikai struktúráját leíró modellparaméterek becslésére. Ezenkívül megalkottunk egy neurális hálókra épülő megközelítést is, amellyel mindezen feladatok alacsony hibával és teljesen interaktívan elvégezhetők. Új módszereink részleteit az 1.3. szakasz ismerteti, amelyeket a 2021-ben [75] és a 2024-ben [76] megjelent cikkünkben mutattunk be a tudományos közösség számára.

Létrehoztunk egy GPU-alapú módszert is az emberi szem nagyszámú pontszórásfüggvényeinek hatékony kiszámítására, amely a kiterjesztett Nijboer–Zernike-féle módszer formuláinak felbontásán alapul. Eljárásunkkal a szimulációhoz szükséges pontszórásfüggvények számítási ideje nagyságrendekkel lecsökkenthető, és a pontszórásfüggvények ritka rácsán alapuló látásszimuláló algoritmusok pontszórásfüggvényei teljesen interaktív módon előállíthatók. Módszerünk részleteit az 1.4. szakasz ismerteti, amelyeket a 2022-ben megjelent publikációnkban [112] foglaltunk össze.

Az aberrációkkal terhelt emberi látás hatékony szimulációjára először létrehoztunk egy komplex fázorokra épülő, szeparábilis konvolúción alapuló eljárást. Módszerünkkel az emberi szem alacsonyrendű aberrációkat tartalmazó pontszórásfüggvényei hatékonyan közelíthetők, a látás pedig valós időben szimulálható. Ezenkívül megalkottunk egy csempézett konvolúciót és a szem pontszórásfüggvényeinek képpontonkénti, GPU-alapú interpolációját alkalmazó módszert is. Algoritmusunkkal a szem tetszőleges vizuális aberrációja és a periférikus látása valós időben szimulálható, a folytonos pontszórásfüggvény pedig magas pontossággal közelíthető. Az így megalkotott új algoritmusaink részleteit az 1.5. szakaszban mutattam be, amelyeket a 2018-ban [123], a 2021-ben [75] és a 2024-ben [76] megjelent cikkünkben tettünk elérhetővé a tudományos közösség számára.

A látásszimuláció nehézségeinél és széles körű felhasználhatóságánál fogva készítettünk egy tanulmányt is az emberi látást szimuláló módszerek jellemzőiről és azok felhasználási lehetőségeiről. Kutatásunk eredményeit a 2023-ban megjelent cikkünkben [36] publikáltuk.

A kamerarendszerek lencsefényfoltjainak tervezésére és szimulációjára létrehoztam a nyílt forráskódú *OpenLensFlare* keretrendszert. Rendszerem egy renderelő és egy tervezői modul segítségével lehetővé teszi a lencsefényfoltok fizikai alapú szimulációját, megjelenésük tervezését, a szimuláció elemzését, illetve a szimulációhoz szükséges adatok kezelését. Ezenkívül megalkottunk egy hatékony paraméterkereső eljárást is Hullin és mtsai. lencsefényfolt-renderelő módszeréhez [41], amellyel valós időben kiszámíthatók az algoritmus hatékony szimulációhoz szükséges fényfoltonkénti paraméterei. A keretrendszerem és a paraméterkereső módszerünk részleteit a 2.2 és a 2.3. szakasz ismerteti, amelyeket a 2017-ben [157] és a 2019-ben [158] megjelent publikációkban mutattunk be a tudományos közösség számára.

Summary

This dissertation presented our new results regarding the computer simulation of imaging systems characteristics. Our research efforts focused on the personalizable simulation of aberrated human vision (Chapter 1) and the planning and rendering of the lens flare effect of camera systems (Chapter 2).

For the purposes of personalized human vision simulation, we first constructed a custom parametric eye model to represent the physical structure of the simulated human eye. We also created supporting methods to calculate the visual aberrations of the eye model, change its focus distance, and estimate a set of model parameters corresponding to the physical structure of the simulated eye. Furthermore, we also created a neural network-based approach to perform these tasks accurately and fully interactively. The details of our new methods are outlined in Section 1.3. We published our new results in [75, 76].

To efficiently compute large sets of human point-spread functions (PSFs), we developed a GPU-based PSF computation method using the extended Nijboer-Zernike (ENZ) method. Our proposed approach utilizes a custom rearrangement of the key terms in the ENZ-based PSF formulation, which allows the precomputation and reuse of several subterms, substantially reducing the overall PSF computation times. Critically, our new method facilitates the fully interactive computation of the coarse PSF grids used by our vision simulation approaches. Our proposed method is described in Section 1.4. We published these results in [112].

Regarding the simulation of aberrated human vision, we first developed a rendering method that utilizes complex phasors as separable convolution kernels. Our new approach efficiently approximates the PSF of eyes with low-order visual aberrations and facilitates the real-time simulation of aberrated vision. Furthermore, we also developed a tile-based convolutional algorithm that utilizes the per-pixel PSF of the simulated eye for a more plausible simulation of vision. Our proposed algorithm includes an accurate, GPU-based method for approximating the true per-pixel PSF, facilitating the real-time simulation of arbitrary visual aberrations and peripheral vision. Furthermore, our new approach allows the dynamic, on-the-fly modification of pupil size and focus distance. The details of our new algorithms are presented in Section 1.5. We published our new results in [75, 76, 123].

Owing to the difficulty and widespread usability of vision simulation, we reviewed the previously existing rendering algorithms for aberrated human vision. Furthermore, we also outlined the main challenges and potential applications of these methods. Our study was published in [36].

For the efficient planning and real-time simulation of camera lens flares, I developed an open-source rendering and editing framework called *OpenLensFlare*. Comprising a runtime library and an editor component, OpenLensFlare facilitates the physically based, real-time simulation of camera lens flares and provides the tools for managing, analyzing, and editing the input optical elements and other simulation parameters. Furthermore, we also developed an efficient parametrization method for the lens flare rendering algorithm of Hullin et al. [41]. Our proposed approach facilitates the real-time computation of the per-ghost parameters needed for generating accurate simulations, allowing the interactive modification of camera parameters and the efficient planning of the corresponding lens flares. The details of the OpenLensFlare framework and our proposed parametrization method are outlined in Section 2.2 and Section 2.3. We published these results in [157, 158].

Köszönetnyilvánítás

Először is köszönettel tartozom témavezetőmnek, dr. Kunkli Rolandnak, aki szakmai előrehaladásomat témavezetői munkájával már alapképzésem óta aktívan támogatja. Munkám során hasznos tanácsaival és észrevételeivel folyamatosan rendelkezésemre állt, amely során tanúsított pontossága és türelme nagymértékben hozzájárult szakmai fejlődésemhez és az értekezésben prezentált eredményeink eléréséhez.

Ezenkívül szeretnék köszönetet mondani családomnak és barátaimnak, akik munkám alatt végig türelemmel és segítőkészséggel álltak mellettem. Külön köszönettel tartozom feleségemnek, Csoba-Bodonyi Andreának, aki a legnehezebb időszakok során is rendületlenül támogatott és a haladásom során bátorított.

Végezetül szeretném még megköszönni a kutatásomat lehetővé tévő 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, „NVIDIA GPU Grant” program keretein belül adományozott Titan Xp videokártyát.

Irodalomjegyzék

[1] T. Li, C. Li, X. Zhang, W. Liang, Y. Chen, Y. Ye és H. Lin. „Augmented Reality in Ophthalmology: Applications and Challenges”. *Frontiers in Medicine* 8 (2021), 733241:1–733241:12. DOI: https://doi.org/10.3389/fmed.2021.733241.

[2] G. Aydindoğan, K. Kavakli, A. Şahin, P. Artal és H. Ürey. „Applications of augmented reality in ophthalmology [Invited]”. *Biomedical Optics Express* 12.1 (2021), 511–538. DOI: https://doi.org/10.1364/BOE.405026.

[3] M. Iskander, T. Ogunsola, R. Ramachandran, R. McGowan és L. A. Al-Aswad. „Virtual Reality and Augmented Reality in Ophthalmology: A Contemporary Prospective”. *Asia-Pacific Journal of Ophthalmology* 10.3 (2021), 244–252. DOI: https://doi.org/10.1097/APO.0000000000000409.

[4] N. Zaman. „EyeSightVR: An Immersive and Automated Tool for Comprehensive Assessment of Visual Function”. Disszertáció. University of Nevada, Reno, USA, 2021. máj. URL: https://scholarworks.unr.edu/handle/11714/7877. Elérés dátuma: 2024. június 3.

[5] J. E. Greivenkamp, J. Schwiegerling, J. M. Miller és M. D. Mellinger. „Visual Acuity Modeling Using Optical Raytracing of Schematic Eyes”. *American Journal of Ophthalmology* 120.2 (1995), 227–240. DOI: https://doi.org/10.1016/S0002-9394(14)72611-X.

[6] A. B. Watson és A. J. Ahumada. „Predicting visual acuity from wavefront aberrations”. *Journal of Vision* 8.4 (2008), 17:1–17:19. DOI: https://doi.org/10.1167/8.4.17.

[7] C. Fülep, I. Kovács, K. Kránitz és G. Erdei. „Simulation of visual acuity by personalizable neuro-physiological model of the human eye”. *Scientific Reports* 9 (2019), 7805:1–7805:15. DOI: https://doi.org/10.1038/s41598-019-44160-z.

[8] J. Tabernero, A. Benito, E. Alcón és P. Artal. „Mechanism of compensation of aberrations in the human eye”. *Journal of the Optical Society of America A* 24.10 (2007), 3274–3283. DOI: https://doi.org/10.1364/JOSAA.24.003274.

[9] J. Tabernero, E. Berrio és P. Artal. „Modeling the mechanism of compensation of aberrations in the human eye for accommodation and aging”. *Journal of the Optical Society of America A* 28.9 (2011), 1889–1895. DOI: https://doi.org/10.1364/JOSAA.28.001889.

[10] X. Cheng, A. Bradley és L. N. Thibos. „Predicting subjective judgment of best focus with objective image quality metrics”. *Journal of Vision* 4.4 (2004), 310–321. DOI: https://doi.org/10.1167/4.4.7.

[11] E. Tural és M. Tural. „Luminance contrast analyses for low vision in a senior living facility: A proposal for an HDR image-based analysis tool”. *Building and Environment* 81 (2014), 20–28. DOI: https://doi.org/10.1016/j.buildenv.2014.06.005.

[12] Y.-Z. Xiong, Q. Lei, A. Calabrèse és G. E. Legge. „Simulating Visibility and Reading Performance in Low Vision”. *Frontiers in Neuroscience* 15 (2021), 671121:1–671121:13. DOI: https://doi.org/10.3389/fnins.2021.671121.

[13] M. Bennett és A. Quigley. „Creating Personalized Digital Human Models Of Perception for Visual Analytics”. *User Modeling, Adaptation, and Personalization*. 6787. köt. Girona, Spanyolország: Springer, 2011, 25–37. DOI: https://doi.org/10.1007/978-3-642-22362-4_3.

[14] M. L. Krueger, M. M. Oliveira és A. L. Kronbauer. „Personalized Visual Simulation and Objective Validation of Low-Order Aberrations of the Human Eye”. *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. Sao Paulo, Brazília: IEEE Computer Society, 2016, 64–71. DOI: https://doi.org/10.1109/SIBGRAPI.2016.018.

[15] S. Barbero és J. Portilla. „Simulating real-world scenes viewed through ophthalmic lenses”. *Journal of the Optical Society of America A* 34.8 (2017), 1301–1308. DOI: https://doi.org/10.1364/JOSAA.34.001301.

[16] B. Jin, Z. Ai és M. Rasmussen. „Simulation of Eye Disease in Virtual Reality”. *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the Engineering in Medicine and Biology Society*. Shanghai, Kína: IEEE Computer Society, 2005, 5128–5131. DOI: https://doi.org/10.1109/IEMBS.2005.1615631.

[17] K. Krösl, C. Elvezio, M. Hürbe, S. Karst, S. Feiner és M. Wimmer. „XR-Eye: Simulating Visual Impairments in Eye-Tracked XR”. *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops*. Atlanta, USA: IEEE Computer Society, 2020, 830–831. DOI: https://doi.org/10.1109/VRW50115.2020.00266.

[18] H. C. Ates, A. Fiannaca és E. Folmer. „Immersive Simulation of Visual Impairments Using a Wearable See-through Display”. *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction*. Stanford, USA: ACM, 2015, 225–228. DOI: https://doi.org/10.1145/2677199.2680551.

[19] W. Wang. „Intelligent Planning for Refractive Surgeries: A Modelling and Visualisation-based Approach”. Disszertáció. University of Liverpool, Liverpool, UK, 2020. jún. DOI: https://doi.org/10.17638/03090577.

[20] J. Tabernero, P. Piers, A. Benito, M. Redondo és P. Artal. „Predicting the Optical Performance of Eyes Implanted with IOLs to Correct Spherical Aberration”. *Investigative Ophthalmology & Visual Science* 47.10 (2006), 4651–4658. DOI: https://doi.org/10.1167/iovs.06-0444.

[21] J. Loos, P. Slusallek és H.-P. Seidel. „Using Wavefront Tracing for the Visualization and Optimization of Progressive Lenses”. *Computer Graphics Forum* 17.3 (2001), 255–265. DOI: https://doi.org/10.1111/1467-8659.00272.

[22] M. Nießner, R. Sturm és G. Greiner. „Real-time Simulation and Visualization of Human Vision Through Eyeglasses on the GPU”. *Proceedings - VRCAI 2012*. Szingapúr, Szingapúr: ACM, 2012, 195–202. DOI: https://doi.org/10.1145/2407516.2407565.

[23] D. Gonzalez Utrera. „Metrology and Simulation with Progressive Addition Lenses”. Disszertáció. The University of Arizona, Arizona, USA, 2018. URL: https://repository.arizona.edu/handle/10150/631382. Elérés dátuma: 2024. június 3.

[24] O. Keleş és E. Anarim. „Adjustment of Digital Screens to Compensate the Eye Refractive Errors via Deconvolution”. *2019 Ninth International Conference on Image Processing Theory, Tools and Applications (IPTA)*.

Isztambul, Törökország: IEEE Computer Society, 2019, 1–6. DOI: https://doi.org/10.1109/IPTA.2019.8936098.

[25] B. A. Barsky, F.-C. Huang, D. Lanman, G. Wetzstein és R. Raskar. „Vision Correcting Displays Based on Inverse Blurring and Aberration Compensation". *Computer Vision - ECCV 2014 Workshops*. Zürich, Svájc: Springer, 2015, 524–538. DOI: https://doi.org/10.1007/978-3-319-16199-0_37.

[26] Y. Itoh és G. Klinker. „Vision Enhancement: Defocus Correction via Optical See-Through Head-Mounted Displays". *Proceedings of the 6th Augmented Human International Conference*. Szingapúr, Szingapúr: ACM, 2015, 1–8. DOI: https://doi.org/10.1145/2735711.2735787.

[27] A. Maimone, A. Georgiou és J. S. Kollin. „Holographic Near-Eye Displays for Virtual and Augmented Reality". *ACM Transactions on Graphics* 36.4 (2017), 85:1–85:16. DOI: https://doi.org/10.1145/3072959.3073624.

[28] K. Yamamoto, I. Suzuki, K. Namikawa, K. Sato és Y. Ochiai. „Interactive Eye Aberration Correction for Holographic Near-Eye Display". *Proceedings - AHs 2021*. Rovaniemi, Finnország: ACM, 2021, 204–214. DOI: https://doi.org/10.1145/3458709.3458955.

[29] M. S. Arefin. „[DC] SharpView AR: Enhanced Visual Acuity for Out-of-Focus Virtual Content". *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops*. Liszszabon, Portugália: IEEE Computer Society, 2021, 731–732. DOI: https://doi.org/10.1109/VRW52623.2021.00248.

[30] 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.

[31] S. A. Cholewiak, G. D. Love és M. S. Banks. „Creating correct blur and its effect on accommodation". *Journal of Vision* 18.9 (2018), 1:1–1:29. DOI: https://doi.org/10.1167/18.9.1.

[32] L. Xiao, A. Kaplanyan, A. Fix, M. Chapman és D. Lanman. „DeepFocus: Learned Image Synthesis for Computational Displays". *ACM Transac-*

tions on Graphics 37.6 (2018), 200:1–200:13. DOI: https://doi.org/10.1145/3272127.3275032.

[33] A. T. Duchowski, D. H. House, J. Gestring, R. I. Wang, K. Krejtz, I. Krejtz, R. Mantiuk és B. Bazyluk. „Reducing Visual Discomfort of 3D Stereoscopic Displays with Gaze-Contingent Depth-Of-Field". *Proceedings, SAP 2014*. Vancouver, Kanada: ACM, 2014, 39–46. DOI: https://doi.org/10.1145/2628257.2628259.

[34] F. Xu és D. Li. „Software Based Visual Aberration Correction for HMDs". *25th IEEE Conference on Virtual Reality and 3D User Interfaces*. Reutlingen, Németország: IEEE Computer Society, 2018, 246–250. DOI: https://doi.org/10.1109/VR.2018.8447557.

[35] A. H. Güzel, J. Beyazian, P. Chakravarthula és K. Akșit. „ChromaCorrect: prescription correction in virtual reality headsets through perceptual guidance". *Biomedical Optics Express* 14.5 (2023), 2166–2180. DOI: https://doi.org/10.1364/BOE.485776.

[36] I. Csoba és R. Kunkli. „Rendering algorithms for aberrated human vision simulation". *Visual Computing for Industry, Biomedicine, and Art* 6 (2023), 5:1–5:25. DOI: https://doi.org/10.1186/s42492-023-00132-9.

[37] C. Raud. „How post-processing effects imitating camera artifacts affect the perceived realism and aesthetics of digital game graphics". Szakdolgozat. Södertörn University, Stockholm, Svédország, 2018. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:sh:diva-34888. Elérés dátuma: 2024. június 3.

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

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

[40] T. Akenine-Möller, E. Haines, N. Hoffman, A. Pesce, M. Iwanicki és S. Hillaire. *Real-Time Rendering*. 4. kiad. Roca Baton, USA: A K Peters/CRC Press, 2018.

[41] 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.

[42] D. A. Atchison és G. Smith. *Optics of the Human Eye*. 2. kiad. Boca Raton, USA: CRC Press, 2023.

[43] R. Navarro, J. Santamaría és J. Bescós. „Accommodation-dependent model of the human eye with aspherics”. *Journal of the Optical Society of America A* 2.8 (1985), 1273–1280. DOI: https://doi.org/10.1364/JOSAA.2.001273.

[44] C. Pruss, E. Garbusi és W. Osten. „Testing Aspheres”. *Optics and Photonics News* 19.4 (2008), 24–29. DOI: https://doi.org/10.1364/OPN.19.4.000024.

[45] J. W. Goodman. *Introduction to Fourier Optics*. 4. kiad. New York, USA: W.H. Freeman, Macmillan Learning, 2017.

[46] A. B. Bhatia és E. Wolf. „On the circle polynomials of Zernike and related orthogonal sets”. *Mathematical Proceedings of the Cambridge Philosophical Society* 50.1 (1954), 40–48. DOI: https://doi.org/10.1017/S0305004100029066.

[47] G.-m. Dai. *Wavefront Optics for Vision Correction*. Bellingham, USA: SPIE, 2008. DOI: https://doi.org/10.1117/3.769212.

[48] J. Herrmann. „Least-squares wave front errors of minimum norm”. *Journal of the Optical Society of America* 70.1 (1980), 28–35. DOI: https://doi.org/10.1364/JOSA.70.000028.

[49] L. N. Thibos, A. Bradley és X. Hong. „A statistical model of the aberration structure of normal, well-corrected eyes”. *Ophthalmic and Physiological Optics* 22.5 (2002), 427–433. DOI: https://doi.org/10.1046/j.1475-1313.2002.00059.x.

[50] A. B. Watson. „Computing human optical point spread functions”. *Journal of Vision* 15.2 (2015), 26:1–26:25. DOI: https://doi.org/10.1167/15.2.26.

[51] S. Van Haver. „The Extended Nijboer-Zernike Diffraction Theory and its Applications”. Disszertáció. Delft University of Technology, Delft, Hol-

landia, 2010. URL: http://resolver.tudelft.nl/uuid:8d96ba75-24da-4e31-a750-1bc348155061. Elérés dátuma: 2024. június 3.

[52] K. Niu és C. Tian. „Zernike polynomials and their applications”. *Journal of Optics* 24.12 (2022), 123001:1–123001:54. DOI: https://doi.org/10.1088/2040-8986/ac9e08.

[53] E. C. Kintner és R. M. Sillitto. „A New 'Analytic' Method for Computing the Optical Transfer Function”. *Optica Acta: International Journal of Optics* 23.8 (1976), 607–619. DOI: https://doi.org/10.1080/713819333.

[54] J. Antonello és M. Verhaegen. „Modal-based phase retrieval for adaptive optics”. *Journal of the Optical Society of America A* 32.6 (2015), 1160–1170. DOI: https://doi.org/10.1364/JOSAA.32.001160.

[55] J. J. M. Braat, S. van Haver, A. J. E. M. Janssen és P. Dirksen. „Assessment of optical systems by means of point-spread functions”. *Progress in Optics* 51 (2008), 349–468. DOI: https://doi.org/10.1016/S0079-6638(07)51006-1.

[56] A. Roorda, D. T. Miller és J. Christou. „Strategies for High-Resolution Retinal Imaging”. *Adaptive Optics for Vision Science*. Hoboken, USA: John Wiley & Sons, Inc., 2006, 235–287.

[57] A. J. E. M. Janssen, J. J. M. Braat és P. Dirksen. „On the computation of the Nijboer-Zernike aberration integrals at arbitrary defocus”. *Journal of Modern Optics* 51.5 (2004), 687–703. DOI: https://doi.org/10.1080/09500340408235546.

[58] J. J. Camp, L. J. Maguire és R. A. Robb. „An Efficient Ray Tracing Algorithm for Modeling Visual Performance from Corneal Topography”. *First Conference on Visualization in Biomedical Computing*. Atlanta, USA: IEEE Computer Society, 1990, 278–285. DOI: https://doi.org/10.1109/VBC.1990.109333.

[59] B. A. Barsky. „Vision-Realistic Rendering: Simulation of the Scanned Foveal Image from Wavefront Data of Human Subjects”. *Proceedings APGV 2004*. Los Angeles, USA: ACM, 2004, 73–81. DOI: https://doi.org/10.1145/1012551.1012564.

[60] B. A. Barsky. „Vision-Realistic Rendering: Simulation of the Scanned Foveal Image with Elimination of Artifacts due to Occlusion and Discretization”. *Computer Vision, Imaging and Computer Graphics. Theory and Applications*. Angers, Franciaország: Springer, 2011, 3–27. DOI: https://doi.org/10.1007/978-3-642-25382-9_1.

[61] J. A. Rodríguez Celaya, P. Brunet Crosa, N. Ezquerra és J. E. Palomar. „A Virtual Reality approach to progressive lenses simulation”. *Actas del XV Congreso Español de Informática Gráfica*. Madrid, Spanyolország: Thomson-Paraninfo, 2005, 43–52.

[62] A. Seidemann, F. Schaeffel, A. Guirao, N. Lopez-Gil és P. Artal. „Peripheral refractive errors in myopic, emmetropic, and hyperopic young subjects”. *Journal of the Optical Society of America A* 19.12 (2002), 2363–2373. DOI: https://doi.org/10.1364/JOSAA.19.002363.

[63] S. Mostafawy, O. Kermani és H. Lubatschowski. „Virtual Eye: Retinal Image Visualization of the Human Eye”. *IEEE Computer Graphics and Applications* 17.1 (1997), 8–12. DOI: https://doi.org/10.1109/38.576849.

[64] J. Wu, C. Zheng, X. Hu és F. Xu. „Realistic Simulation of Peripheral Vision Using An Aspherical Eye Model”. *Eurographics 2011 - Short Papers*. Llandudno, UK: The Eurographics Association, 2011, 37–40. DOI: https://doi.org/10.2312/EG2011/short/037-040.

[65] C. Dias, M. Wick, K. Rifai és S. Wahl. „Peripheral Retinal Image Simulation Based on Retina Shapes”. *Eurographics 2016 - Short Papers*. Lisszabon, Portugália: The Eurographics Association, 2016, 61–64. DOI: https://doi.org/10.2312/egsh.20161015.

[66] T. Lian, K. J. MacKenzie, D. H. Brainard, N. P. Cottaris és B. A. Wandell. „Ray tracing 3D spectral scenes through human optics models”. *Journal of Vision* 19.12 (2019), 23:1–23:17. DOI: https://doi.org/10.1167/19.12.23.

[67] W. Fink és D. Micol. „simEye: computer-based simulation of visual perception under various eye defects using Zernike polynomials”. *Journal of Biomedical Optics* 11.5 (2006), 54011:1–054011:12. DOI: https://doi.org/10.1117/1.2357734.

[68] Q. Wei, S. Patkar és D. K. Pai. „Fast ray-tracing of human eye optics on Graphics Processing Units”. *Computer Methods and Programs in Biomedicine* 114.3 (2014), 302–314. DOI: https://doi.org/10.1016/j.cmpb.2014.02.003.

[69] C. T. Vu, S. Stock, L. T. Fan és W. Stork. „Highly parallelized rendering of the retinal image through a computer-simulated human eye for the design of virtual reality head-mounted displays”. *Optics, Photonics and Digital Technologies for Imaging Applications VI*. 11353. köt. Bellingham, USA: SPIE, 2020, 251–270. DOI: https://doi.org/10.1117/12.2555872.

[70] N. Tang és S. Xiao. „Real-time Human Vision Rendering Using Blur Distribution Function”. *Proceedings VRCAI 2015*. Kóbe, Japán: ACM, 2015, 39–42. DOI: https://doi.org/10.1145/2817675.2817686.

[71] A. R. C. Lima, A. M. Medeiros, V. G. Marques és M. M. Oliveira. „Real-time simulation of accommodation and low-order aberrations of the human eye using light-gathering trees”. *The Visual Computer* 37.9–11 (2021), 2581–2593. DOI: https://doi.org/10.1007/s00371-021-02194-3.

[72] X. Wei és L. N. Thibos. „Modeling the eye's optical system by ocular wavefront tomography”. *Optics Express* 16.25 (2008), 20490–20502. DOI: https://doi.org/10.1364/OE.16.020490.

[73] T. Liu és L. N. Thibos. „Customized models of ocular aberrations across the visual field during accommodation”. *Journal of Vision* 19.9 (2019), 13:1–13:24. DOI: https://doi.org/10.1167/19.9.13.

[74] Ansys, Inc. *Ansys Zemax OpticStudio*. URL: https://www.ansys.com/products/optics/ansys-zemax-opticstudio. Elérés dátuma: 2024. június 3.

[75] I. Csoba és R. Kunkli. „Efficient Rendering of Ocular Wavefront Aberrations using Tiled Point-Spread Function Splatting”. *Computer Graphics Forum* 40.6 (2021), 182–199. DOI: https://doi.org/10.1111/cgf.14267.

[76] I. Csoba és R. Kunkli. „Fast rendering of central and peripheral human visual aberrations across the entire visual field with interactive personalization”. *The Visual Computer* 40.5 (2024), 3709–3731. DOI: https://doi.org/10.1007/s00371-023-03060-0.

[77] A. V. Goncharov és C. Dainty. „Wide-field schematic eye models with gradient-index lens”. *Journal of the Optical Society of America A* 24.8 (2007), 2157–2174. DOI: https://doi.org/10.1364/JOSAA.24.002157.

[78] A. Khan, J. M. Pope, P. K. Verkicharla, M. Suheimat és D. A. Atchison. „Change in human lens dimensions, lens refractive index distribution and ciliary body ring diameter with accommodation”. *Biomedical Optics Express* 9.3 (2018), 1272–1282. DOI: https://doi.org/10.1364/BOE.9.001272.

[79] E. A. Hermans, P. J. W. Pouwels, M. Dubbelman, J. P. A. Kuijer, R. G. L. van der Heijde és R. M. Heethaar. „Constant Volume of the Human Lens and Decrease in Surface Area of the Capsular Bag during Accommodation: An MRI and Scheimpflug Study”. *Investigative Ophthalmology & Visual Science* 50.1 (2009), 281–289. DOI: https://doi.org/10.1167/iovs.08-2124.

[80] L. Lundström és P. Unsbo. „Transformation of Zernike coefficients: scaled, translated, and rotated wavefronts with circular and elliptical pupils”. *Journal of the Optical Society of America A* 24.3 (2007), 569–577. DOI: https://doi.org/10.1364/JOSAA.24.000569.

[81] S. P. Boyd és L. Vandenberghe. *Convex Optimization*. 1. kiad. Cambridge, UK: Cambridge University Press, 2004.

[82] V. Torczon. „On the Convergence of Pattern Search Algorithms”. *SIAM Journal on Optimization* 7.1 (1997), 1–25. DOI: https://doi.org/10.1137/S1052623493250780.

[83] C. Audet és J. E. Dennis. „Analysis of Generalized Pattern Searches”. *SIAM Journal on Optimization* 13.3 (2002), 889–903. DOI: https://doi.org/10.1137/S1052623400378742.

[84] F. Cavas-Martínez, E. De la Cruz Sánchez, J. Nieto Martínez, F. J. Fernández Cañavate és D. G. Fernández-Pacheco. „Corneal topography in keratoconus: state of the art”. *Eye and Vision* 3 (2016), 5:1–5:12. DOI: https://doi.org/10.1186/s40662-016-0036-8.

[85] J. J. Rozema, P. Rodriguez, R. Navarro és M.-J. Tassignon. „SyntEyes: A Higher-Order Statistical Eye Model for Healthy Eyes”. *Investigative*

Ophthalmology & Visual Science 57.2 (2016), 683–691. DOI: https://doi.org/10.1167/iovs.15-18067.

[86] J. J. Rozema, D. A. Atchison, S. Kasthurirangan, J. M. Pope és M.-J. Tassignon. „Methods to Estimate the Size and Shape of the Unaccommodated Crystalline Lens In Vivo". *Investigative Ophthalmology & Visual Science* 53.6 (2012), 2533–2540. DOI: https://doi.org/10.1167/iovs.11-8645.

[87] J. S. Larsen. „Axial length of the emmetropic eye and its relation to the head size". *Acta Ophthalmologica* 57.1 (1979), 76–83. DOI: https://doi.org/10.1111/j.1755-3768.1979.tb06662.x.

[88] I. Kovács, K. Miháltz, J. Németh és Z. Z. Nagy. „Anterior chamber characteristics of keratoconus assessed by rotating Scheimpflug imaging". *Journal of Cataract & Refractive Surgery* 7 (2010), 1101–1106. DOI: https://doi.org/10.1016/j.jcrs.2009.12.046.

[89] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. S. Awwal és V. K. Asari. „A State-of-the-Art Survey on Deep Learning Theory and Architectures". *Electronics* 8.3 (2019), 292:1–292:66. DOI: https://doi.org/10.3390/electronics8030292.

[90] K. He, X. Zhang, S. Ren és J. Sun. „Deep Residual Learning for Image Recognition". *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, USA: IEEE Computer Society, 2016, 770–778. DOI: https://doi.org/10.1109/CVPR.2016.90.

[91] C. Zuo, J. Qian, S. Feng, W. Yin, Y. Li, P. Fan, J. Han, K. Qian és Q. Chen. „Deep learning in optical metrology: a review". *Light: Science & Applications* 11 (2022), 39:1–39:54. DOI: https://doi.org/10.1038/s41377-022-00714-x.

[92] D. Chen, F. Hu, G. Nian és T. Yang. „Deep Residual Learning for Nonlinear Regression". *Entropy* 22.2 (2020), 193:1–193:14. DOI: https://doi.org/10.3390/e22020193.

[93] V. Nair és G. E. Hinton. „Rectified Linear Units Improve Restricted Boltzmann Machines". *Proceedings of the 27th International Conference on Machine Learning*. Haifa, Izrael: Omnipress, 2010, 807–814.

[94] Misra, Diganta. *Mish - A Self Regularized Non-Monotonic Activation Function*. Konferenciaanyag. The 31st British Machine Vision Virtual Conference, BMVC 2020. URL: https://bmvc2020-conference.com/conference/papers/paper_0928.html. Elérés dátuma: 2024. június 3.

[95] S. Ioffe és C. Szegedy. „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". *Proceedings of the 32nd International Conference on Machine Learning*. 37. köt. Lille, Franciaország: JMLR, 2015, 448–456.

[96] J. L. Ba, J. R. Kiros és G. E. Hinton. *Layer Normalization*. Konferenciaanyag. NIPS 2016 Deep Learning Symposium. URL: https://arxiv.org/pdf/1607.06450v1.pdf. Elérés dátuma: 2024. június 3.

[97] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao és J. Han. *On the Variance of the Adaptive Learning Rate and Beyond*. Konferenciaanyag. The 8th International Conference on Learning Representations, ICLR 2020. URL: https://iclr.cc/virtual_2020/poster_rkgz2aEKDr.html. Elérés dátuma: 2024. június 3.

[98] D. P. Kingma és J. Ba. *Adam: A Method for Stochastic Optimization*. Konferenciaanyag. The 3rd International Conference on Learning Representations, ICLR 2015. URL: https://arxiv.org/pdf/1412.6980v5.pdf. Elérés dátuma: 2024. június 3.

[99] H. S. Suresha és S. S. Parthasarathy. „Alzheimer Disease Detection Based on Deep Neural Network with Rectified Adam Optimization Technique using MRI Analysis". *2020 Third International Conference on Advances in Electronics, Computers and Communications (ICAECC)*. Bengaluru, India: IEEE Computer Society, 2020, 62–67. DOI: https://doi.org/10.1109/ICAECC50550.2020.9339504.

[100] I. Dimitrovski, I. Kitanovski, D. Kocev és N. Simidjievski. „Current trends in deep learning for Earth Observation: An open-source benchmark arena for image classification". *ISPRS Journal of Photogrammetry and Remote Sensing* 197 (2023), 18–35. DOI: https://doi.org/10.1016/j.isprsjprs.2023.01.014.

[101] M. R. Zhang, J. Lucas, G. E. Hinton és J. Ba. „Lookahead Optimizer: k steps forward, 1 step back". *Advances in Neural Information Process-*

ing Systems. 32. köt. Vancouver, Kanada: Curran Associates Inc., 2019, 9597–9608.

[102] I. Loshchilov és F. Hutter. *Decoupled Weight Decay Regularization*. Konferenciaanyag. The 7th International Conference on Learning Representations, ICLR 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7. Elérés dátuma: 2024. június 3.

[103] I. Goodfellow, Y. Bengio és A. Courville. *Deep Learning*. Cambridge, USA: The MIT Press, 2016.

[104] V. R. Joseph. „Optimal ratio for data splitting”. *Statistical Analysis and Data Mining: The ASA Data Science Journal* 15.4 (2022), 531–538. DOI: https://doi.org/10.1002/sam.11583.

[105] J. Qi, J. Du, S. M. Siniscalchi, X. Ma és C.-H. Lee. „On Mean Absolute Error for Deep Neural Network Based Vector-to-Vector Regression”. *IEEE Signal Processing Letters* 27 (2020), 1485–1489. DOI: https://doi.org/10.1109/LSP.2020.3016837.

[106] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta és A. A. Bharath. „Generative Adversarial Networks: An Overview”. *IEEE Signal Processing Magazine* 35.1 (2018), 53–65. DOI: https://doi.org/10.1109/MSP.2017.2765202.

[107] A. Sauer, K. Chitta, J. Müller és A. Geiger. „Projected GANs Converge Faster”. *Advances in Neural Information Processing Systems*. 34. köt. New York, USA: Curran Associates Inc., 2021, 17480–17492.

[108] Q. Lyu, M. Chen és X. Chen. „Learning color space adaptation from synthetic to real images of cirrus clouds”. *The Visual Computer* 37.8 (2021), 2341–2353. DOI: https://doi.org/10.1007/s00371-020-01990-7.

[109] *MATLAB version 9.9.0.1495850 (R2020b)*. The Mathworks, Inc. Natick, USA, 2020.

[110] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu és X. Zheng. „TensorFlow: A system for large-scale machine learning”. *Proceedings of the 12th USENIX Symposium on Op-*

erating Systems Design and Implementation. Savannah, USA: USENIX Association, 2016, 265–283.

[111] J. M. Twomey és A. E. Smith. „Performance Measures, Consistency, and Power for Artificial Neural Network Models”. *Mathematical and Computer Modelling* 21.1–2 (1995), 243–258. DOI: https://doi.org/10.1016/0895-7177(94)00207-5.

[112] I. Csoba és R. Kunkli. „Fast, GPU-based Computation of Large Point-Spread Function Sets for the Human Eye using the Extended Nijboer-Zernike Approach”. *2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS)*. Debrecen: IEEE Computer Society, 2022, 69–73. DOI: https://doi.org/10.1109/CITDS54976.2022.9914232.

[113] M. Goldstein és R. M. Thaler. „Recurrence techniques for the calculation of Bessel functions”. *Mathematics of Computation* 13.66 (1959), 102–108.

[114] B. Gough. *GNU Scientific Library Reference Manual*. 3. kiad. Network Theory Ltd., 2009.

[115] J. Kessenich, G. Sellers és D. Shreiner. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.5 with SPIR-V*. 9. kiad. Glenview, USA: Addison-Wesley Professional, 2016.

[116] G. Riguer, N. Tatarchuk és J. Isidoro. „Real-Time Depth of Field Simulation”. *ShaderX2: Shader Programming Tips and Tricks with DirectX*. Plano, USA: Wordware Publishing, Inc., 2004, 529–556.

[117] S. Lee, G. J. Kim és S. Choi. „Real-Time Depth-of-Field Rendering Using Anisotropically Filtered Mipmap Interpolation”. *IEEE Transactions on Visualization and Computer Graphics* 15.3 (2009), 453–464. DOI: https://doi.org/10.1109/TVCG.2008.106.

[118] L. McIntosh, B. E. Riecke és S. DiPaola. „Efficiently Simulating the Bokeh of Polygonal Apertures in a Post-Process Depth of Field Shader”. *Computer Graphics Forum* 31.6 (2012), 1810–1822. DOI: https://doi.org/10.1111/j.1467-8659.2012.02097.x.

[119] T. McGraw. „Fast Bokeh effects using low-rank linear filters”. *The Visual Computer* 31.5 (2015), 601–611. DOI: https://doi.org/10.1007/s00371-014-0986-6.

[120] K. Schuster, P. Trettner és L. Kobbelt. „High-Performance Image Filters via Sparse Approximations”. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 3.2 (2020), 14:1–14:19. DOI: https://doi.org/10.1145/3406182.

[121] O. Niemitalo. *Circularly symmetric convolution and lens blur*. URL: http://yehar.com/blog/?p=1495. Elérés dátuma: 2024. június 3.

[122] K. Garcia. *Circular Separable Convolution Depth of Field*. Konferenciaanyag. The 44th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2017. DOI: https://doi.org/10.1145/3084363.3085022.

[123] I. Csoba és R. Kunkli. „Real-Time Rendering of Sphero-Cylindrical Refractive Errors of the Human Eye using Separable Complex Convolution”. *IX. Magyar Számítógépes Grafika és Geometria Konferencia (konferenciakiadvány)*. Budapest: Neumann János Számítógép-tudományi Társaság (NJSZT), 2018, 38–45.

[124] N. Drasdo és C. W. Fowler. „Non-linear projection of the retinal image in a wide-angle schematic eye”. *British Journal of Ophthalmology* 58.8 (1974), 709–714. DOI: https://doi.org/10.1136/bjo.58.8.709.

[125] R. Rosenholtz. „Capabilities and Limitations of Peripheral Vision”. *Annual Review of Vision Science* 2 (2016), 437–457. DOI: https://doi.org/10.1146/annurev-vision-082114-035733.

[126] J. Bickerdt, H. Wendland, D. Geisler, J. Sonnenberg és E. Kasneci. „Beyond the tracked line of sight - Evaluation of the peripheral usable field of view in a simulator setting”. *Journal of Eye Movement Research* 12.3 (2021). DOI: https://doi.org/10.16910/jemr.12.3.9.

[127] L. Franke, N. Hofmann, M. Stamminger és K. Selgrad. „Multi-Layer Depth of Field Rendering with Tiled Splatting”. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1.1 (2018), 6:1–6:17. DOI: https://doi.org/10.1145/3203200.

[128] A. B. Watson és J. I. Yellott. „A unified formula for light-adapted pupil size”. *Journal of Vision* 12.10 (2012), 12:1–12:16. DOI: https://doi.org/10.1167/12.10.12.

[129] 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.

[130] 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.

[131] M. J. Kilgard. *Fast OpenGL-rendering of Lens Flares*. URL: https://www.opengl.org/archives/resources/features/KilgardTechniques/LensFlare/. Elérés dátuma: 2024. június 3.

[132] Y. King. „2D Lens Flare”. *Game Programming Gems*. Newton, USA: Charles River Media, 2000, 515–518.

[133] C. Maughan. „Texture Masking for Faster Lens Flare”. *Game Programming Gems 2*. Newton, USA: Charles River Media, 2001, 474–480.

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

[135] C. Oat. „A Steerable Streak Filter”. *ShaderX3*. 2. köt. Newton, USA: Charles River Media, 2004, 341–348.

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

[137] 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*. New York, USA: ACM, 1995, 317–324. DOI: https://doi.org/10.1145/218380.218463.

[138] 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.

[139] Q. Zheng és C. Zheng. „NeuroLens: Data-Driven Camera Lens Simulation Using Neural Networks”. *Computer Graphics Forum* 36.8 (2017), 390–401. DOI: https://doi.org/10.1111/cgf.13087.

[140] 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.

[141] J. Chaumond. *Realistic Camera - Lens Flares*. URL: http://graphics.stanford.edu/cs348b-07/JulienChaumond/FinalProject. Elérés dátuma: 2024. június 3.

[142] A. Keshmirian. „A Physically-Based Approach for Lens Flare Simulation”. Diplomamunka. University of California, San Diego, USA, 2008. URL: https://escholarship.org/uc/item/5n07m4p6. Elérés dátuma: 2024. június 3.

[143] 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.

[144] M. Lendermann, J. S. Q. Tan, J. M. Koh és K. H. 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.

[145] 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.

[146] 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.

[147] 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.

[148] 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.

[149] H. Joo, S. Kwon, 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.

[150] 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.

[151] 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.

[152] 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.

[153] 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.

[154] D. Johnston, M. Maher és B. Torok. *The Witcher 3: Enabling Next-Gen Effects through NVIDIA GameWorks*. Konferencia-előadás. Game Developers Conference 2014. URL: https://www.gdcvault.com/play/1020664/The-Witcher-3-Enabling-Next. Elérés dátuma: 2024. június 3.

[155] Marion, Gilles and Kramer, Lou. *A guided tour of Blackreef: rendering technologies in Deathloop*. Konferencia-előadás. Game Developers Conference 2022. URL: https://gpuopen.com/gdc-presentations/2022/GDC_A_Guided_Tour_Of_Blackreef.pdf. Elérés dátuma: 2024. június 3.

[156] Synopsys, Inc. *Code V*. URL: https://www.synopsys.com/optical-solutions/codev.html. Elérés dátuma: 2024. június 3.

[157] I. Csoba. „OpenLensFlare: an Open-Source, Lens Flare Designing and Rendering Framework”. *WSCG 2017. Short Papers Proceedings*. Computer Science Research Notes. Plzeň, Csehország: Vaclav Skala–UNION Agency, 2017, 195–203.

[158] I. Csoba és R. Kunkli. „Efficient Parametrization Method for Real-time Lens Flare Rendering Algorithm”. *Graphics and Application—The 12th Asian Forum on Graphic Science (AFGS 2019). Abstracts*. Peking, Kína: China Graphics Society, 2019, 118–120.

[159] M. Laikin. *Lens Design*. 4. kiad. Boca Raton, USA: CRC Press, 2006.

[160] W. J. Smith. *Modern Lens Design*. 2. kiad. New York, USA: McGraw Hill, 2004.

[161] NVIDIA Corporation. *NVIDIA Gameworks*. URL: https://docs.nvidia.com/gameworks/. Elérés dátuma: 2024. június 3.

[162] Advanced Micro Devices, Inc. *AMD FidelityFX*. URL: https://www.amd.com/en/products/graphics/technologies/fidelityfx.html. Elérés dátuma: 2024. június 3.

[163] B. Karadžić. *bgfx*. URL: https://bkaradzic.github.io/bgfx/. Elérés dátuma: 2024. június 3.

[164] O. Cornut. *Dear ImGui*. URL: https://github.com/ocornut/imgui. Elérés dátuma: 2024. június 3.

[165] The Qt Company. *Qt 5*. URL: https://www.qt.io/product/framework. Elérés dátuma: 2024. június 3.

[166] F. L. Pedrotti, L. M. Pedrotti és L. S. Pedrotti. *Introduction to Optics*. 3. kiad. Glenview, USA: Addison-Wesley, 2006.

[167] B. Cao, H. Yang, P. Jiang, W. Caiyang, M. Zhou, S. Mao és Y. Qin. „Modified ray transfer matrix method for accurate non-sequential ray tracing between arbitrary reflective mirrors”. *Optics Express* 28.12 (2020), 17732–17740. DOI: https://doi.org/10.1364/OE.393045.

Függelék

A. Paraméteres szemmodellünk paraméterei

	Param.	Mérték.	$\bar{x}_l$	σ_l	Tartomány	a_l
Szem	T	mm	23,82	0,81	[21,82, 25,82]	2,0
Szaruhártya	T	mm	0,55	0,03	[0,4, 0,7]	32,0
	R^A	mm	(7,81, 7,81)	(0,25, 0,25)	[6,5, 9,81]	1,0
	R^P	mm	(6,44, 6,44)	(0,23, 0,23)	[5,5, 8,44]	1,0
	k^A		-0,29	0,09	[−2,29, 1,71]	1,0
	k^P		-0,34	0,24	[−2,34, 1,66]	1,0
	Φ^A	deg	0,0	0,0	[−45,0, 45,0]	0,1
	Φ^P	deg	0,0	0,0	[−45,0, 45,0]	0,1
	Z^A_{1-4}	mm	0,0	0,0	[−0,10, 0,10]	1,0
	Z^A_{5-6}	mm	0,0	0,0	[−0,05, 0,05]	1,0
Csarnok	T	mm	2,90	0,39	[2,2, 3,5]	1,0
Lencse	D	mm	11,1	0,3	[10,6, 11,6]	1,0
	V	mm^3	160,1	2,5	[153,1, 167,1]	0,1
	k^A		-4,4	1,6	[−10,4, −1,0]	2,0
	k^P		-4,0	2,0	[−10,0, −1,0]	2,0
	Δ	mm	(0,0, 0,0)	(0,0, 0,0)	[−0,8, 0,8]	8,0
	α	deg	(0,0, 0,0)	(0,0, 0,0)	[−7,0, 7,0]	1,0

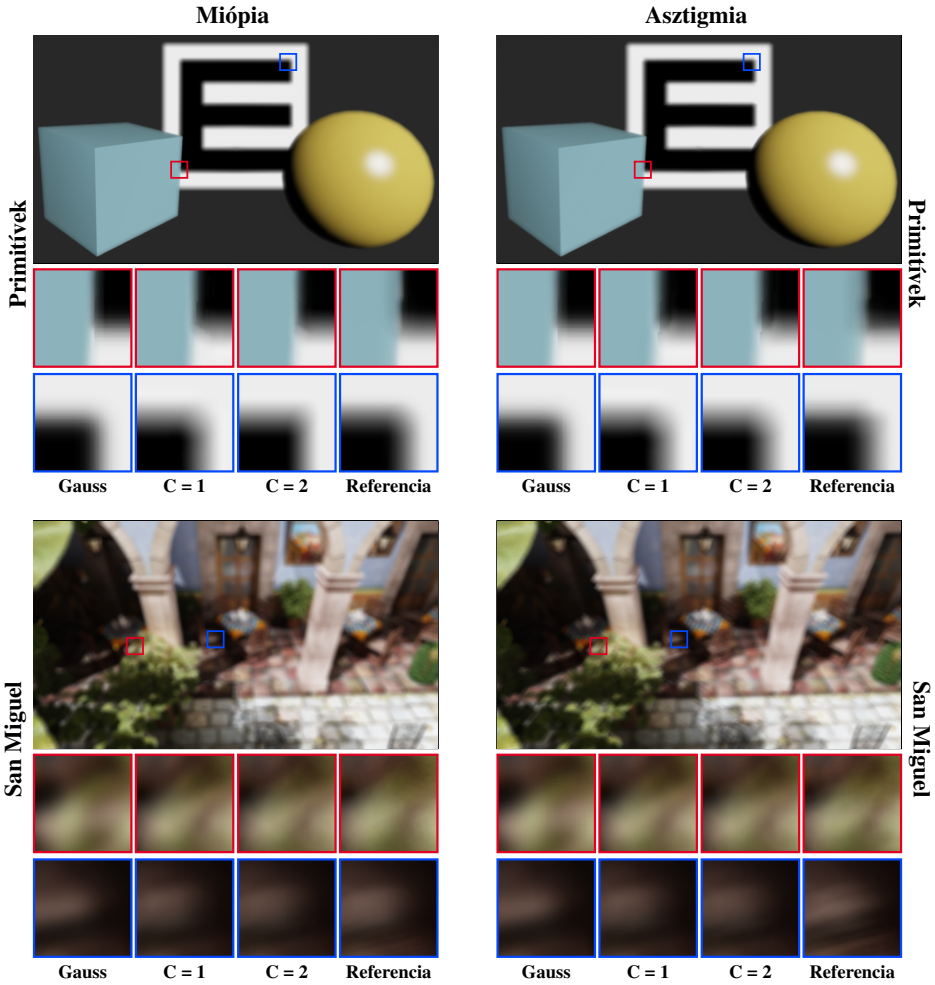
A.1. táblázat. Paraméteres szemmodellünk populációs adatok alapján meghatározott értékei a mintakereséses optimalizáción alapuló szemrekonstrukciós módszerünkhöz. A legfőbb paramétertípusok a vastagság (két szomszédos fénytörő felület közötti távolság, T), az átmérő (D), a térfogat (V), a görbületi sugár (R), a kónikus konstans (k), az ofszetfelület n-edfokú Zernike együtthatói (Z_n), a decentralizáció (Δ), a dőlés (α), valamint az optikai tengely körüli forgatás (Φ). A szaruhártya és a szemlencse esetén a felső index utal a komponens külső (A) és belső (P) felületét reprezentáló elemre.

	Param.	Mérték.	Érték		Param.	Mérték.	Érték
Szaruhártya	T	mm	[0,5, 0,6]	Csarnok	T	mm	[1,5, 4,5]
	R^A	mm	[7,0, 8,6]				
	A^P		[0,9, 1,1]	Lencse	V	mm^3	[150, 165]
	r_R^P		[0,80, 0,85]		k^A		[−10, −1]
	k^A		[−1,50, −0,01]		k^P		[−6, −1]
	k^P		[−1,50, −0,01]		Δ_x	mm	[−0,2, 0,2]
	Φ^A	deg	[−45, 45]		Δ_y	mm	[−0,2, 0,2]
	Φ^P	deg	[−45, 45]		α_x	deg	[−3, 3]
	Z^A_{1-2}	mm	[−0,10, 0,10]		α_y	deg	[−3, 3]
	Z^A_{3-4}	mm	[−0,06, 0,06]				
	Z^A_{5-6}	mm	[−0,03, 0,03]	Szem	T	mm	[22,5, 25,5]

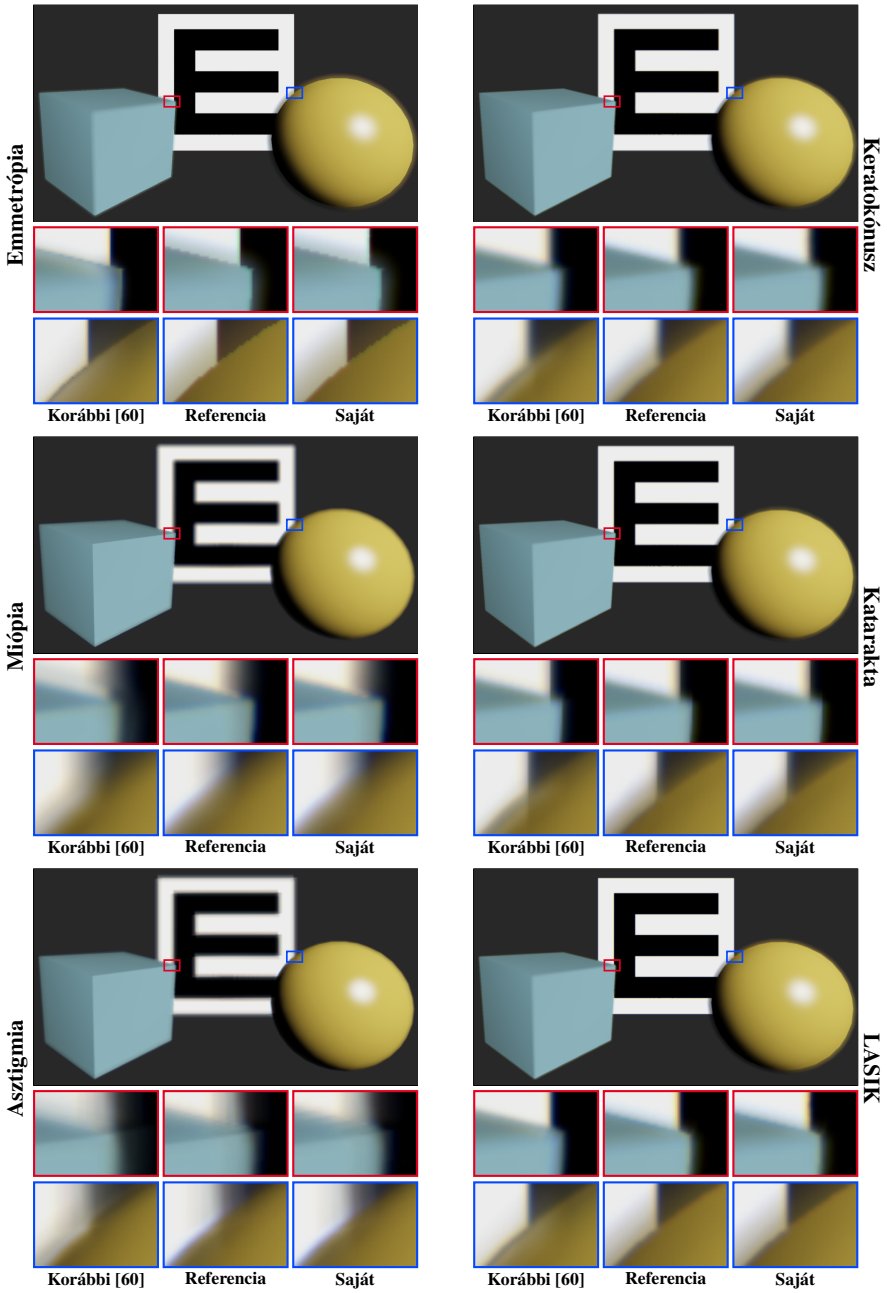
	Param.	Mérték.	Szem	Fókusz	Aberráció
Lencse	D	mm	[9,25, 9,75]	[9,25, 9,75]	[8,4, 9,75]
Pupilla	D	mm	[3, 6]	[2, 7]	[2, 7]
Fókusz	f	D	–	[0,125, 8,125]	–
Sugarak	λ	nm	[450, 900]	–	[450, 700]
	h	deg	–	–	[−45, 45]
	v	deg	–	–	[−26, 26]

A.2. táblázat. Tanító adathalmazok mintavételezéséhez használt paramétertartományok a neurális hálókat alkalmazó módszerünkhöz. A legfőbb paramétertípusok a vastagság (két szomszédos fénytörő felület közötti távolság, T), a görbületi sugár (R), az asztigmia (A), a külső és a belső felület görbületének aránya (r_R), az átmérő (D), a kónikus konstans (k), az optikai tengely körüli forgatás (Φ), az ofszetfelület n-edfokú Zernike együtthatói (Z_n), a térfogat (V), a decentralizáció (Δ), valamint a dőlés (α). A szaruhártya és a szemlencse esetén a felső index utal a komponens külső (A) és belső (P) felületét reprezentáló elemre.

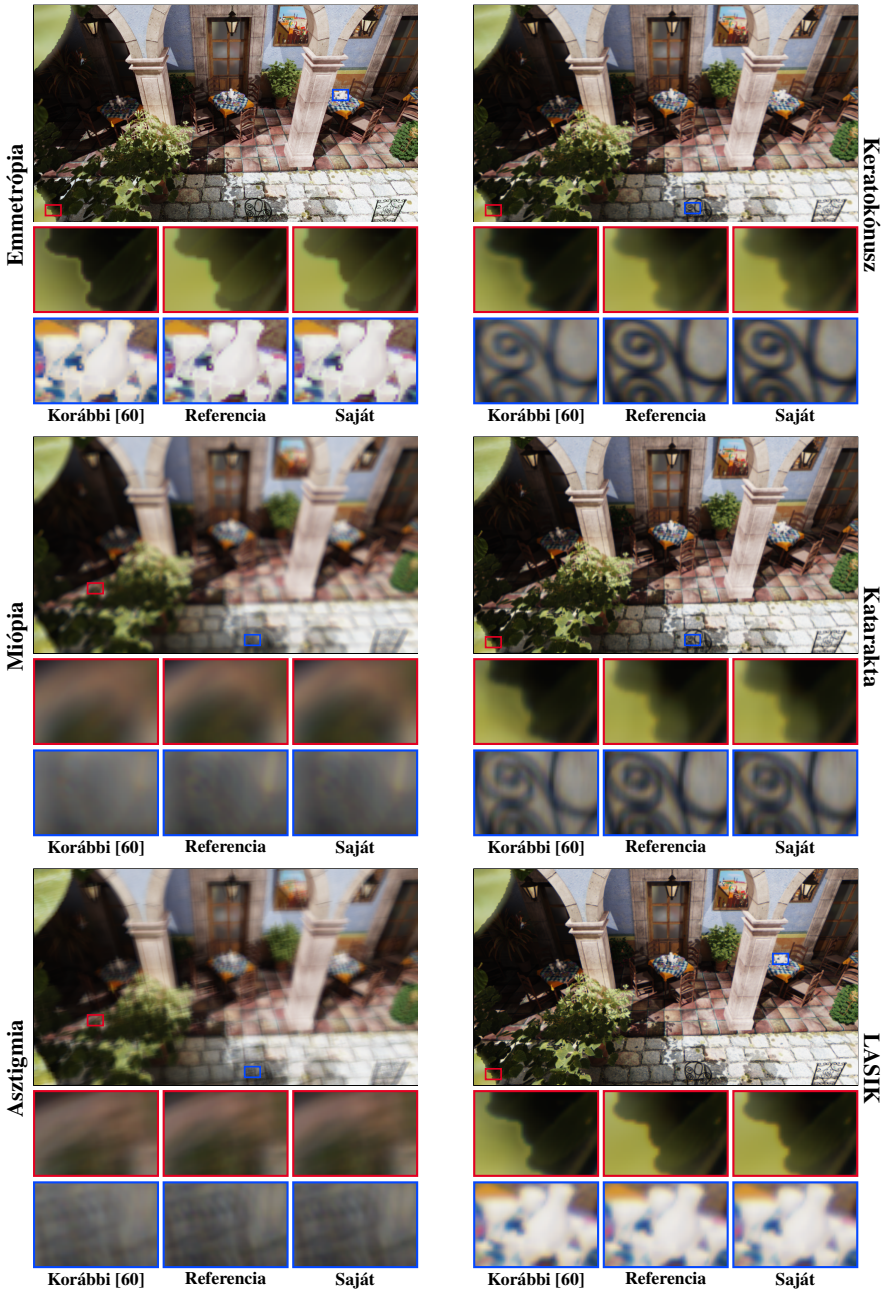
B. Saját módszereinkkel készített látásszimulációk



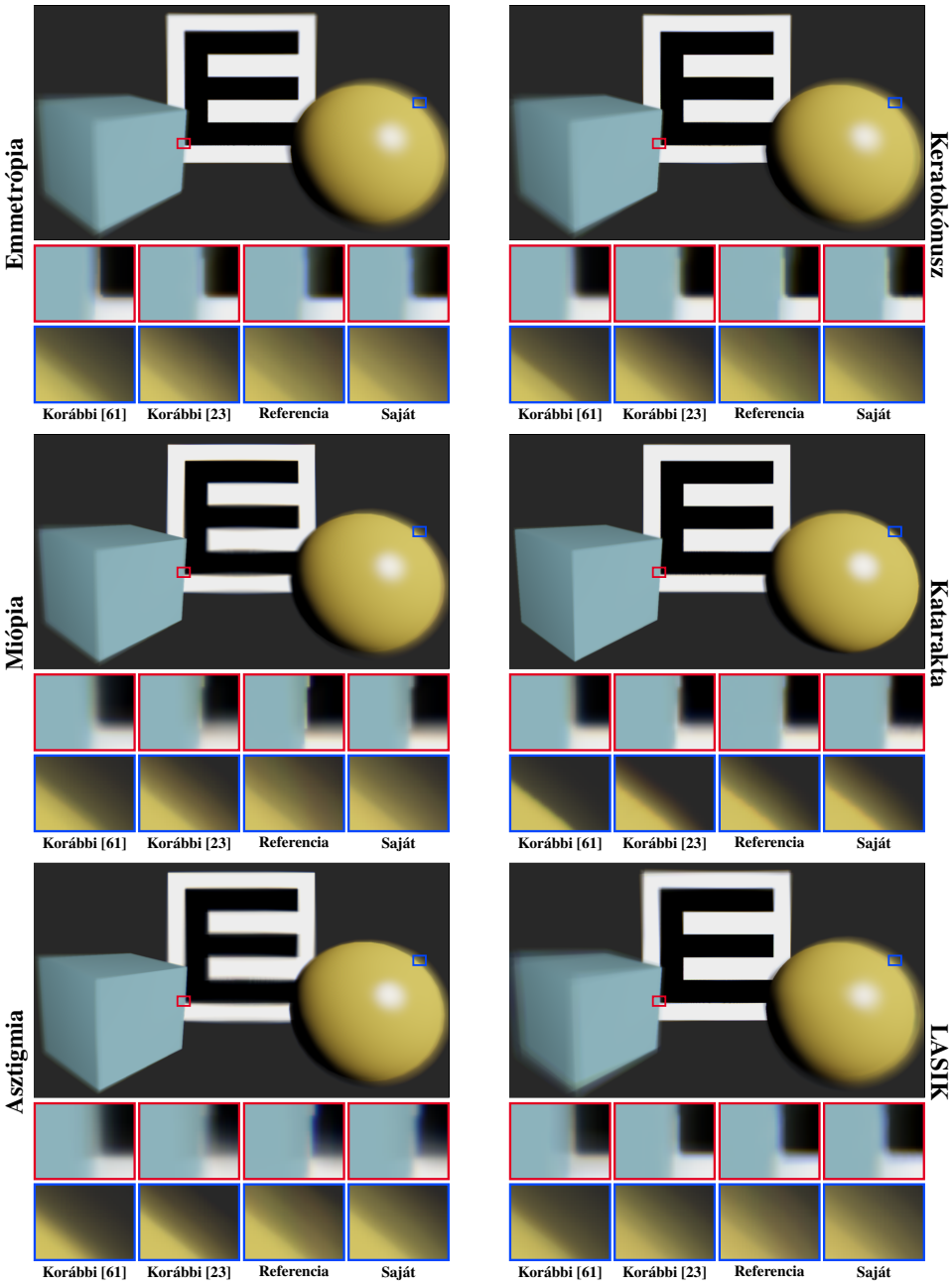
B.1. ábra. Alacsonyrendű aberrációk szimulációja a Gauss-függvénnyel és a komplex fázoros módszerünkkel. A kiemelt régiókon látható, hogy módszerünk pontosabban közelíti a referenciákat, mint a tradicionális Gauss-függvény.



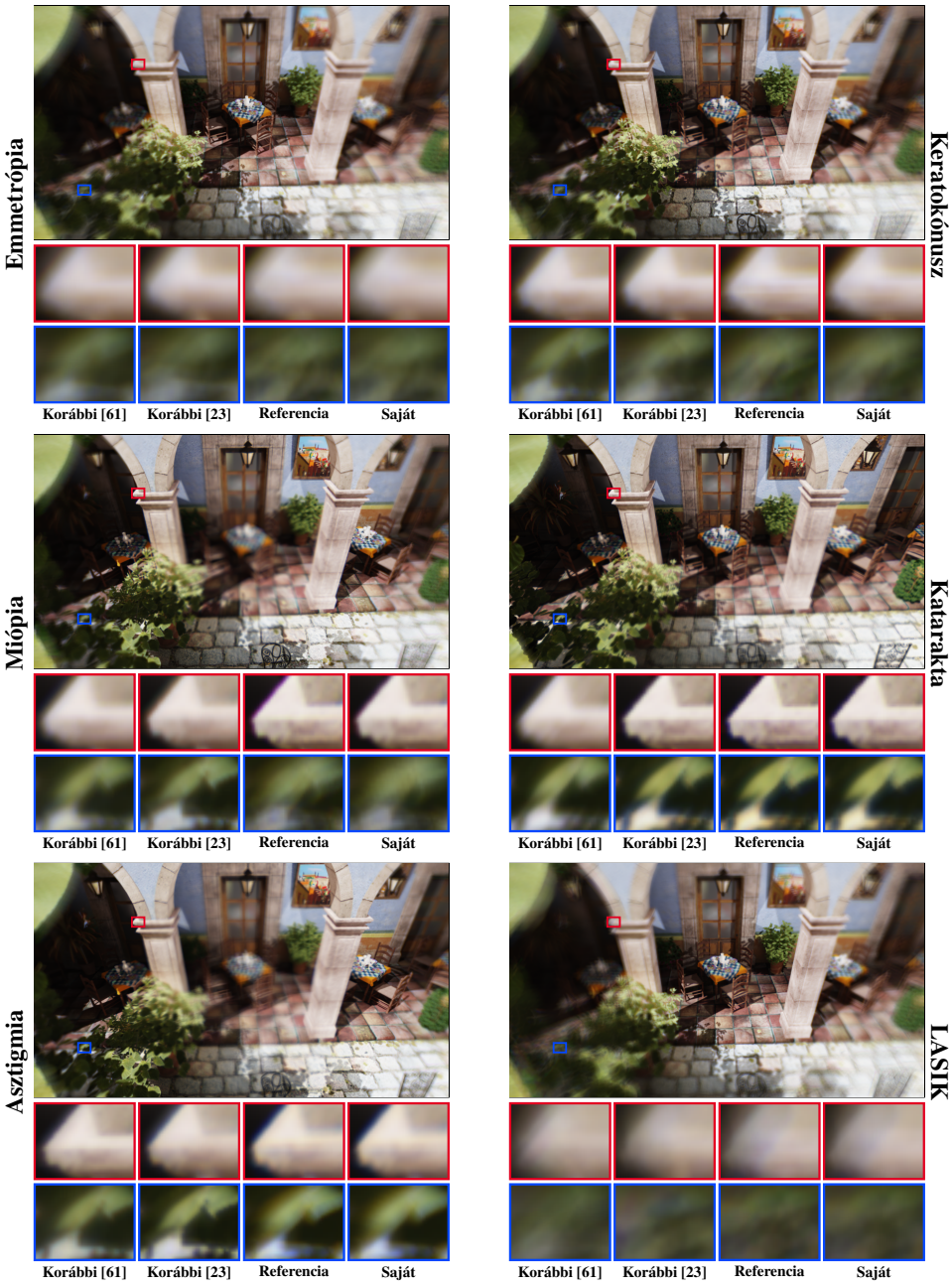
B.2. ábra. Optikaitengely-pontok aberrációinak szimulációja Barsky módszerével [60] és csempézett konvolúciós algoritmusunkkal. A kiemelt régiók alapján megállapítható, hogy eljárásunk magas pontossággal közelíti a referenciákat, és jelentősen kevesebb hibát generál, mint a korábbi módszer.



B.3. ábra. Optikaitengely-pontok aberrációinak szimulációja Barsky módszerével [60] és csempézett konvolúciós algoritmusunkkal. A kiemelt régiók alapján jól látható, hogy módszerünk magas pontossággal, a korábbi módszerhez képest pedig számottevően kevesebb hibával közelíti a referenciákat.



B.4. ábra. Perifériku s látás szimulációja Rodríguez Celaya és mtsai. [61] módszerével, Gonzalez Utrera [23] eljárásával, valamint csempézett konvolúciós algoritmusunkkal. A kiemelt régiókon megfigyelhető, hogy eljárásunk nagy pontossággal, a korábbi módszereknél jelentősen kevesebb hibával közelíti a referenciákat.



B.5. ábra. Perifériku s látás szimulációja Rodríguez Celaya és mtsai. [61] módszerével, Gonzalez Utrera [23] eljárásával, valamint csempézett konvolúciós algoritmusunkkal. A kiemelt régiókon látható, hogy eljárásunk nagy pontossággal, a korábbi módszereknél szignifikánsan kevesebb hibával közelíti a referenciákat.

Publikációs lista

Referált folyóiratcikkek

[F1] **I. Csoba** és R. Kunkli. „Fast rendering of central and peripheral human visual aberrations across the entire visual field with interactive personalization”. *The Visual Computer* 40.5 (2024), 3709–3731. DOI: https://doi.org/10.1007/s00371-023-03060-0.
Folyóirat besorolása: Q2 (impakt faktor: 3,5).

[F2] **I. Csoba** és R. Kunkli. „Rendering algorithms for aberrated human vision simulation”. *Visual Computing for Industry, Biomedicine, and Art* 6 (2023), 5:1–5:25. DOI: https://doi.org/10.1186/s42492-023-00132-9.
Folyóirat besorolása: D1 (impakt faktor: 2,8).

[F3] **I. Csoba** és R. Kunkli. „Efficient Rendering of Ocular Wavefront Aberrations using Tiled Point-Spread Function Splatting”. *Computer Graphics Forum* 40.6 (2021), 182–199. DOI: https://doi.org/10.1111/cgf.14267.
Folyóirat besorolása: D1 (impakt faktor: 2,363).

Konferenciakötetben megjelent cikkek

[K1] **I. Csoba** és R. Kunkli. „Fast, GPU-based Computation of Large Point-Spread Function Sets for the Human Eye using the Extended Nijboer-Zernike Approach”. *2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS)*. Debrecen: IEEE Computer Society, 2022, 69–73. DOI: https://doi.org/10.1109/CITDS54976.2022.9914232.

[K2] **I. Csoba** és R. Kunkli. „Real-Time Rendering of Sphero-Cylindrical Refractive Errors of the Human Eye using Separable Complex Convolution”. *IX. Magyar Számítógépes Grafika és Geometria Konferencia*. Budapest: Neumann János Számítógép-tudományi Társaság (NJSZT), 2018, 38–45.

[K3] **I. Csoba**. „OpenLensFlare: an Open-Source, Lens Flare Designing and Rendering Framework”. *WSCG 2017. Short Papers Proceedings*. Computer Science Research Notes. Plzeň, Csehország: Vaclav Skala–UNION Agency, 2017, 195–203.

További konferencia-előadások

[E1] **I. Csoba** és R. Kunkli. „Efficient Rendering of Ocular Wavefront Aberrations using Tiled Point-Spread Function Splatting”. The 33rd Eurographics Symposium on Rendering (EGSR 2022). Prága, Csehország, 2022.

[E2] **I. Csoba** és R. Kunkli. „Real-Time Rendering of Sphero-Cylindrical Refractive Errors of the Human Eye using Separable Complex Convolution”. 12th Conference of the Hungarian Association for Image Processing and Pattern Recognition (KÉPAF 2019). Debrecen, 2019.

[E3] **I. Csoba**. „Hatékony paraméterkeresési módszer valós idejű lencsefényfolt renderelő algoritmushoz”. XXXIV. Országos Tudományos Diákköri Konferencia. Budapest, 2018.

Poszterprezentációk

[P1] **I. Csoba** és R. Kunkli. „Efficient Parametrization Method for Real-time Lens Flare Rendering Algorithm”. The 12th Asian Forum on Graphic Science (AFGS 2019). Kunming, Kína, 2019.

[P2] **I. Csoba** és R. Kunkli. „Real-Time Rendering of Low-Order Visual Aberrations of the Human Eye using Complex Phasors”. 5th Winter School of PhD Students in Informatics and Mathematics. Debrecen, 2018.

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/170/2024.PL
Tárgy: PhD Publikációs Lista

Jelölt: Csoba István
Doktori Iskola: Informatikai Tudományok Doktori Iskola
MTMT azonosító: 10061645

A PhD értekezés alapjául szolgáló közlemények

Idegen nyelvű tudományos közlemények külföldi folyóiratban (3)

1. **Csoba, I.**, Kunkli, R.: Fast rendering of central and peripheral human visual aberrations across the entire visual field with interactive personalization.
   Visual Comput. 40 (5), 3709-3731, 2024. ISSN: 0178-2789.
   DOI: http://dx.doi.org/10.1007/s00371-023-03060-0
   IF: 3.5 (2022)

2. **Csoba, I.**, Kunkli, R.: Rendering algorithms for aberrated human vision simulation.
   Vis. Comput. Ind. Biomed. Art. 6, 1-25, 2023. EISSN: 2524-4442.
   DOI: http://dx.doi.org/10.1186/s42492-023-00132-9
   IF: 2.8 (2022)

3. **Csoba, I.**, Kunkli, R.: Efficient Rendering of Ocular Wavefront Aberrations using Tiled Point-Spread Function Splatting.
   Comput. Graph. Forum. 40 (6), 182-199, 2021. ISSN: 0167-7055.
   DOI: http://dx.doi.org/10.1111/cgf.14267
   IF: 2.363

Idegen nyelvű konferencia közlemények (3)

4. **Csoba, I.**, Kunkli, R.: Fast, GPU-based Computation of Large Point-Spread Function Sets for the Human Eye using the Extended Nijboer-Zernike Approach.
   In: 2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS) / István Fazekas, IEEE Computer Society, Washington, 69-73, 2022. ISBN: 9781665496537

5. **Csoba, I.**, Kunkli, R.: Real-time rendering of sphero-cylindrical refractive errors of the human eye using separable complex convolution.
   In: IX. Magyar Számítógépes Grafika és Geometria Konferencia. Szerk.: Szirmay-Kalos László, Renner Gábor, Neumann János Számítógép-tudományi Társaság, Budapest, 38-45, 2018. ISBN: 9789633132821

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. **Csoba, I.**: OpenLensFlare: an Open-Source, Lens Flare Designing and Rendering Framework. In: WSCG 2017 Short Paper Proceedings. Eds.: Vaclav Skala, Union Agency, Plzen, Czech Republic, 195-203, 2017, (Computer Science Research Notes, ISSN 2464-4625 ; 2702) ISBN: 9788086943459

Idegen nyelvű absztrakt kiadványok (2)

7. **Csoba, I.**, Kunkli, R.: Efficient Parametrization Method for Real-Time Lens Flare Rendering Algorithm.
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, 118-120, 2019. ISBN: 9787893910319

8. **Csoba, I.**, Kunkli, R.: Real-time rendering of low-order visual aberrations using complex phasors. In: 5th Winter School of PhD Students in Informatics and Mathematics. Ed.: Hudoba Péter, Doktoranduszok Országos Szövetsége, Budapest, 23, 2018. ISBN: 9786155586231

A közlő folyóiratok összesített impakt faktora: 8,663
A közlő folyóiratok összesített impakt faktora (az értekezés alapjául szolgáló közleményekre): 8,663

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

Debrecen, 2024.04.23.

DEBRECENI EGYETEM
Egyetemi és Nemzeti Könyvtár