

TDK-dolgozat

Lévai Balázs László

LED technológiájú közvilágítási lámpatestek fényerejének számítása

Szerző:

Lévai Balázs László
programtervező informatikus MSc
2. évfolyam

Témavezető:

Dr. Bánhelyi Balázs
egyetemi adjunktus

Szegedi Tudományegyetem
Természettudományi és Informatikai Kar
Informatikai Tanszékcsoport
Számítógépes Optimalizálás Tanszék

Bevezetés

Amikor környezetszennyezésről hallunk általában az élővizek vagy levegő szennyezése jut leggyakrabban eszünkbe, azonban a technológia fejlődésével és térnyerésével egyre újabb és újabb nemkívánatos mellékhatással kell számolnunk. Ilyen modern kori jelenség a közvilágítás okozta fényszennyezés [1], mely főleg a fejlett országok nagy népsűrűségű vidékein érzékelhető. Fényszennyezésnek nevezzük azt, ha a világítótestek olyan felületeket is megvilágítanak, amelyre nincs szükség. A jelenség legszemléletesebb példái a világűrből készített, a Föld éjszakai oldalát ábrázoló műholdfelvételek, amelyeken a jelentős köztéri világítás miatt apró pontokként kirajzolódnak a világ nagyvárosai.

Az utcákon és tereken manapság elhelyezett lámpákból nem csak a megvilágítandó területre jut fény. Az égbolt felé irányuló fényszennyezés a városok közelében lehetetlenné teszi a csillagászati megfigyeléseket, mivel érzékelhetetlenné válnak a halványabb égitestek. A feleslegesen megvilágított területek egyben felesleges energiafelhasználást is jelentenek, ami a manapság egyre fontosabbá váló energiatakarékosság témájában vet fel kérdéseket.

A fejlett országokban növekvő hangsúlyt kap a fényszennyezés hatásainak csillapítása. A LED-es közvilágítási lámpatestek alkalmazása remek megoldást kínál erre a problémára [2]. A LED félvezető anyagból készült fényforrás. A kibocsátott fény spektruma az infravörös-től az ultraibolyáig terjed. A LED-ek kibocsátotta fény pontszerű, továbbá fényszórásuk nem gömb karakterisztikájú, ezért a LED-ek által kibocsátott fényt különböző lencsékkel szabályozzák, hogy a kívánt formájúra alakítsák az általuk megvilágított területet. Az egy-egy LED-del megvilágítható térrész meglehetősen kicsi, ezért a LED-es lámpák több, eltérő szögben beállított LED-et tartalmaznak, hogy megfelelően nagy területet tudjanak megvilágítani. A biztonságos közlekedés érdekében törvényi előírások határozzák meg az utak és terek éjszakai megvilágításának mértékét, ezért az egyes LED-eket úgy kell elhelyezni a lámpában, hogy minél egyenletesebb és az elvárt korlátok között legyen a fényerősség a megvilágított területen. Ha figyelembe vesszük, hogy hányféle szempontot kell számításba venni egy ilyen világítótest tervezésekor, könnyen láthatjuk, hogy a minél optimálisabb LED szám és azok irányítottságának meghatározása komplex feladat amelyhez elengedhetetlen, hogy minél pontosabban meg tudjuk határozni a LED-es lámpák megvilágítási képét, azaz a kérdéses utcafelület megvilágítottságát. Dolgozatunkban ezen feladat megoldására mutatunk be egy módszert és annak két implementációját.

Tartalomjegyzék

Bevezetés	1
1. Feladateleírás	3
1.1. A megvilágítandó terület és a lámpaoszlop	3
1.2. A lámpa	4
1.3. Bemenő adatok	4
2. Számítási módszer	6
2.1. Mérési pontok	6
2.2. Egyetlen LED fényereje	7
2.3. A mérési pont iránya	8
2.4. Bilineáris interpoláció	10
3. A számítás megvalósítása	12
3.1. A megvilágítási kép meghatározása processzoron	12
3.2. A CUDA technológia	13
3.3. Az algoritmus megvalósítása grafikus hardveren	14
4. A programok összehasonlítása	15
4.1. A futtatási környezet	15
4.2. A tesztadatok és a kapott eredmények	16
5. Összefoglalás	18
Köszönetnyilvánítás	19
Irodalomjegyzék	20

1. fejezet

Feladateleírás

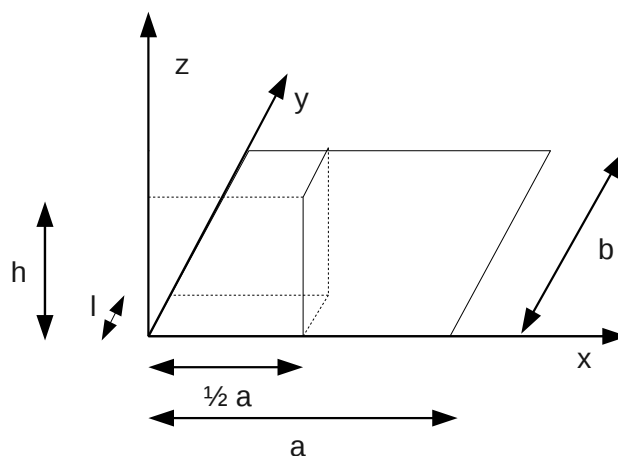
Maga a számítási feladat tömör megfogalmazásban a következőképpen írható le. Adott egy téglalap alakú utcaszakasz és egy LED lámpa, amellyel szeretnénk azt megvilágítani. A feladatunk, hogy kellő pontossággal és részletességgel meghatározzuk az utcaszakaszon a megvilágítottságot, ami a megvilágított felületre eső fénysugár és a megvilágított felület nagyságának hányadosa, ahol a fénysugár a fényforrás minden irányban szétsugárzott fényteltjesítményének összessége. A megvilágítottság mértékegysége LUX ($\frac{lumen}{m^2}$).

1.1. A megvilágítandó terület és a lámpaoszlop

A megvilágítandó területet síkfelületként kezeljük, amely téglalap alakú ismert oldalakkal. A téglalap azon oldalának hosszára, amely mentén a lámpaoszlop elhelyezkedik a téglalap hosszaként (a), a vele szomszédos oldalak hosszára a téglalap szélességeként (b) fogunk hivatkozni.

A feladat végrehajtásához kellő sűrűséggel mérési pontokat veszünk fel a felületen, amelyek egy reguláris négyzetrács rácspontjaira illeszkednek. A megvilágítottságot ezen pontokban fogjuk kiszámítani.

A lámpa egy lámpaoszlopon helyezkedik el, amelynek ismerjük a magasságát (h) és azt, hogy a rajta lévő lámpa milyen mértékben (l) lóg be az utcára. A lámpaoszlop az utca hosszanti oldalának felezőpontjában van.



1.1. ábra. Az utca.

A lámpaoszlop és a mérési pontok helyzetének megadása az utca koordináta rendszerben

történik, amely egy balsodrású, három dimenziós, derékszögű koordináta rendszer, amelynek origója a téglalap alakú felület azon csúcsában helyezkedik el, amely az óramutató járása szerinti első téglalap csúcs a lámpaoszlop után. Az x tengely a téglalap hosszanti oldalával, a y tengely a téglalap másik oldalával, az z tengely a lámpaoszloppal párhuzamos.

1.2. A lámpa

A lámpát pontszerűnek tekintjük. A LED-eket úgy kezeljük, mintha ebben az egyetlen pontban lennének elhelyezve. Ezzel csökkentjük számításaink bonyolultságát amellet, hogy a kapott eredmények pontossága nem változik. A relaxáció a lámpát megépítő és a LED-eket a lámpában elhelyező mérnökökre terhel egyszerű többlet feladatot.

A lámpába helyezett egyes LED-ek pozíciója az előbbieket miatt megegyezik a lámpa pozíciójával. Az elhelyezkedésen kívül a LED-ek másik fontos adata a lámpán belüli irányultságuk, azaz hogy merre világítanak. Ezt az irányt két szög segítségével jellemezzük.

α szög : a lámpába helyezett, alapállapotában az y tengely pozitív felével párhuzamos állású LED horizontális (xy síkkal párhuzamos) elfordulásának szöge. $\alpha \in [0, \pi]$

β szög : az alapállapotából horizontálisan már α szöggel elforgatott LED vertikális (xz síkkal párhuzamos) elfordulásának szöge. $\beta \in [0, \frac{\pi}{2}]$

A szögek értékkészletének magyarázatára a számítási módszer ismertetésekor kerül sor.

A fényerősség kiszámításhoz elengedhetetlen, hogy ismerjük a LED-ek megvilágítási karakterisztikáját. Ez az információ a következő formában áll rendelkezésünkre. Minden LED típusra ismert a LED-től 1 méter távolságra az általa kibocsátott fény keltette megvilágítottság azokban az irányokban, ahol a LED az alapállapotából vertikálisan és horizontálisan is a $[0, \frac{\pi}{2}]$ intervallumbeli, egész értékű, fokban számolt szöggel lett elforgatva. Ez az adatsor egy 91×91 méretű mátrixot alkot.

$$L = \begin{pmatrix} l_{0,0} & \dots & l_{0,90} \\ l_{1,0} & \dots & l_{1,90} \\ \vdots & \ddots & \\ l_{90,0} & \dots & l_{90,90} \end{pmatrix}$$

A mátrix $l_{i,j}$ eleme megadja LUX-ban kifejezve, hogy mekkora a $i^\circ \alpha$ $j^\circ \beta$ irányban 1 méter távolságra a LED által keltett megvilágítottság. Mivel a LED-ek felfelé nem világítanak és a megvilágítási karakterisztikájuk két vertikális tükrösíkkal rendelkezik, ezért elegendő az összes lehetséges irányok ezen adatsor által reprezentált egy nyolcadának ismerete, mert a megfelelő tükrözésekkel megkaphatjuk a hiányzó térrészek megvilágítottság értékeit is.

1.3. Bemenő adatok

A számítási feladat megoldásához ismerjük mind a lámpaoszlop mind az utca eddig tárgyalt adatait. Ezeket globális információkként kezeljük, melyek minden függvény és algoritmus számára szabadon hozzáférhetőek, nem szükséges külön paraméterként átadni őket.

BEMENŐ ADAT	JELÖLÉS
Utcaszakasz hossza	a
Utcaszakasz szélessége	b
Lámpaoszlop magassága	h
Lámpa belógása	l

1.1. táblázat. *A számítási probléma utcaszakaszának és lámpaoszlopának adatai*

Tisztában vagyunk a lámpába helyezett LED-ek számával és azok irányával, amelyek a korábban bemutatott rotációs szögekkel adottak. Minden LED-nek ismerjük a típusát és a típushoz tartozó megvilágítási karakterisztikát, amely egy mátrix formájában áll rendelkezésünkre. A LED-ekkel kapcsolatos információk szintén globális adatok.

BEMENŐ ADAT	JELÖLÉS
A LED-eket tartalmazó tömb	$LEDS$
A LED-eket tartalmazó tömb mérete	$LEDS.size$
Egy LED horizontális elforgatásának szöge	$LED.\alpha$
Egy LED vertikális elforgatásának szöge	$LED.\beta$
Egy LED típusának karakterisztikus mátrixa	$LED.L$

1.2. táblázat. *A számítási probléma LED-jeinek adatai*

2. fejezet

Számítási módszer

A számítási feladatunk a bemenő adatok ismeretében mérési pontok generálásából majd ezen pontok megvilágítottságának meghatározásából áll. Az első lépés könnyű feladat. A számítások bonyolultságáért és költségeiért a második rész végrehajtása a felelős.

2.1. Mérési pontok

A mérési pontokat egy 1 méter oldalhosszúságú négyzetrács rácspontjaiban vesszük fel. A pontok pozícióját az utca koordináta-rendszerben adjuk meg. Ennek megfelelően a bemenőadat struktúrák mellé a számítások elvégzéséhez a mérési pontokat a következő módon reprezentáljuk.

MÉRÉSI PONTOK ADATAI	JELÖLÉS
A mérési pontokat tartalmazó tömb	MP
A mérési pontokat tartalmazó tömb mérete	$MP.size$
Egy mérési pont x koordinátája	$MeasurePoint.x$
Egy mérési pont y koordinátája	$MeasurePoint.y$
Egy mérési pont z koordinátája	$MeasurePoint.z$
Egy mérési pont megvilágítottsága	$MeasurePoint.L$

2.1. táblázat. A számítási probléma mérési pontjainak adatai

Az első mérési pont pozíciója a megvilágítandó terület hosszától és szélességétől függően

$$\begin{aligned}MP[1, 1].x &= \frac{\{a\}}{2} \\MP[1, 1].y &= \frac{\{b\}}{2} \\MP[1, 1].z &= 0\end{aligned}$$

ahol $\{x\}$ az x tört része. A i -edik sorban és j -edik oszlopban lévő mérési pont koordinátái ennek megfelelően

$$MP[i, j].x = \frac{\{a\}}{2} + i - 1$$

$$MP[i, j].y = \frac{\{b\}}{2} + j - 1$$

$$MP[i, j].z = 0$$

Összesen $[a] \cdot [b]$ mérési pontot veszünk fel, ahol $[x]$ x egész részét jelöli. A mérési pontokat generáló függvény a korábban bevezetett jelöléseknek megfelelően az MP tömbben adja vissza a kimenetét.

Függvény 1 *GENERATE MEASURE POINTS()*

```

1:  $x_{max} := [a]$ 
2:  $y_{max} := [b]$ 
3: for  $i := 1$  to  $x_{max}$  do
4:   for  $j := 1$  to  $y_{max}$  do
5:      $MP[i, j].x = \frac{\{a\}}{2} + i - 1$ 
6:      $MP[i, j].y = \frac{\{b\}}{2} + j - 1$ 
7:      $MP[i, j].z = 0$ 
8:      $MP[i, j].L = 0$ 
9:   end for
10: end for
11:  $MP.size := (x_{max}, y_{max})$ 
12: return  $MP$ 

```

2.2. Egyetlen LED fényereje

A számítási feladat elemi részét képezi, hogy meghatározzuk egyetlen LED fénye mekkora megvilágítottságot eredményez egy mérési pontban. A megvilágítottságot az alábbi egyenlettel számíthatjuk ki

$$E = \frac{L_m}{d^2}$$

ahol L_m a LED keltette megvilágítottság a LED-től mért 1 méteres távolságban a mérési pont irányába nézve, d a LED és a mérési pont távolsága. A bemenő adatok mellé az egyenlet alkalmazásához meg kell határozni, hogy

- a LED-ből nézve milyen irányba esik a mérési pont;
- a mérési pont irányának ismeretében, mekkora a LED keltette megvilágítottság a kérdéses irányban 1 méterre a LED-től;
- milyen távol van egymástól a LED és a mérési pont.

Az első két részfeladat megoldása összetett, ezért ezeket később részletesen is tárgyaljuk. A távolság meghatározására a jól ismert Euklédesszi távolság formulát használjuk, ami nem szorul további magyarázatra. A részfeladatot megoldó *CALCULATE LUX* függvény két paraméterrel rendelkezik. Az első egy mérési pont, a második egy LED, amely keltette megvilágítottságot szeretnénk meghatározni a mérési pontban.

Függvény 2 *CALCULATE LUX*($m : MeasurePoint, L : LED$)

- 1: $(\alpha_m, \beta_m) := \text{CALCULATE ANGLES}(m, L)$
 - 2: $L_m := \text{BILINEAR INTERPOLATION}(L, \alpha_m, \beta_m)$
 - 3: $distance := \text{sqr}t(\text{pow}(m.x - a/2, 2) + \text{pow}((m.y - l), 2) + \text{pow}(m.z - h, 2))$
 - 4: **return** $L_m / \text{pow}(distance, 2)$
-

A *CALCULATE ANGLES* függvény az előbbi három pontos lista első, a *BILINEAR INTERPOLATION* a második pontját valósítja meg. A fejezet hátralevő részében ezen függvények számításainak elméleti hátterét és magukat a függvényeket fogjuk ismertetni.

2.3. A mérési pont iránya

Először meghatározzuk a LED irányvektorának az utca síkjával vett dőfési pontját. Ez a LED α és β szögeiből továbbá a lámpa és a mérési pont koordinátáiból könnyen kiszámítható a következő képleteket használva.

$$\begin{aligned}d_x &= \frac{a}{2} + \cos(\alpha + \frac{\pi}{2}) \cdot h \cdot \tan(\beta) \\d_y &= l + \sin(\alpha + \frac{\pi}{2}) \cdot h \cdot \tan(\beta) \\d_z &= 0\end{aligned}$$

A dőfési pont koordinátaival a birtokunkban (d_x és d_y) elkészítjük azt a B bázis mátrixot, amely az utca koordináta rendszerből a LED koordináta rendszerbe transzformálja a vektorokat. A LED koordináta rendszerének origója a lámpa, tengelyei alapállapotban párhuzamosak az utca koordináta rendszer megfelelő tengelyeivel. A koordináta rendszer aktuális állását megkaphatjuk, ha először horizontálisan α majd vertikálisan β szöggel elforgatjuk az alapállapotból.

Ha a dőfési pont éppen a lámpa alatt helyezkedik el ($d_x = \frac{a}{2}, d_y = l$), akkor a bázis az identikus mátrix. Ellenkező esetben a bázis z vektora,

$$\begin{aligned}b_z &= \left[\frac{(d_x - \frac{a}{2})}{t}, \frac{(d_y - l)}{t}, \frac{-h}{t} \right] \\t &= \sqrt{(d_x - \frac{a}{2})^2 + (d_y - l)^2 + h^2}\end{aligned}$$

ahol t a dőfési pont és a lámpa távolsága. Ha a LED α szöge nem $\frac{\pi}{2}$, akkor a bázis x vektorát kiszámíthatjuk a következő képletek segítségével,

$$\begin{aligned}b_x &= \left[\text{sig} \cdot \frac{(m_x - \frac{a}{2})}{t}, \text{sig} \cdot \frac{(d_y - l)}{t}, \text{sig} \cdot \frac{-h}{t} \right] \\t &= \sqrt{(m_x - \frac{a}{2})^2 + (d_y - l)^2 + h^2} \\m_x &= \frac{(d_x - \frac{a}{2})^2 + 2 \cdot (d_y - l)^2 + 2 \cdot h^2 - d_x^2 + (\frac{a}{2})^2}{-2 \cdot d_x + 2 \cdot \frac{a}{2}} \\ \text{sig} &= \begin{cases} 1, & \text{ha } m_x < \frac{a}{2} \\ -1, & \text{egyébként} \end{cases}\end{aligned}$$

ahol t a bázis x vektorának és a $z = 0$ sík metszéspontjának a távolsága, m_x e metszéspont x koordinátája és sig egy előjelet tartalmazó érték, amely biztosítja, hogy a bázis első két vektora ne forduljon el π -vel a z tengely körül, ha $d_x - \frac{a}{2}$ előjelet vált.

A bázis y vektorát megkaphatjuk, ha $\frac{\pi}{2}$ -vel elforgatjuk az x vektort a z vektor körül.

$$b_y = (R \cdot x^T)^T$$

$$R = \begin{pmatrix} b_{z,1}^2, & b_{z,1} \cdot b_{z,2} - b_{z,3}, & b_{z,1} \cdot b_{z,3} - b_{z,2} \\ b_{z,1} \cdot b_{z,2} + b_{z,3}, & b_{z,2}^2, & b_{z,2} \cdot b_{z,3} - b_{z,1} \\ b_{z,1} \cdot b_{z,3} - b_{z,2}, & b_{z,2} \cdot b_{z,3} + b_{z,1}, & b_{z,3}^2 \end{pmatrix}$$

Ha a LED α szöge $\frac{\pi}{2}$, akkor az előbbi esettel analóg módon járunk el, de ekkor az x és z bázisvektorok első koordinátája megegyezik, ezért az x vektor második koordinátáját kell meghatározunk.

$$b_x = \left[sig \cdot \frac{(d_x - \frac{a}{2})}{t}, sig \cdot \frac{(m_y - l)}{t}, sig \cdot \frac{-h}{t} \right]$$

$$t = \sqrt{(d_x - \frac{a}{2})^2 + (m_y - l)^2 + h^2}$$

$$m_y = \frac{(d_y - l)^2 + 2 \cdot (d_x - \frac{a}{2})^2 + 2 \cdot h^2 - d_y^2 + (l)^2}{-2 \cdot d_y + 2 \cdot l}$$

$$sig = \begin{cases} 1, & \text{ha } m_y < \frac{a}{2} \\ -1, & \text{egyébként} \end{cases}$$

A y vektor meghatározása szintén ugyanazon módszerrel történik, mint az előző esetben, azonban kissé eltérő R forgatási mátrixot kell alkalmaznunk.

$$b_y = (R \cdot x^T)^T$$

$$R = \begin{pmatrix} b_{z,1}^2, & b_{z,1} \cdot b_{z,2} + b_{z,3}, & b_{z,1} \cdot b_{z,3} - b_{z,2} \\ b_{z,1} \cdot b_{z,2} - b_{z,3}, & b_{z,2}^2, & b_{z,2} \cdot b_{z,3} + b_{z,1} \\ b_{z,1} \cdot b_{z,3} + b_{z,2}, & b_{z,2} \cdot b_{z,3} - b_{z,1}, & b_{z,3}^2 \end{pmatrix}$$

Miután kiszámítottuk a B bázismátrixot, a mérési pont v_l irányvektorát a LED koordináta rendszerében könnyedén megkaphatjuk egy egyszerű mátrix szorzás segítségével az utca koordináta rendszerben adott v_s irányvektorból

$$v_l = B \cdot v_s$$

$$v_s = \left[m_x - \frac{a}{2}, m_y - l, m_z - h \right]$$

ahol m_x , m_y és m_z a mérési pont megfelelő koordinátái. A v_l vektor ismeretében a mérési pont irányát megadó, a LED-ek irányának megadása során már megismert α és β szögek.

$$\alpha = \arctan \left(\frac{v_{l,2}}{v_{l,1}} \right)$$

Ha α nagyobb, mint π , akkor tükrözzük az origóra, azaz $\alpha = \pi - \alpha$. Az esetleges tükrözés után a β szöget megkaphatjuk a következő képletek alkalmazásával.

$$\beta = \arctan \left(\frac{\text{sig} \cdot \sqrt{v_{l,1}^2 + v_{l,2}^2}}{v_{l,3}} \right)$$

$$\text{sig} = \begin{cases} -1, & \text{ha } \alpha\text{-t tükrözni kellett} \\ 1, & \text{egyébként} \end{cases}$$

A *CALCULATE ANGLES* függvény ezen számításokat megvalósítva határozza meg, hogy paramétereként átadott LED-ből a szintén átadott mérési pont milyen szögben látszik. A számítások el nem hangzott háttéréről többek között Megyesi László Lineáris algebra [5] c. könyvében olvashatunk.

2.4. Bilineáris interpoláció

Ha tudjuk, hogy egy LED-ből nézve mekkora α és β horizontális és vertikális szögben látszik egy mérési pont, akkor megbecsülhetjük, hogy a LED-től 1 méteres távolságban mekkora megvilágítottságot kelt a LED a kérdéses irányban. Mivel a megvilágítottság mérési adatai fokban számítva csak az egész értékekre állnak rendelkezésünkre, ezért szükség van egy módszerre, amellyel a nem egész értékű α és β irányokban is képesek vagyunk megfelelő pontosságú közelítést adni a megvilágítottságra.

A bilineáris interpoláció a lineáris interpoláció kiterjesztése két változós, regulárisan mintavételezett függvények interpolációjához. Az eljárás két egymás után elvégzett lineáris interpoláción alapszik, azonban maga az eljárás nem lineáris. Az interpolációról, mint matematika problémáról Virágh János Numerikus matematika [6] könyvében olvashatunk bővebben.

Legyen $P = (x, y)$ az a pont, amelyben szeretnénk az f két változós függvényünk értékét meghatározni. Legyenek x_1, x_2, y_1, y_2 értékek olyanok, hogy $x_1 \leq x \leq x_2$ és $y_1 \leq y \leq y_2$ továbbá a $Q_{1,1} = (x_1, y_1), Q_{1,2} = (x_1, y_2), Q_{2,1} = (x_2, y_1), Q_{2,2} = (x_2, y_2)$ pontok alkossanak egy négyzetet. Ezen négy pontot felhasználva két lépésben határozzuk meg az $f(x, y)$ közelítő értéket.

1. Lineáris interpoláció x mentén, ahol $R_1 = (x, y_1), R_2 = (x, y_2)$.

$$f(R_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{1,1}) + \frac{x - x_1}{x_2 - x_1} f(Q_{1,2})$$

$$f(R_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{2,1}) + \frac{x - x_1}{x_2 - x_1} f(Q_{2,2})$$

2. Lineáris interpoláció y mentén.

$$f(P) = \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2)$$

Legyenek α és β a mérési pont már említett horizontális és vertikális irányának szögei, továbbá $a_1, a_2, x_1, x_2, y_1, y_2$ vegyék fel értéküket a következőképpen.

$$a_1 = \begin{cases} [\pi - \alpha], & \text{ha } \alpha > \frac{\pi}{2} \\ [\alpha], & \text{egyébként} \end{cases}, a_2 = a_1 + 1$$

$$x_1 = [\alpha], x_2 = x_1 + 1$$

$$y_1 = [\beta], y_2 = y_1 + 1$$

Az a_1 és a_2 értékek bevezetésére azért van szükség, mert az L megvilágítottság mátrix indexeléséhez az x_1 és x_2 értékek nem megfelelőek, mivel az adatok tárolása kihasználja a LED megvilágítási karakterisztikájának tükörszimmetriáját, azonban az x_1 és x_2 értékek ezt nem tárolják. A bilineáris interpolációt használva a megvilágítottság az ismert irányban és távolságra az alábbi módon megkapható.

$$\begin{aligned} L_{R_1} &= \frac{x_2 - \alpha}{x_2 - x_1} L_{a_1, y_1} + \frac{\alpha - x_1}{x_2 - x_1} L_{a_2, y_1} \\ L_{R_2} &= \frac{x_2 - \alpha}{x_2 - x_1} L_{a_2, y_1} + \frac{\alpha - x_1}{x_2 - x_1} L_{a_2, y_2} \\ L_P &= \frac{y_2 - \beta}{y_2 - y_1} L_{R_1} + \frac{\beta - y_1}{y_2 - y_1} L_{R_2} \end{aligned}$$

Amennyiben az indexeléshez használt a_2 és y_2 közül bármelyiknek kívül esne az értéke a megengedett indextartományon, a megvilágítottságot 0-nak vesszük.

A *BILINEAR INTERPOLATION* függvény az előbbiekben ismertetett módszerrel határozza meg a mérési pontokban az egyetlen LED által okozott megvilágítottságot.

3. fejezet

A számítás megvalósítása

3.1. A megvilágítási kép meghatározása processzoron

Az előző fejezetben áttekintettük, hogyan határozható meg egy mérési pontban valamelyik LED által keltett megvilágítottság. A részproblémát megoldó függvényt felhasználva megkonstruálhatjuk az utcaszakasz teljes megvilágítottságát kiszámító algoritmust. Az utca megvilágítási képétől elvárható, hogy szimmetrikus legyen $x = \frac{a}{2}$ síkra, ezért a LED lámpa tervezésekor eleve csak az egyik „utcafélbe” világító LED-ek adatait kapjuk meg, mivel az utca másik fele pontosan ugyanannyi LED által megvilágított azzal az eltéréssel, hogy azok iránya tükörszimmetrikus a már említett síkra.

A fentebb említettek ismeretében a teljes utcaszakasz megvilágítottságát meghatározó algoritmus csak az utca egyik felébe eső mérési pontokra végzi el a már bemutatott számításokat. Ezután kijelöli minden ismert megvilágítottságú mérési pont tükörképét és a korábbi lépésben kapott adatok egyszerű átmásolása révén ezen másik utcafélbe eső pontok megvilágítási értékeit is meghatározza.

Függvény 3 *CALCULATE ILLUMINANCE()*

```
1:  $MP := \text{GENERATE MEASURE POINTS}()$ 
2: for  $i := 1$  to  $MP.size[0]$  do
3:   for  $j := 1$  to  $MP.size[1]$  do
4:     for  $k := 1$  to  $LEDS.size$  do
5:        $E[i, j] := E[i, j] + \text{CALCULATE LUX}(MP[i, j], LEDS[k])$ 
6:     end for
7:   end for
8: end for
9: for  $i := 1$  to  $MP.size[0]$  do
10:  for  $j := (MP.size[1] + 1)/2$  to  $MP.size[1]$  do
11:     $E[i, j] := E[i, MP.size[1] - j + 1]$ 
12:  end for
13: end for
```

A megvilágítási kép elkészítése során több olyan számítás egymásutáni elvégzése szükséges, amelyek végeredményei nincsenek egymásra befolyással, egyszerű, főleg aritmetikai műveletek elvégzéséből tevődnek össze, bemenő adataikat nem változtatják meg, mindezek által alkalmasak a párhuzamosításra. A fejezet további részében az algoritmus párhuzamosítással történő gyorsításával fogunk foglalkozni.

3.2. A CUDA technológia

A CUDA [7] az nVidia által kifejlesztett párhuzamos programozási architektúra, amely a több maggal rendelkező grafikus processzorokban, az ún. GPU-kban rejlő képességeket aknázza ki. A korábbi szokásokkal szakítva az nVidia mérnökei a két megszokott *vertex* és *pixel* csővezeték helyett egyetlen általános célú csővezetékét építettek a grafikus hardverbe, amely lehetővé teszi a programozók számára, hogy a megoldandó feladat típusától függetlenül a hardver minden számítási egységének kapacitását igénybe vehessék. Az architektúra tervezésekor az elsődleges célpont a sokrétű felhasználhatóság volt. A nyilvánvaló grafikus feladatok ellátásán kívül a gyártó nem titkolt szándéka az általa gyártott kártyák gazdasági és tudományos számítások elvégzésére történő használata. Az architektúra több, jelenleg is intenzíven fejlődő technológia alapjául szolgál, mint az *OpenCL*, a *Direct Compute* vagy az általunk is használt *CUDA C/C++* programozási nyelv és paradigma.

A CUDA C/C++ egy hatékony és könnyen kezelhető interfészt ad a névadó architektúra használatához. A filozófiája középpontjában a programozó munkájának tehermentesítése áll. A processzormagok számát, az elérhető lebegő pontos számítási pontosságot és a többi, a programírás során előre nem ismerhető részletet afféle középréteg módjára elfedi a programozó előtt lehetővé téve, hogy a programozó kizárólag a párhuzamosítási feladat megoldására koncentrálhasson anélkül, hogy ehhez ismernie kellene a program futtatásához később felhasználandó hardver technikai paramétereit.

Szintaxisának a C nyelv az alapja, amit kibővíti néhány a C++-ban bevezetett kényelmi megoldással, illetve a párhuzamos programozáshoz szükséges új operátorokkal és kulcsszavakkal.

A párhuzamosítással megoldandó feladatok kódolása során abból az alapfeltevésből indulhatunk el, hogy végtelen sok számítási egységet használhatunk fel, amelyek egymástól függetlenül, de akár összehangoltan is képesek működni. A rendelkezésünkre álló és az általunk használt processzor magok száma közötti különbség lekezelése az architektúra feladata. A CUDA C/C++ programok mind a CPU-t mind a GPU-t használhatják. A grafikus hardveren kétféle függvény futtatható. A *global* függvények a CPU-n futó függvényekből lehet meghívni, a *device* függvények kizárólag másik device vagy global függvényekből érhetők el. Valahányszor a GPU-n futtatunk egy függvényt az architektúra egy új szálat indít. A szálak ötféle indexértékkel rendelkeznek. Az egyes indexek által reprezentált dimenziók mentén a szálak nagyobb számítási egységekbe rendezhetők. Minden szál egy-egy *blokk*nak nevezett háromdimenziós egységbe tartozik, a blokkok egy kétdimenziós rácsot alkotnak. Az egy blokkon belüli szálaknak lehetőségük van osztott memóriaterületet használni a kommunikációra. Az egyes szálakon belül elérhető az adott szál különböző indexeinek értéke, amelyet az összetettebb feladatok különböző szálak által végrehajtható részfeladatokra bontásának koordinálására használhatunk. Bár az egyes szálak egymástól függetlenül hajtják végre a feladataikat, ami különböző végrehajtási időket jelent, lehetőség van a probléma megoldásának egyes lépéseit szinkronizálni azáltal, hogy olyan pontokat kódolunk a programba, ahol a szálak összevárják egymás működését és nem folytatják a futást, amíg minden szál el nem jut a kérdéses kódrészletig.

A grafikus kártya memóriájának kezelése tér el a legjobban a C/C++-ban megismert dinamikus memóriakezeléstől. Az egy dimenziós tömbök kezelése megegyezik a már ismert pointereken keresztül történő memóriaeléréssel. A kettő- és háromdimenziós tömbök létrehozására speciális beépített függvények állnak rendelkezésünkre. Ennek oka, hogy a gyors és optimális memóriaelérés érdekében a többdimenziós tömbök, mint összetett adatstruktúrák belső memória címei regiszterekben tárolódnak, ezért nem érhetők el közvetlenül.

A CUDA technológia működéséről és használatának részleteiről átfogó képet kaphatunk Jason Sanders és Edward Kandrot munkájából [4].

3.3. Az algoritmus megvalósítása grafikus hardveren

A számítási feladatot megoldó algoritmus tervezésekor is már sejthető volt az, ami a processzoron futó program profilozása után kiválóan látszott, hogy a futási idő tetemes része a mérési pontok kiértékeléséből áll, ezért kínálta magát a lehetőség, hogy megpróbáljunk valamilyen módon gyorsítani ezen. A megvilágítási képet meghatározó program tervezésekor már figyelembe vettük a kihasználható teljesítmény optimalizálási lehetőségeket, ezért kizárólag egy új technológia vagy architektúra jöhetett számításba a program felgyorsítására, így esett a választás a program CUDA-képes grafikus hardverre történő átírására.

A CPU-ra tervezett algoritmusmag végrehajtásának párhuzamosítását tűztük ki célul az új program írásakor. Az egyes mérési pontok megvilágítottságának meghatározását egy-egy, egymással párhuzamosan futó szál hajtja végre, amelyek az *EVALUATE MEASURE POINT* eljárást hajtják végre. Ez a függvény ugyanazt a módszert használja, mint amit korábban megismertünk.

Függvény 4 *EVALUATE MEASURE POINT*(*MP* : array, *i* : integer, *j* : integer)

```
1: for k := 1 to LEDS.size do  
2:   E[i, j] := E[i, j] + CALCULATE LUX(MP[i, j], LEDS[k])  
3: end for
```

A mérési pontok generálását és a megvilágítási kép tükrözését továbbra is egyetlen szál végzi a mérési pontok megvilágítási értékeinek kiszámítása után.

Függvény 5 *CALCULATE ILLUMINANCE* ()

```
1: MP := GENERATE MEASURE POINTS()  
2: for i := 1 to MP.size[0] do  
3:   for j := 1 to MP.size[1] do  
4:     START NEW THREAD EVALUATE MEASUREPOINT(MP, i, j)  
5:   end for  
6: end for  
7: WAIT FOR EVALUATIONS  
8: for i := 1 to MP.size[0] do  
9:   for j := (MP.size[1] + 1)/2 to MP.size[1] do  
10:    E[i, j] := E[i, MP.size[1] - j + 1]  
11:   end for  
12: end for
```

A párhuzamosításból eredő nyilvánvaló gyorsuláson kívül olyan plusz feladatok is hárulnak ránk, mint a CPU és a GPU memóriája között a LED-ek és az utca adatainak átvitele. A többlet adatmozgatás nyilvánvalóan növeli a szükséges futási időt. Erre mindösszesen kétszer van szükség. Először a lámpák és az utca adatainak a GPU memóriájába másolásakor, másodsor, miután a grafikus hardver elvégezte a számításokat, az utca megvilágítottsági adatait kell visszamásolnunk a CPU memóriájába. A mérési pontok első felének megvilágítottságát kiszámító szálak futásának befejeződését meg kell várnunk mielőtt elvégezhetnénk a kiszámított értékek tükrözését. A párhuzamos programozási paradigmát mélyebben megismerhetjük Iványi Antal művéből [3].

4. fejezet

A programok összehasonlítása

Az első algoritmus párhuzamosított változatának megtervezése után, a korábbi programfutások profilozása és a CUDA technológiában rejlő lehetőségek ismeretében arra számítottunk, hogy közepes méretű feladatokon átlagosan kétszer olyan gyorsan fog működni az algoritmus grafikus hardvert használó változata. Az új architektúrára való áttérés ezen várakozásainknál is jobb hatékonyság növekedést eredményezett.

4.1. A futtatási környezet

A megvilágítottság kiszámítását több teszt utcára is elvégeztük külön-külön a két megvalósításra. A programok hatékonyságának mérésére a következő paraméterekkel rendelkező processzort használtuk fel.

INTEL CORE I3-370M	
Magok száma	2
Memória	3 GB
Órajel	2.4 GHz
Utastításkészlet	64 bit

4.1. táblázat. A számításokhoz felhasznált processzor

A számítási módszer CUDA-t használó megvalósításának futtatásához a következő tulajdonságokkal bíró grafikus kártyát használtuk fel.

NVIDIA GEFORCE GT 335M	
Magok száma	72
Memória	1025 MB
Órajel	450 MHz
Memóriabusz mérete	128 bit

4.2. táblázat. A számításokhoz felhasznált processzor

A két programot 64 bites Windows 7 operációs rendszeren futtattuk.

4.2. A tesztadatok és a kapott eredmények

Többféle utcaméretet, elérendő minimális megvilágítottsági szintet és LED számot használtunk a tesztek elkészítése során. Egy-egy teszt utcára a számításokat 500 alkalommal végeztük el, majd ez alapján számítottunk átlagos futási időket. A teszt utcákat és a két megvalósítás összehasonlítását a következő táblázatokban foglaltuk össze.

TESZT UTCÁK					
#	Utca hossza	Utca szélessége	Lámpaoszlop magassága	LUX	LED-ek száma
1.	30 m	10 m	4.5 m	6	27
2.	30 m	10 m	4.5 m	8	37
3.	30 m	10 m	4.5 m	10	48
4.	35 m	20 m	4.5 m	6	38
5.	40 m	20 m	4.5 m	8	68
6.	50 m	25 m	4.5 m	8	100

4.3. táblázat. A programok hatékonyságának összehasonlításához használt teszt adatok

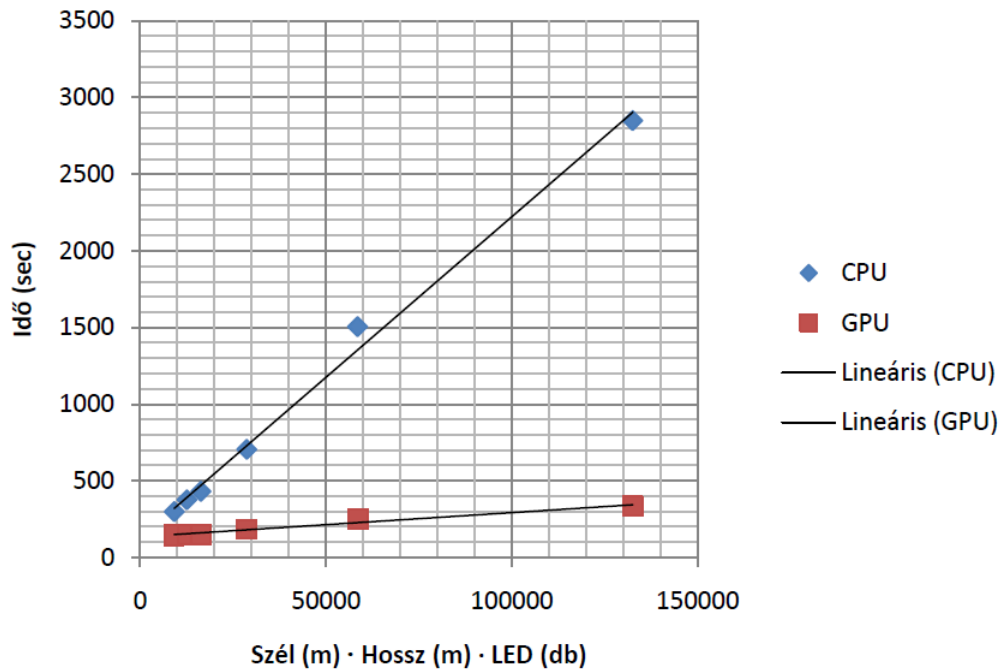
A vizsgált utcák mindegyikén gyorsabbnak bizonyult a grafikus hardvert használó megvalósítás, ezért kizárólag a párhuzamosított és a csak processzoron futó változat futási idejeinek hánynadosát, jelenítjük meg.

A KÉT MEGVALÓSÍTÁS ÖSSZEHASONLÍTÁSA	
#	A párhuzamosított programmal elért gyorsulás
1.	2.076
2.	2.506
3.	2.854
4.	3.831
5.	5.996
6.	8.529

4.4. táblázat. Az egyes teszt utcákon elért gyorsulás a grafikus hardvert használó programmal

Jól megfigyelhető, hogy a megvilágítottság meghatározásához szükséges számítások számának növekedésével a GPU-t használó megvalósítás arányaiban egyre gyorsabban képes elvégezni a kitűzött feladatot, mint a számításokat kizárólag a CPU-n végző program. Ez összhangban áll a korábbi feltevésünkkel, miszerint a feladat skálázódásával arányaiban jelentősen többet nyerünk a megvilágítási kép meghatározásának felgyorsításával, mint amennyi többletköltség jelentkezik a GPU-n történő számítások előkészítésével.

A számítási feladat mérete az utca szélességétől és hosszúságától továbbá a megvilágításhoz felhasznált LED-ek számától egyaránt lineárisan függ. Mindezeket figyelembe véve megvizsgálhatjuk, hogy a $(\text{utca szélessége}+1) \cdot (\text{utca hossza}+1) \cdot (\text{LED-ek száma})$ szorzat milyen viszonyban áll a futási idővel.



4.1. ábra. Az algoritmusok futási idejei a feladat méretének függvényében

Az ábrát megvizsgálva azonnal észrevehető a két mennyiség közötti lineáris összefüggés.

$$CPU = 0,02094 \cdot x + 127$$

$$CPU_{GPU} = 0,00158 \cdot x + 134$$

ahol x a fenti szorzattal megkapható feladatméret, CPU jelöli a párhuzamosítás nélküli, a GPU a párhuzamosított program futási idejét. Megfigyelhető, hogy nagyjából 2-3 perc a fix költsége a számításoknak. Az ezen felüli futási időt tekintve határértékben sikerült nagyjából 13-szoros ($0,02094/0,00158$) gyorsulást elérnünk az eredeti algoritmus párhuzamosításával. Ez a hatékonyság növekedés csak a nagyobb feladatok esetében jelentkezik igazán. A meghatározandó megvilágítási kép méretnövekedése valamilyen szinten a fix költségek növekedését is okozza, de ez elhanyagolható a többi számítás miatti futásidő megemelkedéséhez képest.

Mennyit számít a gyakorlatban mindez a gyorsulás, merülhet fel bennünk a kérdés? Egy LED-es utcai lámpa tervezése során akár 10000-es nagyságrendben is előfordulnak különböző lehetséges lámpabúra változatok, eltérő számú és állású LED-del, amelyek megvilágítási képeit meg kell határoznunk. Egy átlagos méretű LED lámpa megtervezése az első algoritmust felhasználva nagyjából 3 órát igényel. Ha a párhuzamosított változatot alkalmazzuk a megvilágítási képek meghatározására, akkor ugyanezt a feladatot 20-30 perc alatt is elvégezhetjük.

5. fejezet

Összefoglalás

Az ipar és a tudomány fejlődésével a mindennapi élet problémáira a korábban drága vagy nem hatékony alternatívák rentábilis és fenntartható megoldásokká léphetnek elő. Dolgozatunkban egy új közvilágítási technológiával kapcsolatos, a LED-es utcai lámpák tervezése során felmerülő probléma megoldását és ezen megoldás továbbfejlesztését mutattuk be.

Az első fejezetben magát a problémát ismertettük annak bemenő adataival és elvégzendő feladat pontos megadásával. Bevezettük a később ismertetésre kerülő algoritmusokban felhasznált konstansok jelöléseit.

A második fejezetben részletekbe menően bemutattuk a problémát alkotó számítási feladatot. Ennek keretein belül pontosítottuk, hogy mit értünk a megvilágítási kép meghatározásán, amelyhez a megvilágítandó területen ekvidisztáns mérési pontokat vettünk fel. Az eredeti feladatot elemi, egyenrangú részfeladatokra bontottuk, nevezetesen egy-egy mérési pont megvilágítottságának meghatározására, majd megadtuk egy-egy ilyen részfeladat megoldásának lépéseit. A fejezet nagyobb része ezen lépések elméleti hátterének, mint például a bilineáris interpoláció, leírásából áll.

A harmadik fejezetben egy, a korábbiakban kidolgozott számítási módszert felhasználó algoritmust adtunk meg. A feladat szerkezetét vizsgálva felismertük, hogy az eredeti algoritmus hatékonysága bizonyos részsámítások párhuzamosíthatóságából fakadóan növelhető. Ennek elvégzéséhez bemutattunk egy új technológiát, az nVIDIA által kifejlesztett CUDA-t, majd megadtuk az algoritmusunk egy új, párhuzamosított változatát.

A negyedik fejezetben az algoritmusban eszközölt változtatások tényleges hatékonyságát vizsgáltuk néhány példa utca megvilágítási képének meghatározásán keresztül. Az eredmények áttekintése után kimutattuk, hogy a párhuzamosítás határértékben egy 13-szor gyorsabb algoritmust eredményezett.

Köszönetnyilvánítás

Szeretnék köszönetet mondani témavezetőmnek Dr. Bánhelyi Balázsnak. Szaktudása és mindenre kiterjedő precizitása nélkül e dolgozat nem készülhetett volna el.

Irodalomjegyzék

- [1] <http://fenyszennyezés.hu>, Fényszennyezés, elérve 2012.04.10.
- [2] Gary Gereffi, Kristen Dubay, Marcy Lowe, Manufacturing Climate Solutions Carbon-Reducing Technologies and U.S. Jobs Center on Globalization, Governance & Competitiveness, Duke University, 2008.
- [3] Iványi Antal, Párhuzamos algoritmusok, ELTE Eötvös Kiadó, Budapest, 2005.
- [4] Jason Sanders, Edward Kandrot, CUDA by Example, an introduction to general-purpose CPU-programming, Addison-Wesley, New York, 2010.
- [5] Megyesi László, Lineáris algebra, Polygon, 2007.
- [6] Virágh János, *Numerikus matematika*, JATEPress, Szeged, 1997.
- [7] http://www.nvidia.com/object/cuda_home_new.html, A CUDA technológia hivatalos oldala, elérve 2012.04.10.