Képek manipulálása – a matematika, ami mögötte van

Írta: Zahalka Bence ©2011

Tartalomjegyzék

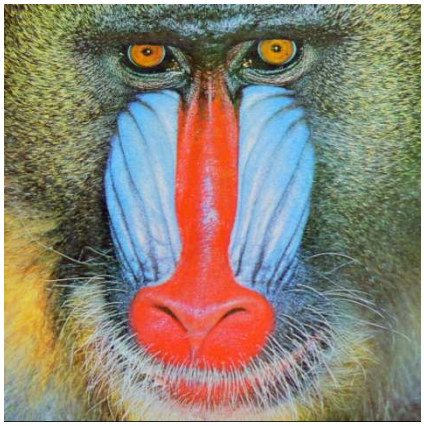
Bevezető

A hagyományoktól eltérően ez a cikk nem valamilyen képszerkesztő program használatát szándékozik bemutatni, sokkal inkább azt próbálja megmutatni, hogy a képek manipulálását hogyan lehet megfogalmazni és alkalmazni egy (ingyenes) matematikai szoftver segítségével.

A cikkben szereplő Octave programra véletlenül akadtam rá és rögtön megtetszett, miután kiderült, hogy az általam már ismert és a főiskolán használt (cserébe viszont drága – a tanuló verzió is 99 dollár a cikk megírásakor!) MATLAB programmal teljesen kompatibilis! Még a MATLAB-ban készített programok is futnak benne (olykor esetleg egy kis módosítás után).

A képmanipuláló „algoritmusok” bemutatásához az ilyenkor közkedvelt lena.bmp és mandrill.bmp képeket fogom használni.

1 – lena.bmp 512x512*24bbp



2 – mandrill.bmp 512x512*24bpp

Alapvetően szükséges ismeretek

Ahhoz, hogy nekiláthassunk, meg kell értenünk, hogyan is kezeli, mit lát a program a képből, hiszen a számítógép nem egy nőt, vagy egy majmot lát a képen, hanem képpontokat, s ráadásul a képpontokat is több szám jellemezheti.

Jelen esetünkben színes (24bites) képekkel fogom a példákat szemléltetni, de ugyanígy lehetne fekete-fehér (1bites), szürkeárnyalatos (8bites), vagy akár más színmélységű színes képeket is használni. Ezekről a varázsszavakról pár szót említsünk meg. 1bit = 2 választási lehetőség, azaz igen, vagy nem. A 8 bites kép azt jelenti, hogy a színpalettájának melyik színét viseli az adott képpont. Ezért

szeretik a 8 bites színes képeket egy egyszerű palettamódosítással szürkeárnyalatossá tenni, ami rettentő sebességnövekedést jelent nagyméretű képek esetében…

Azon képeknél, amikkel most foglalkozni fogok, a 24bit azt jelenti, hogy minden egyes színre (R – vörös, G – zöld, B – kék) 8 bitnyi információ jut, azaz az adott színjellemző egy 0-255 közti skálán képviselteti magát. Érdemes lehet megemlíteni, hogy léteznek magasabb bitszámú képek is, a következő lépés a 32 bit-es RGBA (vagy RGBAlpha), ahol a 24 bitnyi színinformáció mellé bekerül egy negyedik, átlátszósági információ is, illetve a CYMK színmodellű képek is 32biten tárolódnak. Fényképezőgépeknél fordul elő még a 40, vagy magasabb bitértékű képek, ezeknél a fényképező formátuma válogatja, milyen információt takarnak és milyen sorrendben a bitek.

Még egy kicsit számolgassunk mit is jelentenek ezek a számok. Példának vegyük a lena.bmp-t, ami egy 512 pont széles 512 pont magas 24 bit színmélységű kép, ami azt jelenti, hogy minden pontot 24 bit ír le, tehát 6291456 bit = 768 kilobájt helyet foglal a kép tömörítetlenül.

Ennyi technika után most már jöhet egy kicsit izgalmasabb rész, azaz vajon mit lát a választott matematikai program Lénából?

Első körben – ha még nem tettük meg –, a http://www.gnu.org/software/octave/index.html honlapra látogassunk el és töltsük le és telepítsük az Octave aktuális verzióját (telepítéskor az image library kelleni fog). Miután ez megtörtént, az indításkor jöhet a meglepetés: bizony ez NEM GRAFIKUS FELÜLETŰ program! Itt mindent parancsokkal irányítunk, de természetesen van grafikus kimenetünk is, amikor arra szükségünk lesz.

Ahhoz, hogy betöltsünk egy képet, egy egyszerű utasítást kell csupán kiadnunk.

lena = imread(”f:\\worktemp\\Octave\\lena.bmp”);

Ez a rész egy kis magyarázatra szorul. Első fontos megemlítenivaló, hogy balra megfeleltetés van az Octave nyelvében, azaz a baloldalon megadott „lena” változónak adjuk meg az imread fv. eredményét, végül pontosvesszővel lezárjuk az utasítást. A pontosvessző utasítja a programot, hogy az eredményt ne ömlessze azonnal a képernyőre (ami sokszor nagyon zavaró lehet!), ha elhagyjuk, akkor az enter lenyomására azonnal elkezdi a lena.bmp-ből kiolvasott mátrixokat kiírni. Azt is fontos megjegyezni, hogy kettős perjelet kell használni az elérési útvonal megadásakor, mivel az egyes perjelet ún. escape karakterként értelmezi az őt közvetlenül követő karakterrel együtt (így lehet pl. soremelést iktatni egy szövegbe, ha arra van szükség).

Ha meg akarjuk jeleníteni a változónk értékét (azaz a beolvasott képet), akkor egyszerűen be kell írnunk a nevét és leütnünk az entert, s már meg is jelenik (ha pontosvesszőt rakunk a végére, akkor nem jelenik meg).

Ekkor tapasztalhatjuk, hogy a képünkből bizony egy mátrix lett…

lena =

ans(:,:,1) =

Columns 1 through 15:

193 198 195 195 200 201 199 202 197 198 196 192 192 193 190
196 197 194 197 202 197 194 202 203 201 196 189 187 187 185
199 192 200 197 193 198 195 196 197 193 190 191 195 195 190
200 193 200 198 194 198 195 195 197 193 191 192 196 196 191
200 195 200 198 194 197 196 195 197 194 192 192 195 196 192
199 195 198 198 194 195 195 193 196 192 190 191 193 193 191
198 196 196 198 194 193 196 192 191 188 188 188 190 191 189
194 193 192 195 192 190 195 190 189 187 185 187 188 188 188
192 192 190 194 191 189 196 190 189 187 186 185 188 189 189
191 192 189 194 191 189 196 190 189 188 187 186 189 189 189
192 191 196 193 184 187 191 183 186 186 189 183 190 184 189
194 193 191 189 188 191 189 182 182 182 186 181 187 183 186
191 193 189 188 191 190 185 182 183 184 187 185 189 187 189
186 194 193 188 190 186 182 186 183 185 188 188 191 190 191
187 193 192 190 192 185 182 188 179 182 184 187 188 189 187
192 190 188 188 192 190 184 185 179 184 185 190 189 192 188
195 188 187 191 190 189 187 185 182 187 187 194 191 195 190
195 187 191 194 186 184 190 189 179 185 184 192 188 192 186
lines 1-24 -- (f)orward, (b)ack, (q)uit

Ha jobban megfigyeljük, akkor egy hasznos információ elleshető a kimenetünkből.

ans(:,:,1)=

Vagyis, „ans” az előző változó érték, avagy előző eredmény, éppúgy, mint a jobb számológépeken, majd megmutatja hogyan is kell egy mátrix (több dimenziós tömb) elemeire hivatkozni.

mátrixváltozóneve(első dimenzióból megjeleníteni kívánt elemek, második dimenzió, …)

Az ember azonban jobban szereti látni, hogy mi is történik, ezért a képet meg tudjuk jeleníteni számok helyett „normálisan” is.

imshow(változónév)

Ha egy mátrix méreteire vagyunk kíváncsiak, akkor nincs más dolgunk, mint kiadni a következő parancsot.

size (változónév)

Amire válaszul lena esetében azt kapjuk,

ans =
512 512 3

Mivel nekünk a képünk három dimenzióval rendelkezik (szélesség, magasság, színcsatornák száma), így egy adott képpont színét (színcsatornákra bontva) így kaphatjuk meg:

változónév(oszlopkoordináta, sorkoordináta,:)

Ha egy változóra már nincs többé szükségünk, akkor a clear paranccsal törölhetjük.

clear változónév

Ebben a cikkben jelenleg csak a mátrixműveletekkel végrehajtható szűrőkről írok, így már csak egyetlen fontos elemet kell megismernünk, hogy könnyen dolgozhassunk a képeinken. Ez pedig nem más, mint a conv2 parancs, ami mátrix-konvolúciót (egy speciális mátrix műveletet) végez.

Ahhoz, hogy magunk definiáljunk kézzel egy többdimenziós tömböt, ügyelnünk kell a vesszőinkre, zárójeleinkre és pontosvesszőinkre. Példának okáért egy 3x3-as emboss szűrő konvolúciós mátrixát így kell megadni:

```
octave-3.2.4.exe:16> emboss2_3x3 = [-1,0,0;0,1,0;0,0,0]
emboss2_3x3 =

 -1  0  0
  0  1  0
  0  0  0
```

Jól látható, hogy megadáshoz szögletes zárójelet használtam, majd az egyes sorokban lévő elemeket egymástól vesszővel, a sorokat pedig pontosvesszővel választottam el, minek egy 3x3-as mátrix lett az eredménye.

Egy megfelelő mátrixot ezután képként kimenteni az imwrite függvénnyel lehet.

imwrite(változónév, ”célfájl”);

A képek módosítása – alapműveletek

Invertálás

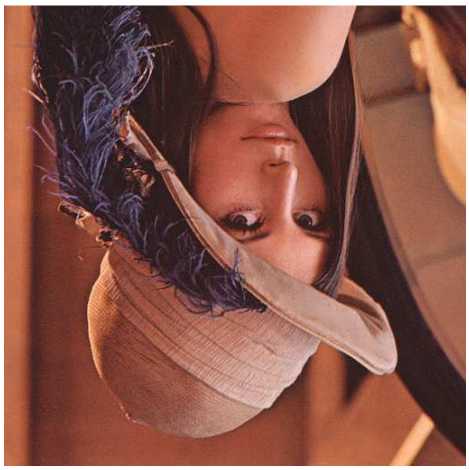
Az invertálás azt jelenti, színek megfordítása az ellentettjükre. Itt mindössze azt kell tudnunk, hogy a 8 bites csatornáink miatt a mátrixunkban (ezentúl csak így fogok a képre hivatkozni) az értékek 0-tól 255-ig terjedhetnek, tehát az invertálás mindössze egy sima kivonás.

```
kimenet = 255 – lena;
imshow(kimenet)
```

Tükrözés, forgatás derékszöggel

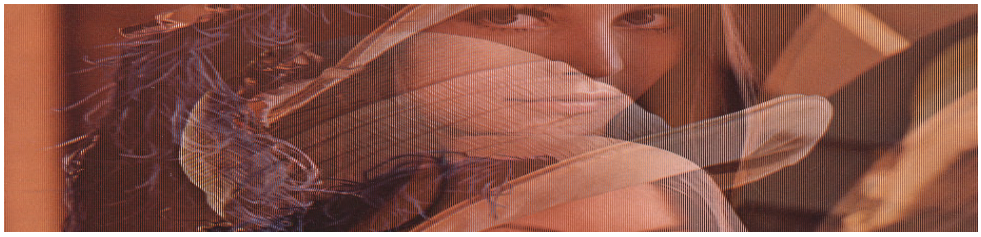
Mivel mátrixokról beszélünk, így az Octave beépített mátrixmódosító függvényeivel rendkívül gyorsan tudunk tükrözni, vagy 90°-al forgatni. Mivel azonban a beépített függvények többnyire csak 2D mátrixokkal működnek, így előfordulhat, hogy az egyes csatornákra külön-külön kell elvégezni a kívánt műveletet.

```
kimenet (:,:,1) = flipud( lena (:,:,1) );
kimenet (:,:,2) = flipud( lena (:,:,2) );
kimenet (:,:,3) = flipud( lena (:,:,3) );
imshow(kimenet)
```



A többi hasonlóan érdekes fv.:

rot90(mátrix)
fliplr(mátrix)
reshape(mátrix, új első dimenzió, új második dimenzió, …)

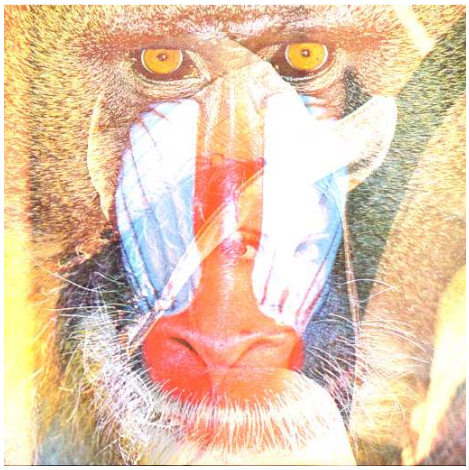


lena kicsit átrendezve…

Két kép összeolvasztására használható különböző módszerek

Összeadás

kimenet = lena + mandrill;
imshow(kimenet)



Kivonás

kimenet = lena – mandrill;
imshow(kimenet)

Szorzás

Itt, bár a neve erre utal, nem egyszerű szorzásról van szó, ugyanis a kapott eredményt be is kell igazítani a 8 bites tartományunkba (0..255), tehát a végén még egy osztást is el kell végeznünk.

$$\frac{p_1 * p_2}{255}$$

Ez a művelet sajnos már több odafigyelést igényel, mivel a mátrixok szorzása nem ugyanaz, mint a mátrix elemeinek szorzása, márpedig itt az elemeken kell végrehajtanunk a műveletet, azaz a képpontokat egyesével kell manipulálni.
Ehhez viszont már egy kicsit programoznunk kell, azaz meg kell ismerkednünk a ***for*** ciklussal.

```
for x = 1:512
    for y = 1:512
        kimenet (x,y,1) = uint16( lena (x,y,1) ) * uint16( mandrill (x,y,1) ) /255 ;
        kimenet (x,y,2) = uint16( lena (x,y,2) ) * uint16( mandrill (x,y,2) ) /255 ;
        kimenet (x,y,3) = uint16( lena (x,y,3) ) * uint16( mandrill (x,y,3) ) /255 ;
    endfor
endfor
```

A fenti kód némi magyarázatra szorul. Az Octave automatikusan számoláskor a megfelelő értéktartományba helyezi az eredményt. Túlcsordulás helyett automatikusan csonkít a tartomány határára. Így tehát, ha megmaradunk az uint8 (8 biten ábrázolható pozitív egész) számok körében, akkor a szorzásunk eredménye maximum 255 lehet (uint8 tartományon ábrázolva az 1000 éppúgy 255, mint a 260), vagyis teljesen kifehérítjük a képet, majd az osztással teljesen befeketítjük (mivel minden érték 0, vagy 1 lesz). Emiatt át kell térnünk egy bővebb tartományba, ami a 16 biten ábrázolható pozitív egész számok halmaza, s végül az eredményt ismét beszorítjuk az uint8-as tartományba az osztással.

Az Octave azonban a ciklusok ismerete nélkül is ajánl megoldást arra a problémára, hogy a mátrixok elemein végezzük el a műveleteket. Ha egy pontot teszünk a szorzás, vagy az osztás jel elé, akkor a mátrixok megfelelő elemein fogja végrehajtani az adott műveletet, ami a processzor utasításkészletre optimalizált kód miatt jelentősen gyorsabb a ciklushasználatnál.

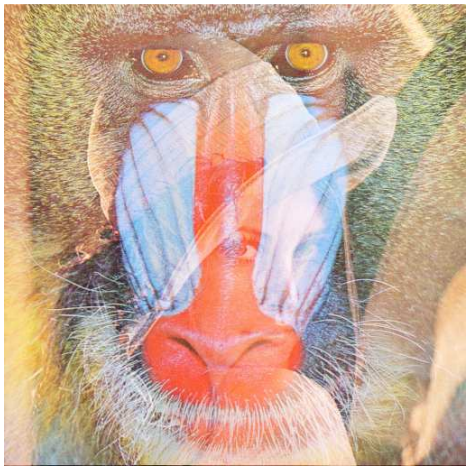
```
kimenet = uint8( ( uint16( lena ) .* uint16( mandrill ) ) / 255 );
imshow(kimenet)
```

Egyik kép rávetítése a másikra (screen)

Ez az eljárás olyan, mintha fognánk a két kép negatívját, egymásra raknánk őket, majd az eredményt egy negatív papírra kinyomtatnánk. Matematikailag ez így néz ki:

$$255 - \frac{(255 - p_1) * (255 - p_2)}{255}$$

kimenet = 255 - uint8((uint16(255 - lena) . uint16(255 - mandrill)) / 255);*
imshow(kimenet)



Természetesen ezeket az eljárásokat a mátrixok egy kisebb részletén is elvégezhetjük, nem feltétlen kell az egészet használnunk.

A képfeldolgozás során gyakoribb probléma szokott lenni, hogy valamit meg akarunk keresni, éleket kiemelni, … Ilyen esetekben sokszor konvolúciós mátrixokkal dolgozunk, s egy-egy szűrőt az ő saját konvolúciós mátrixával adhatunk meg.

Képek szűrése konvolúciós mátrixokkal

Mi az a konvolúció?

A konvolúció képlete az alábbi:

$$(f * g)(n) = \sum_{k \in D} f(k)g(n - k)$$

Bár ez rendkívül csúnyán nézhet ki első ránézésre, a gyakorlatban nagyon egyszerű használni. Ha például egy szűrőnk konvolúciós mátrixa így néz ki (ez egy egyszerű, minden irányban érzékeny élkiemelő szűrő):

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Akkor a kép minden pontjára az új értéket úgy kell kiszámolni, hogy az éppen vizsgált pont a konvolúciós mátrix középső értékének súlyával lesz figyelembe véve, míg a körülötte lévő pontok értékét hozzáadjuk a konvolúciós mátrixnak megfelelő értékben. Ha egy pont és a környezete:

$$\begin{bmatrix} 197 & 194 & 197 \\ 192 & 200 & 197 \\ 193 & 200 & 198 \end{bmatrix}$$

Akkor az erre a pontra számított kimeneti érték meghatározása:

$$p_{ki} = (-1 * 197) + (-1 * 194) + (-1 * 197) + (-1 * 192) + (9 * 200) + (-1 * 197) + (-1 * 193) + (-1 * 200) + (-1 * 198) = 232$$

Ahhoz, hogy ezt a szűrőt ráereszthessük a mátrixunkra, mindössze a ***conv2*** utasítást kell használnunk, azzal az aprócska megjegyzéssel, hogy ezt színcsatornánként kell megtennünk.

```
kimenet(:,:,1) = conv2 (lena(:,:,1), elkiemelo, "same");
kimenet(:,:,2) = conv2 (lena(:,:,2), elkiemelo, "same");
kimenet(:,:,3) = conv2 (lena(:,:,3), elkiemelo, "same");
imshow (kimenet)
```

További konvolúciós mátrixokat az interneten tömkelegével lehet találni, azonban egy dologra oda kell figyelni használatukkor. Ha egy konvolúciós mátrix elemeinek összege nagyobb, mint egy, akkor a kimeneti képünk könnyedén kifehéredik, azaz könnyedén érhetünk el túlcsordulást (lépünk ki a megengedett 0..255 tartományból). Ugyanígy oda kell figyelni a negatív értékekre is, tehát célszerű úgy megválasztani a konvolúciós mátrixunkat, hogy az elemeinek összegének értéke a [0,1] tartományba essen.

Végszó

Ebben a rövidke írásban igyekeztem bemutatni az Octave program egy kis szegletét, azonban maga a program oly sokrétűen használható, mint a matematika maga, így tehát teljes áttekintést adni róla nem állhatott szándékomban.

Forrásjegyzék

Octave hivatalos honlap: http://www.gnu.org/software/octave/
Wikipédia: http://en.wikibooks.org/wiki/Octave_Programming_Tutorial/
Werner D. Streidt – Digital Image Processing with FilterMeister, 1999