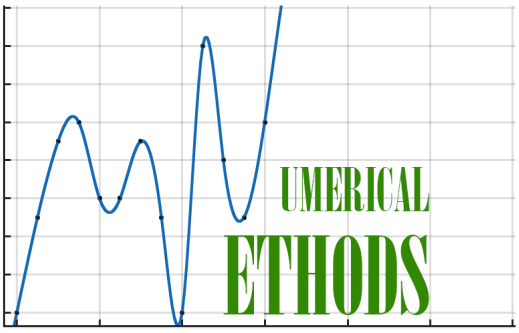


Tema de casă
Interpolare aplicată pe imagini

Coordonator: Clementin-Dumitru Cercel
Autori: Andrei-Antonio Carapcea, Bogdan Ciobanu

Aprilie 2021

1 Introducere

Interpolarea este o metodă prin care se obțin valori intermediare aproximative ale unei funcții pentru care se cunosc doar o parte din valori.
O imagine poate fi interpretată ca o funcție $f : Z^2 \to R$, cu semnificația că $f(x, y)$ reprezintă intensitatea luminoasă a pixel-ului de la poziția dată de indicii x și y.
Observație: x și y sunt întregi! Un ecran fizic nu poate avea o jumătate de pixel.
Desigur, o singură funcție este suficientă doar pentru a descrie o imagine alb negru. O imagine colorată poate fi descrisă prin 3 astfel de funcții, fiecare asociată unei anumite culori, cel mai comun sistem de culori fiind roșu, galben și albastru (RGB).
Astfel, a aplica o interpolare pe o imagine înseamnă, de fapt, a aplica o interpolare pe o funcție de doi parametrii, cu mențiunea că într-o imagine finală ne vor interesa doar valorile de la coordonate întregi.

1.1 Aplicații

În practică, interpolarea aplicată pe imagini este folosită pentru:

1.1.1 Transformări geometrice

Atunci când se aplică o transformare geometrică asupra unei imagini (fie ea de scalare, de rotație, etc.), este posibil ca un pixel din imaginea finală să nu corespundă exact unui pixel din imaginea

originală, adică poziția acestuia să nu fie un număr întreg. Interpolarea oferă o modalitate de a calcula valoarea luminoasă în acest punct în funcție de pixelii din jur.

1.1.2 Texture filtering

Texture filtering este un concept din grafica calculatoarelor și se ocupă de corespondența unei texturi (a unei imagini) pe un model 3D. Un pixel al modelului 3D nu va corespunde neapârat cu un pixel al texturii (numit texel). Lucrurile se complică ținând cont că modelul respectiv poate fi privit din orice poziție si din orice unghi. Astfel, se utilizează o interpolare pentru a afla culoarea unui pixel.

1.1.3 Image inpainting

O imagine poate fi ușor deteriorată, astfel încât părți din ea să lipsească. Image inpainting se ocupă cu "umplerea" părților lipsă din imagine pentru a aproxima imaginea corectă.

Figure 1: Imaginea originală si reparată [2].

2 Metode abordate

În cadrul temei, se vor implementa trei metode de interpolare aplicate pe imagini:

- Interpolare nearest-neighbour;
- Interpolare biliniară;
- Interpolare bicubică.

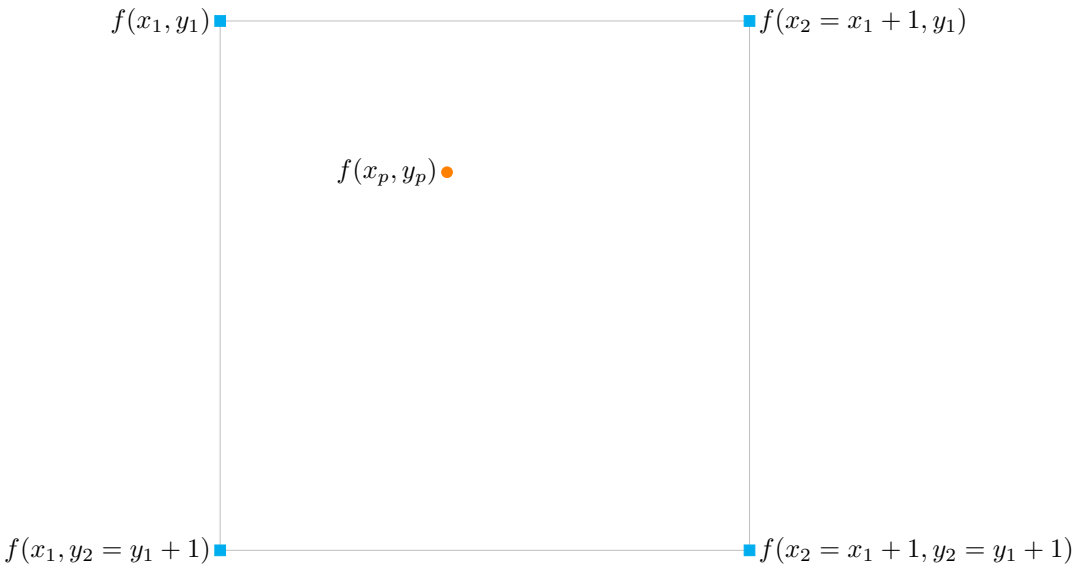
Concret, acestea vor fi folosite pentru a redimensiona și pentru a roti imagini.
Dorim să aplicăm o transformare geometrică, notată cu T, asupra unei imagini. Transformările geometrice sunt reprezentate în formă matriceală pentru că facilitează procesul de compunere (doar se înmulțesc matricile asociate, $T_{compunere} = T_1 * T_2 * ...$).

În urma acestei transformări unii pixeli nu o să mai aibă coordonate întregi, dar în imaginea finală adresarea pixelilor se face doar în astfel de coordonate întregi. Interpolarea oferă o modalitate de a calcula valorile pixelilor din imaginea finală pe baza celor din imaginea inițială.

Mai exact, se aplică următoarea procedură:

Pentru fiecare pixel (x_o, y_o) din imaginea finală, se aplica transformarea inversă T^{-1} pentru a trece în coordonatele imaginii inițiale și se obține punctul (x_p, y_p). În urma acestei transformări inverse, poziția rezultată nu o să mai fie neapărat întreagă, ci o sa aibă și o parte fracționară.

Se determină coordonatele celor patru pixeli din imaginea inițială ce înconjoară punctul (x_p, y_p):



Apoi, se va folosi una din metodele de interpolare menționate pentru a calcula $f(x_p, y_p)$ pe baza celor 4 puncte înconjurătoare, obținându-se astfel valoarea pixelului (x_o, y_o) din imaginea finală.

2.1 Redimensionare

Dorim să redimensionăm o imagine de dimensiune $m \times n$ astfel încât aceasta să devină de dimensiune $p \times q$ (cu p și q mai mari ca m, respectiv n). Aceasta se realizează printr-o transformare a sistemului de coordonate caracterizată de 2 constante de scalare:

$$s_x = \frac{q}{n} \qquad s_y = \frac{p}{m}$$

Astfel încât:

$$x_o = x_i * s_x \qquad y_o = y_i * s_y$$

O altă modalitate de a reprezenta transformarea de mai sus este:

$$\begin{bmatrix} x_o \\ y_o \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

Transformarea și inversa ei sunt:

$$T = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \qquad T^{-1} = \begin{bmatrix} \frac{1}{s_x} & 0 \\ 0 & \frac{1}{s_y} \end{bmatrix}$$

Observație pentru implementarea temei: în urma scalării, originea imaginii ar trebui să rămână nemișcată. În Octave, indexarea se face de la 1, iar originea imaginii se reprezintă prin coordonatele $(1, 1)$. Dacă se înmulțește acest vector cu transformarea de scalare, se va obține punctul (s_x, s_y), care nu mai reprezintă originea în imaginea finală. O soluție ar fi ca în operațiile cu transformate să se folosească coordonate indexate de la 0.

2.2 Rotație

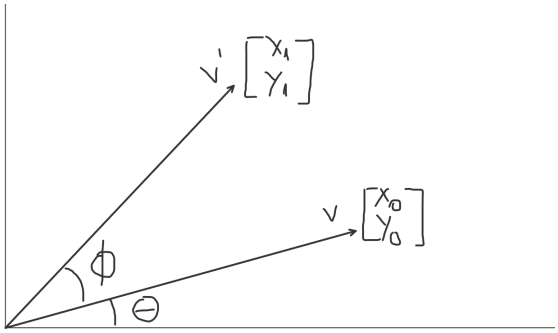


Figure 2: Rotirea unui vector 2D.

Pornind de la coordonatele polare ale varfurilor vectorilor v și v':

$$\begin{aligned} x_0 &= rcos\theta \\ y_0 &= rsin\theta \\ x_1 &= rcos(\theta + \phi) \\ y_1 &= rsin(\theta + \phi) \end{aligned}$$

Se deduc formulele pentru vectorul v':

$$\begin{aligned} x_1 &= r(cos\theta cos\phi - sin\theta sin\phi) = x_0 cos\phi - y_0 sin\phi \\ y_1 &= r(sin\theta cos\phi + cos\theta sin\phi) = y_0 cos\phi + x_0 sin\phi \end{aligned}$$

În forma matriceală:

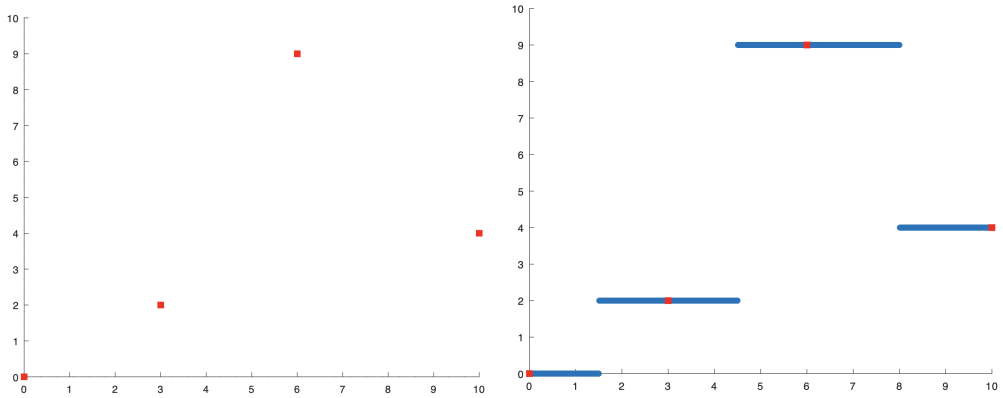
$$\begin{bmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

Așadar, putem obține locația oricărui punct din plan, după o rotire la stânga cu ϕ radiani, înmulțind coordonatele punctului, la stânga, cu matricea de rotație:

$$T_{rot} = \begin{bmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{bmatrix}$$

3 Interpolare nearest-neighbour

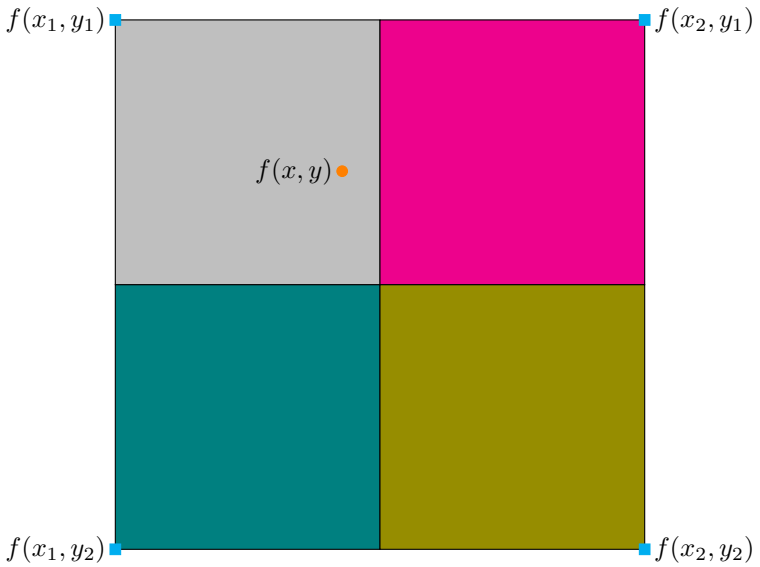
Se consideră un caz 1D, o funcție $f(x)$ cunoscută în doar câteva puncte. Metoda "celui mai apropiat vecin" este cea mai simplă metodă de interpolare și funcționează în concordanță cu



(a) Punctele cunoscute. (b) Rezultatul interpolării.

Figure 3: Exemplu interpolare 1D nearest-neighbour.

numele: în loc să calculeze valori noi intermediare, pur și simplu se caută cel mai apropiat punct cunoscut și se replică valoarea funcției din acesta.
Observație: funcția obtinuta în urma acestui proces nu este continuă. Ea este doar continuă (chiar constantă) pe porțiuni. Extinzând această metodă la cazul 2D, pentru fiecare pixel nou din imaginea rezultat trebuie găsit cel mai apropiat pixel din imaginea sursă. Se fac două interpolări 1D pe fiecare dimensiune, o dată pe axa OX pentru a determina dacă cel mai apropiat punct este unul din punctele din stânga sau din dreapta, și o dată pe axa OY pentru a determina dacă este unul din punctele de sus sau de jos. Rezultatul obținut va delimita 4 zone de replicare ale valorilor funcției.



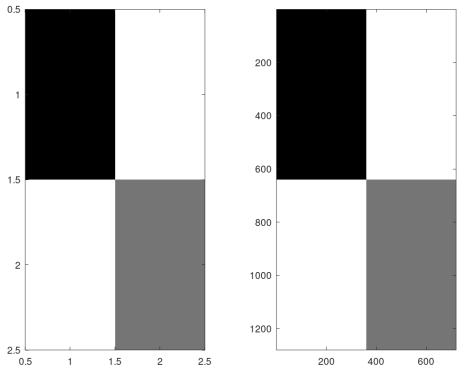


Figure 4: Rezultatul scalării folosind interpolare nearest-neighbour.

3.1 Observații rezultate

Pentru acest exemplu de mai sus cu linii drepte rezultatul final este unul bun. Însă din cauza faptului ca funcția obtinuta nu este continuă, pentru o imagine obișnuită rezultatul arată destul de "pixelat".

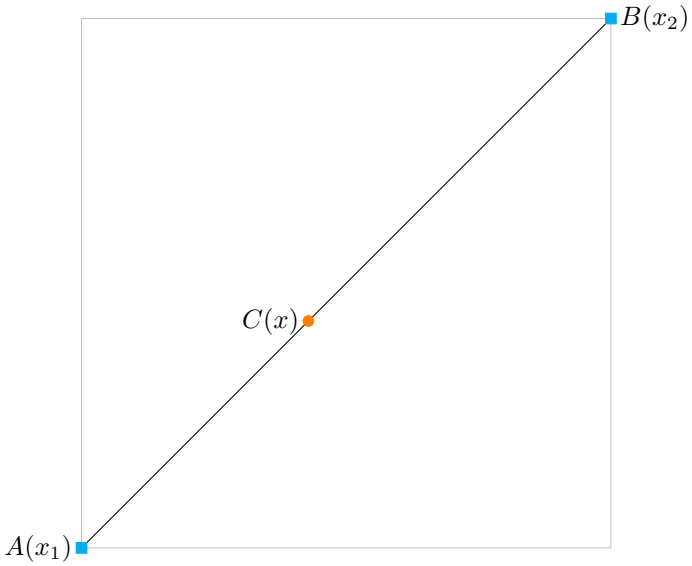
(a) Imagine originală.

(b) Rezultatul interpolării.

Figure 5: Interpolare nearest-neighbour pe o imagine generică.

4 Interpolare biliniară

Se consideră mai întâi cazul 1D de interpolare liniară între două puncte:



Se trage o linie dreaptă din A în B. Panta acestei linii va fi:

$$\frac{df}{dx} = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

Punctul C, cu x arbitrar, aflându-se pe dreaptă, va păstra aceeași pantă:

$$\frac{f(x_2) - f(x_1)}{x_2 - x_1} = \frac{f(x) - f(x_1)}{x - x_1}$$

$$f(x) = f(x_1) + \frac{(x - x_1)(f(x_2) - f(x_1))}{x_2 - x_1}$$

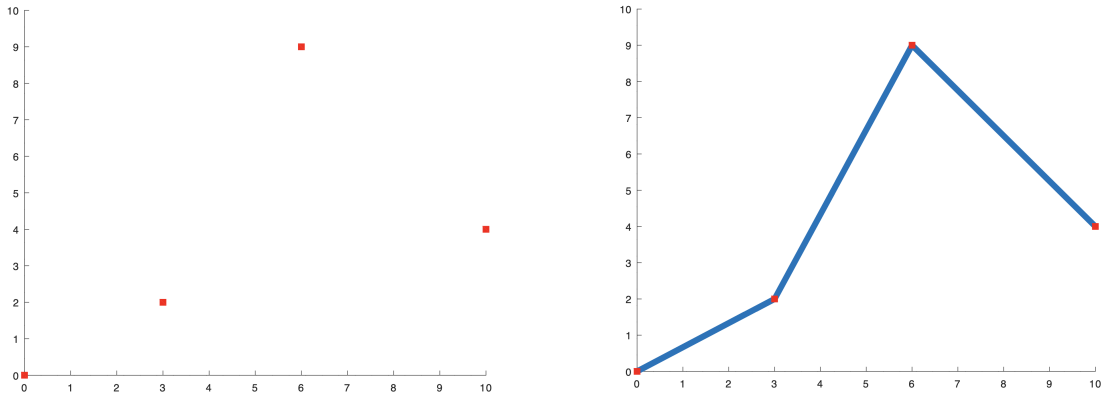
$$f(x) = \frac{(x_2 - x_1)f(x_1) + (x - x_1)f(x_2) - (x - x_1)f(x_1)}{x_2 - x_1}$$

$$f(x) = \frac{x_2 - x}{x_2 - x_1} f(x_1) + \frac{x - x_1}{x_2 - x_1} f(x_2)$$

Pentru mai multe puncte, se interpolează cu spline-uri liniare între doua puncte adiacente:
Observație: funcția obtinuta este continuă, dar ea nu are derivata continuă!
Interpolarea biliniară consta in două interpolări liniare, una pe axa OX și una pe axa OY.
Se face mai întâi o interpolare pe axa OX, fixând y drept y_1 apoi y_2.

$$f(x, y_1) = \frac{x_2 - x}{x_2 - x_1} f(A_{11}) + \frac{x - x_1}{x_2 - x_1} f(A_{21})$$

$$f(x, y_2) = \frac{x_2 - x}{x_2 - x_1} f(A_{12}) + \frac{x - x_1}{x_2 - x_1} f(A_{22})$$



(a) Punctele cunoscute

(b) Rezultatul interpolării

Figure 6: Exemplu interpolare liniară 1D.

Apoi se interpolează pe axa OY:

$$f(x,y) = \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2)$$

$$f(x,y) = \frac{y_2 - y}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(A_{11}) + \frac{x - x_1}{x_2 - x_1} f(A_{21}) \right) + \frac{y - y_1}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(A_{12}) + \frac{x - x_1}{x_2 - x_1} f(A_{22}) \right)$$

$$f(x,y) = \frac{1}{(x_2 - x_1)(y_2 - y_1)} \begin{bmatrix} x_2 - x & x - x_1 \end{bmatrix} \begin{bmatrix} f(A_{11}) & f(A_{12}) \\ f(A_{21}) & f(A_{22}) \end{bmatrix} \begin{bmatrix} y_2 - y \\ y - y_1 \end{bmatrix}$$

Observație: deși s-au aplicat două interpolări liniare, o dată pe axa OX și o dată pe axa OY, funcția rezultantă nu este liniară! Apare și un termen de forma $c \cdot xy$.
Astfel, o alternativă de rezolvare se obține dacă se consideră forma finală a funcției:

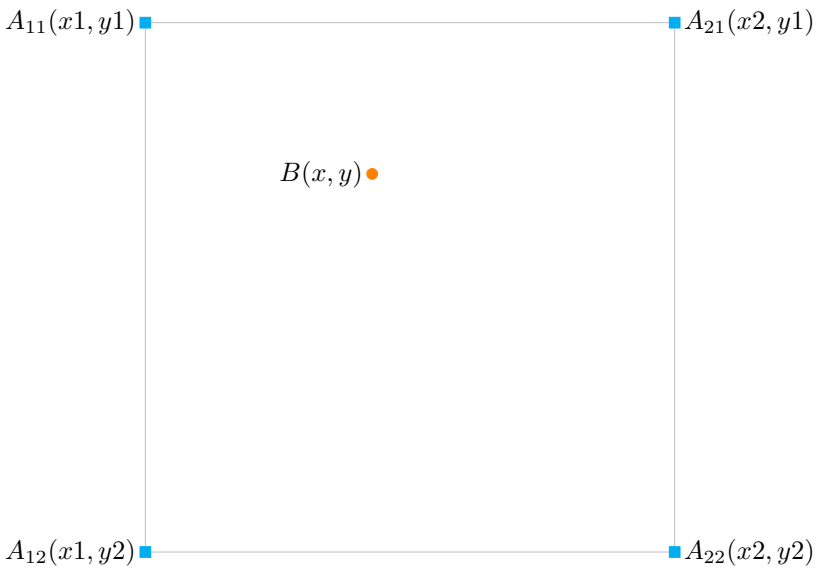
$$f(x,y) = a_0 + a_1 x + a_2 y + a_3 xy.$$

Și se impune condiția ca funcția să dea valorile cunoscute în punctele cunoscute, obținându-se un sistem liniar de ecuații care poate fi rezolvat cu metodele studiate:

$$\begin{bmatrix} 1 & x_1 & y_1 & x_1 y_1 \\ 1 & x_1 & y_2 & x_1 y_2 \\ 1 & x_2 & y_1 & x_2 y_1 \\ 1 & x_2 & y_2 & x_2 y_2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} f(A_{11}) \\ f(A_{12}) \\ f(A_{21}) \\ f(A_{22}) \end{bmatrix}$$

4.1 Observații rezultate

În urma interpolării biliniare, imaginea rezultată este mult mai "lină" decât interpolarea nearest-neighbour. Acest lucru poate fi atât un avantaj, cât și un dezavantaj, pentru că face ca toate muchiile să fie mult mai nedefinite. Se observă fenomenul mai ales pentru text:



5 Interpolare bicubică

Interpolarea bicubică reprezintă extensia interpolării cu funcții spline cubice la două dimensiuni. Se consideră o funcție $f : [0, 1] \times [0, 1]$. Se cunosc valorile funcției și ale derivatelor sale ($\frac{\partial f}{\partial x} = f_x$, $\frac{\partial f}{\partial y} = f_y$, $\frac{\partial^2 f}{\partial x \partial y} = f_{xy}$) în punctele (0,0), (0,1), (1,0) și (1,1). Rezultatul interpolării bicubice este de forma: $f(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j$, având derivatele parțiale:

$$\frac{\partial f}{\partial x} = f_x = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} i x^{i-1} y^j \tag{1}$$

$$\frac{\partial f}{\partial y} = f_y = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} j x^i y^{j-1} \tag{2}$$

$$\frac{\partial^2 f}{\partial x \partial y} = f_{xy} = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} i j x^{i-1} y^{j-1} \tag{3}$$

$$\tag{4}$$

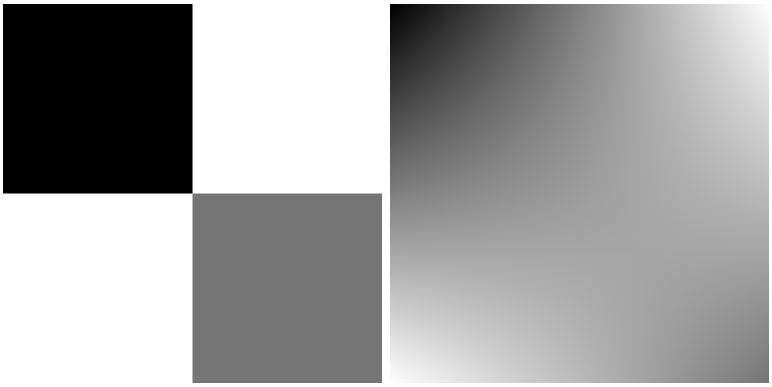
Trebuie aflați cei 16 coeficienți a_{ij}. Din condiția ca funcția să aibă valorile cunoscute apar 4 ecuații:

$$f(0, 0) = a_{00} \tag{5}$$

$$f(0, 1) = a_{00} + a_{01} + a_{02} + a_{03} \tag{6}$$

$$f(1, 0) = a_{00} + a_{10} + a_{20} + a_{30} \tag{7}$$

$$f(1, 1) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} \tag{8}$$



(a) Rezultatul nearest-neighbour (b) Rezultatul interpolării biliniare

Figure 7: Interpolare biliniară.

(a) Rezultatul nearest-neighbour (b) Rezultatul interpolării biliniare

Figure 8: Interpolare biliniară.

Asemanător, impunând valorile cunoscute ale derivatelor față de x și y apar 8 ecuații:

$$f_x(0,0) = a_{10} \tag{9}$$
$$f_x(0,1) = a_{10} + a_{11} + a_{12} + a_{13} \tag{10}$$
$$f_x(1,0) = a_{10} + 2a_{20} + 3a_{30} \tag{11}$$
$$f_x(1,1) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}i \tag{12}$$
$$f_y(0,0) = a_{01} \tag{13}$$
$$f_y(0,1) = a_{01} + a_{11} + a_{21} + a_{31} \tag{14}$$
$$f_y(1,0) = a_{01} + 2a_{02} + 3a_{03} \tag{15}$$
$$f_y(1,1) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}j \tag{16}$$

Iar pentru derivata mixtă:

$$f_{xy}(0,0) = a_{11} \tag{17}$$
$$f_{xy}(0,1) = a_{11} + 2a_{12} + 3a_{13} \tag{18}$$
$$f_{xy}(1,0) = a_{11} + 2a_{21} + 3a_{31} \tag{19}$$
$$f_{xy}(1,1) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}ij \tag{20}$$

Aceste 16 ecuații formează un sistem de ecuații liniare ce poate fi rezolvat prin metodele studiate. Coeficienții a_{ij} pot fi grupați într-o matrice 4 x 4, iar sistemul de ecuații este următorul:

$$\begin{bmatrix} f(0,0) & f(0,1) & f_y(0,0) & f_y(0,1) \\ f(1,0) & f(1,1) & f_y(1,0) & f_y(1,1) \\ f_x(0,0) & f_x(0,1) & f_{xy}(0,0) & f_{xy}(0,1) \\ f_x(1,0) & f_x(1,1) & f_{xy}(1,0) & f_{xy}(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 0 & 3 \end{bmatrix}$$

Având soluția:

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) & f_y(0,0) & f_y(0,1) \\ f(1,0) & f(1,1) & f_y(1,0) & f_y(1,1) \\ f_x(0,0) & f_x(0,1) & f_{xy}(0,0) & f_{xy}(0,1) \\ f_x(1,0) & f_x(1,1) & f_{xy}(1,0) & f_{xy}(1,1) \end{bmatrix} \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Funcția poate fi scrisă și astfel:

$$f(x,y) = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 \\ y \\ y^2 \\ y^3 \end{bmatrix}$$

5.1 Aproximarea derivatelor când acestea nu sunt cunoscute

În practică, în lucrul cu imagini, valorile derivatelor în cele 4 puncte nu sunt cunoscute. Acestea sunt aproximate folosind diferențe finite.
Se consideră 16 puncte, organizate într-un pătrat 4 x 4. Față de punctele considerate anterior, se consideră și coordonatele $x, y = -1$ și $x, y = 2$ Astfel, se obțin:

$$f_x(x,y) = \frac{f(x+1,y) - f(x-1,y)}{(x+1)-(x-1)} = \frac{f(x+1,y) - f(x-1,y)}{2}$$
$$f_y(x,y) = \frac{f(x,y+1) - f(x,y-1)}{2}$$

$$f_{xy}(x,y) = \frac{\partial f_y}{\partial x} = \frac{f_y(x+1,y) - f_y(x-1,y)}{2} = \frac{\frac{f(x+1,y+1)-f(x+1,y-1)}{2} - \frac{f(x-1,y+1)-f(x-1,y-1)}{2}}{2}$$

$$f_{xy}(x,y) = \frac{f(x-1,y-1) + f(x+1,y+1) - f(x+1,y-1) - f(x-1,y+1)}{4}$$

La marginile imaginii se poate considera că derivatele sunt 0.

5.2 Observații rezultate

(a) Interpolare biliniară

(b) Interpolare bicubică

Figure 9: Comparație interpolare bicubică și biliniară

Se observă că interpolarea bicubică pare să pastreze mai bine detaliile imaginii, în timp ce interpolarea biliniară are tendința de a pierde muchiile bine definite. Acest lucru vine, desigur, cu costul unui algoritm mai complicat și a unui timp de execuție mai încet.

6 Cerințe și punctaj

README - 0.5p

6.1 Interpolare nearest-neighbour - 3 p

Să se implementeze următoarele funcții folosind interpolarea nearest-neighbour:

- nn_2x2(f, STEP) care aplică interpolare nearest-neighbour pe o imagine alb-negru f 2x2 cu puncte intermediare echidistante, având între ele distanța dată de STEP.
- nn_2x2_RGB(f, STEP) care realizează același lucru dar pentru o imagine RGB.
- nn_resize(I, p, q) care redimensionează imaginea alb-negru I de dimensiune m x n a.î. aceasta să aibă dimensiunea p x q
- nn_resize_RGB(I, p, q) care realizează același lucru, dar pentru o imagine RGB.

6.2 Interpolare biliniară - 3p

Să se implementeze următoarele funcții folosind interpolarea biliniară:

- bilinear_coef(f, x1, y1, x2, y2) care calculează coeficienții a de interpolare biliniară între 4 puncte.
- bilinear_2x2(f, STEP) care aplică interpolare nearest-neighbour pe o imagine alb-negru f 2x2 cu puncte intermediare echidistante, având între ele distanța dată de STEP.
- bilinear_2x2_RGB(f, STEP) care realizează același lucru dar pentru o imagine RGB.
- bilinear_resize(I, p, q) care redimensionează imaginea alb-negru I de dimensiune m x n a.î. aceasta să aibă dimensiunea p x q
- bilinear_resize_RGB(I, p, q) care realizează același lucru, dar pentru o imagine RGB.
- bilinear_rotate(I, angle) care rotește o imagine alb-negru I cu unghiul dat
- bilinear_rotate_RGB(I, angle) care realizează același lucru, dar pentru o imagine RGB.

6.3 Interpolare bicubică - 3.5p

Să se implementeze următoarele funcții folosind interpolarea bicubică:

- precalc_d(I) care precalculează derivatele dx, dy, dxy în fiecare punct al imaginii folosind diferențe finite
- bicubic_coef(f, Ix, Iy, Ixy, x1, y1, x2, y2) care calculează matricea A de coeficienți de interpolare bicubică între cele 4 puncte, primind la intrare matricile cu derivate precalculate
- bicubic_resize(I, p, q) care redimensionează imaginea alb-negru I de dimensiune m x n a.î. aceasta să aibă dimensiunea p x q
- bicubic_resize_RGB(I, p, q) care realizează același lucru dar pentru o imagine RGB

Pentru temă veți avea la dispoziție un schelet de cod și un checker pentru verificare automată.

References

[1] Image rotation. https://www.sciencedirect.com/topics/computer-science/image-rotation.

[2] Wikipedia inpainting. https://en.wikipedia.org/wiki/Inpainting.

[3] Wikipedia, interpolare triliniară. https://en.wikipedia.org/wiki/Trilinear_interpolation.

[4] Williams Lance. Pyramidal parametrics (mipmaps). https://web.archive.org/web/\\20140414134825/http://staff.cs.psu.ac.th/iew/cs344-481/p1-williams.pdf.