

Concepțe și Aplicații în Vederea Artificială - Tema 2

Redimensionarea imaginilor cu păstrarea conținutului

Dăscălescu Dana

1 Introducere

Tema proiectului este redimensionarea imaginii cu păstrarea conținutului. Tehnica descrisă mai jos este o implementare a algoritmului propus de S.Avidan și A.Shamir în articolul *"Seam Carving for Content-Aware Image Resizing"*.

2 Implementarea algoritmului

Dorim să păstrăm conținutul cel mai "interesant" atunci când reducem/mărim dimensiunea unei imagini. Pentru aceasta trebuie să eliminăm/adăugăm însiruirile de pixeli cu gradient mic ce conectează extremitățile.

2.1 Micșorarea imaginii pe lățime

Operația de micșorare pe lățime a unei imagini se realizează cu ajutorul funcției *decrease_width*. Această funcție elimină pixelii de pe drumurile neregulate verticale alese în funcție de opțiunea utilizatorului:

1. aleator(Figura 1)
2. folosind metoda Greedy (Figura 2 și 3)
3. folosind programarea dinamica (Figura 4 și 5)

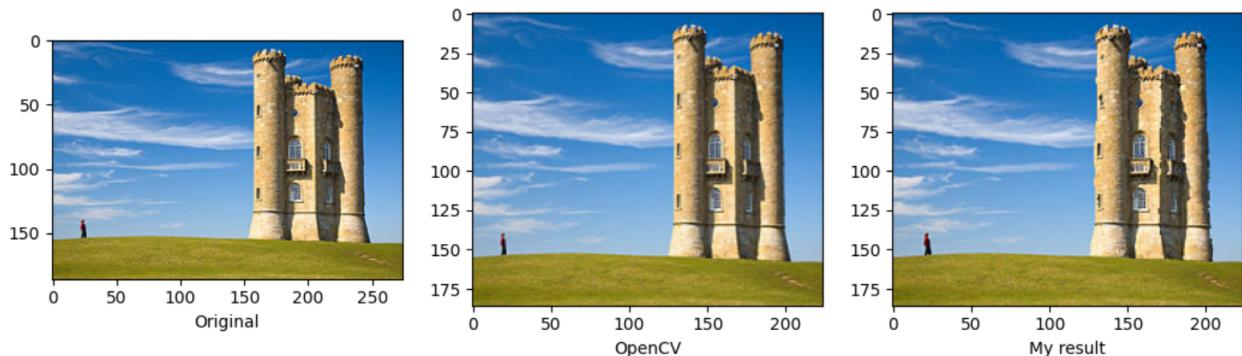


Figure 1: Imaginea originală, imaginea redimensionată folosind funcția *resize* din *OpenCV* și imaginea micșorată cu 50 de pixeli pe lățime folosind algoritmul de redimensionare ce alege drumurile în mod aleator

Se poate observa că un algoritm bazat pe metoda Greedy nu conduce la rezultatele dorite. Aceasta se datorează faptului că la fiecare pas se alege cea mai bună soluție locală (se aleg pixelii cu gradientul cel mai mic de pe linia respectivă), iar combinarea optimelor locale nu conduce la optim global.



Figure 2: Micșorarea imaginii cu 50 de pixeli pe lățime eliminând drumuri folosind metoda Greedy

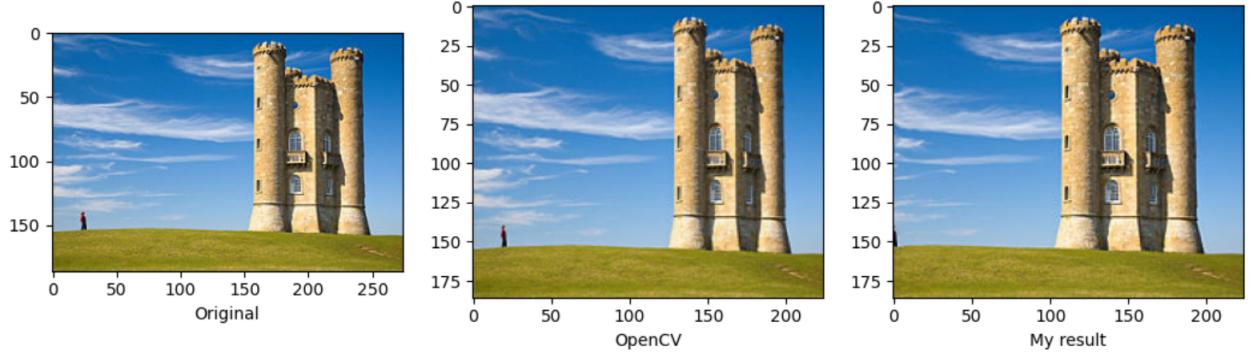


Figure 3: Imaginea originală, imaginea redimensionată folosind funcția *resize* din *OpenCV* și imaginea micșorată cu 50 de pixeli pe lățime cu algoritmul ce alege drumurile folosind metoda Greedy

Soluția ce folosește programarea dinamică elimină la fiecare pas un drum (o înlăturarea de pixeli) fără a afecta vizibil imaginea. Definim ‘costul unui drum’ ca fiind suma energiilor pixelilor ce alcătuiesc drumul. Drumul optim minimizează acest cost.

O funcție de energie definește asemănarea unui pixel față de pixelii vecini. Cu cât pixelul este mai asemănător cu cel al pixelilor vecini, cu atât este mai mică valoarea energetică.

Funcția *compute_energy* calculează energia pentru fiecare pixel pe baza magnitudinilor gradientilor folosind ecuația (1) din articol [1]:

$$e_1(I) = \left| \frac{\partial}{\partial x} I \right| + \left| \frac{\partial}{\partial y} I \right|. \quad (1)$$

unde I este imaginea inițială transformată în grayscale.

Funcția `select_dynamic_programming_path` calculează drumul de cost minim. Aceasta creează o matrice ce stochează pentru fiecare pixel (i,j) valoarea drumului de cost minim ce trece prin (i,j) folosind formula recursivă:

$$M(i, j) = E(i, j) + \min(M(i - 1, j - 1), M(i - 1, j), M(i - 1, j + 1)) \quad (2)$$

Valoarea minimă din ultima linie din M reprezintă valoarea drumului de cost minim, iar poziția acesteia localizează ultimul pixel din drum.

Drumul de cost minim se găsește parcurgând matricea M de la ultima linie către prima cu ajutorul celor trei vecini.



Figure 4: Micșorarea imaginii cu 50 de pixeli pe lățime folosind metoda programării dinamice

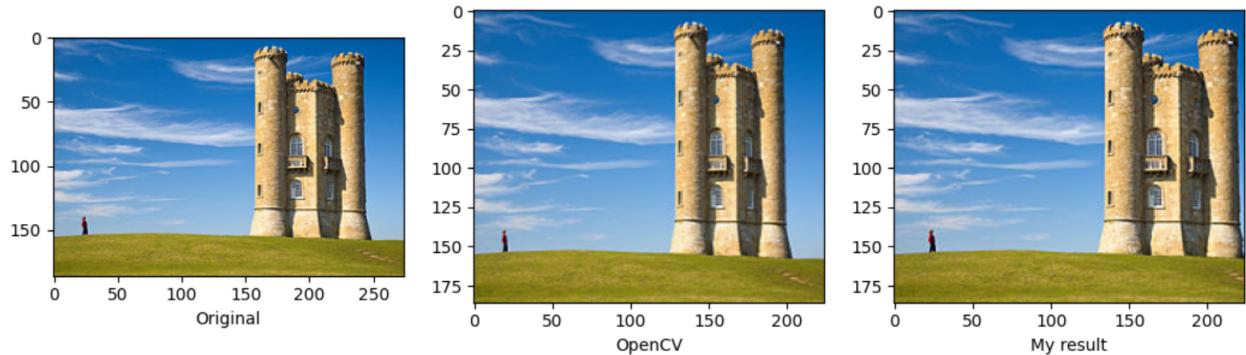


Figure 5: Imaginea originală, imaginea redimensionată folosind funcția `resize` din `OpenCV` și imaginea micșorată cu 50 de pixeli pe lățime folosind metoda programării dinamice

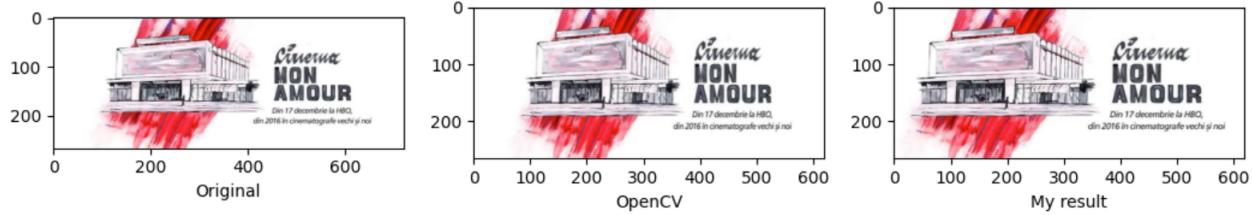


Figure 6: Imaginea originală, imaginea redimensionată folosind funcția *resize* din *OpenCV* și imaginea micșorată cu 200 de pixeli pe lățime folosind metoda programării dinamice

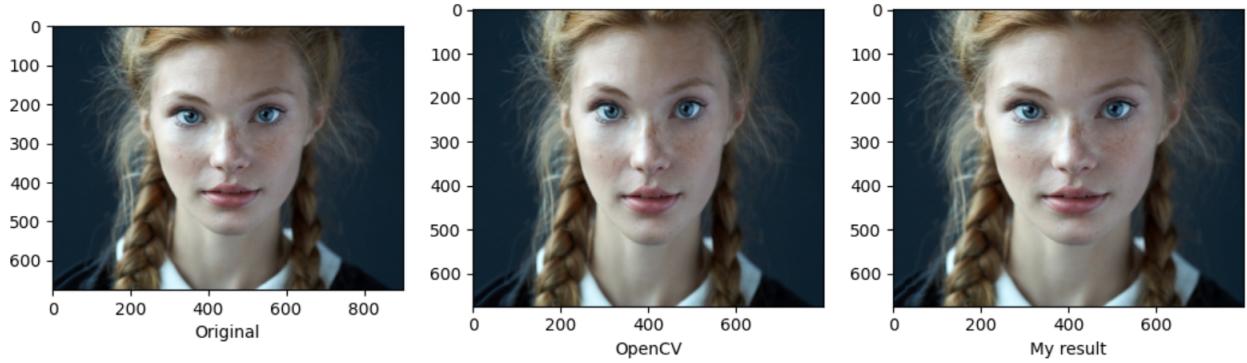


Figure 7: Imaginea originală, imaginea redimensionată folosind funcția *resize* din *OpenCV* și imaginea micșorată cu 100 de pixeli pe lățime folosind metoda programării dinamice

2.2 Micșorarea imaginii pe lungime

Operația de micșorare pe înălțime se realizează cu ajutorul funcției *decrease_height*. Funcția elimină pixelii de pe drumurile orizontale ce conectează prima coloană cu ultima coloană printr-un procedeu similar funcției *decrease_width*. Această funcție rotește imaginea cu 90 de grade în sensul acelor ceasornicului, astfel, problema eliminării însăruirilor orizontale de pixeli se transformă în cea a eliminării însăruirilor verticale.

2.3 Amplificarea conținutului imaginilor

Algoritmii de micșorare a imaginii pe lățime, respectiv pe lungime prezențați anterior pot fi utilizați și pentru amplificarea conținutului imaginilor. Pentru a păstra conținutul imaginii cât mai mult posibil, se scalează imaginea cu factorul de amplificare setat de utilizator, iar apoi aplicăm funcțiile ce realizează operațiile de micșorare pe înălțime și lățime pentru a aduce imaginea la dimensiunea originală, păstrând raportul dintre lățimea și înălțimea imaginii initiale (figura 10).

2.4 Eliminarea unui obiect din imagine

La fiecare iteratie, algoritmul elibera insiruirea de pixeli cu costul minim (suma magnitudinilor gradientilor pixelilor ce alcătuiesc drumul este minimă). Pentru a forța algoritmul să aleagă drumuri cu pixeli din regiunea selectată, vom modifica matricea de energie (figura 11), adăugând pe portiunea selectată valori negative pentru a anula efectul portiunilor cu gradient mare din portiunea respectivă.

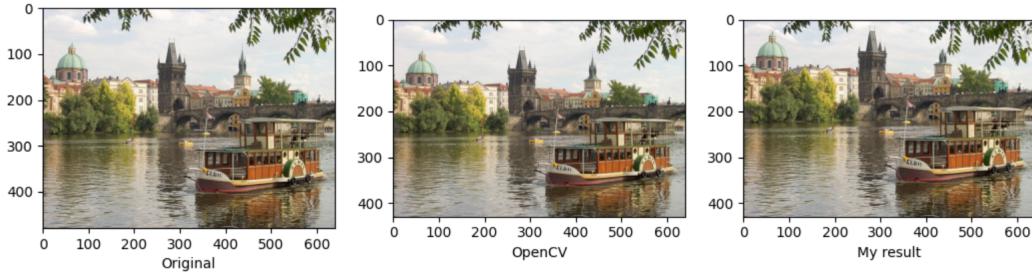


Figure 8: Imaginea originală, imaginea redimensionată folosind funcția *resize* din *OpenCV* și imaginea micșorată cu 50 de pixeli pe înălțime folosind metoda programării dinamice

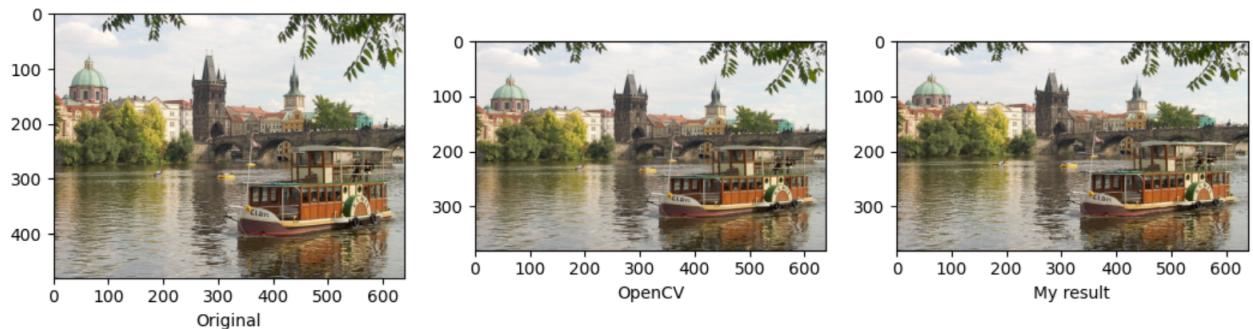


Figure 9: Imaginea originală, imaginea redimensionată folosind funcția *resize* din *OpenCV* și imaginea micșorată cu 100 de pixeli pe înălțime folosind metoda programării dinamice

3 Concluzii

Algoritmul prezentat anterior are rezultate bune pe o gamă largă de imagini (figurile 14, 6,) în comparație cu tehnici obișnuite de redimensionare. Cu toate aceste există câteva limitări.

Una dintre aceste limitări o reprezintă conținutul dintr-o imagine. Dacă imaginea este foarte condensată, în sensul că nu conține porțiuni cu gradienți puternici, atunci algoritmul va fi nevoit să eliminate zone cu conținut important (Figura 16).

O altă limitare este aspectul conținutului imaginii. Anumite imagini, deși nu sunt condensate, conținutul este așezat într-un mod care împiedică alegerea unor drumuri ce ocolește regiunile importante. De exemplu, avem o zonă cu "coținut interesant" și cu o magnitudine a gradientului mare, ce este înconjurată de o zonă mult mai mare cu o magnitudine a gradientului mică. În acest caz, zona cu "coținut interesant" contribuie cu puțin la costul drumurilor ce trece prin acea porțiune și este posibil ca unul dintre aceste drumuri să aibă cost minim și prin urmare să fie eliminat. Se poate observa că acest lucru s-a întâmplat în figura 9, unde au fost eliminate drumuri ce trec prin acoperișuri.

Un alt exemplu poate fi vizualizat în figura 19. În această imagine, "conținutul interesant" (cei doi căței) este înconjurat de o porțiune cu o magnitudine a gradienților mare (iarba). O parte din drumurile de cost minim trec prin zona cu "conținutul interesant".

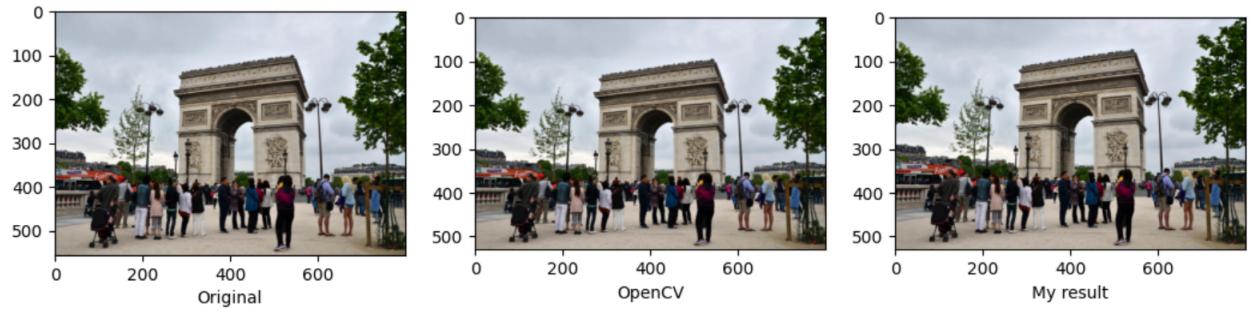


Figure 10: Imaginea originală, imaginea redimensionată folosind funcția *resize* din *OpenCV* și imaginea micșorată cu 200 de pixeli pe lățime folosind metoda programării dinamice

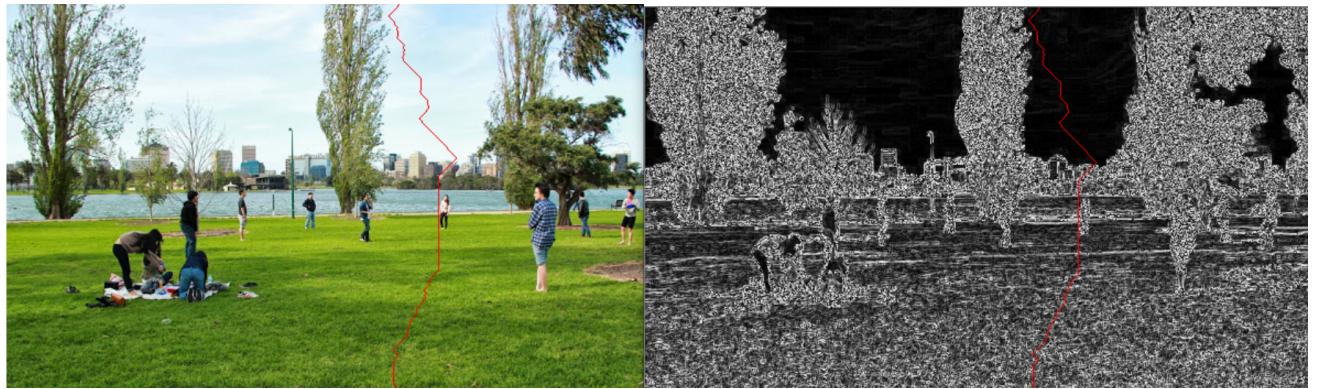


Figure 11: Eliminarea unui obiect din imaginea lac.jpg

References

- [1] Shai Avidan and Ariel Shamir. “Seam carving for content-aware image resizing”. In: *ACM Trans. Graph.* 26 (July 2007), p. 10. DOI: 10.1145/1276377.1276390.



Figure 12: Rezultatul eliminării unui obiect din imaginea lac.jpg

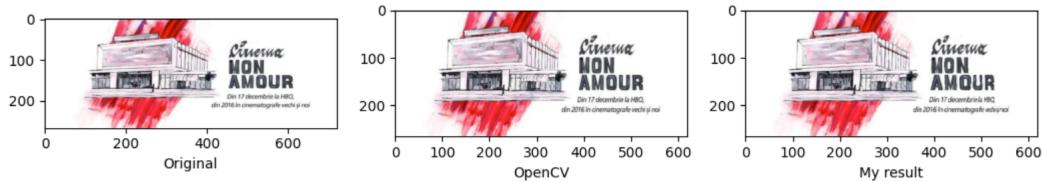


Figure 13: Imaginea originală, imaginea redimensionată folosind funcția *resize* din *OpenCV* și imaginea micșorată cu 200 de pixeli pe lățime eliminând drumuri în mod aleator.

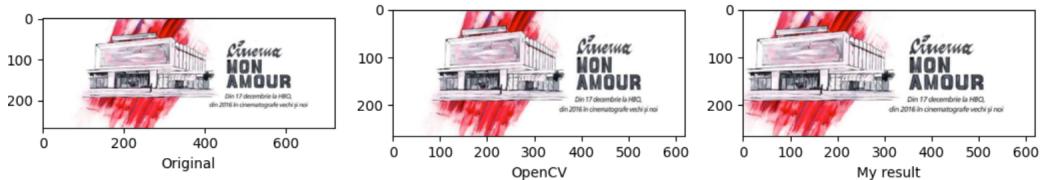


Figure 14: Imaginea originală, imaginea redimensionată folosind funcția *resize* din *OpenCV* și imaginea micșorată cu 200 de pixeli pe lățime folosind metoda greedy

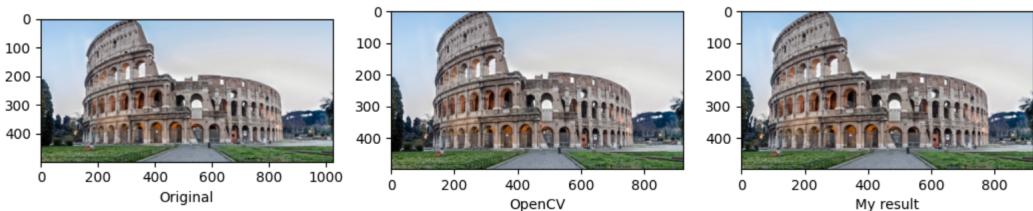


Figure 15: Imaginea originală, imaginea redimensionată folosind funcția *resize* din *OpenCV* și imaginea micșorată cu 100 de pixeli pe lățime folosind metoda programării dinamice

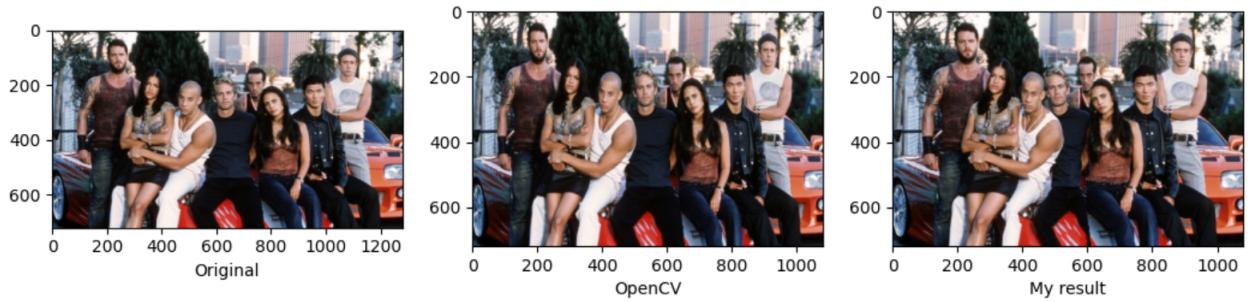


Figure 16: Imaginea originală, imaginea redimensionată folosind funcția *resize* din *OpenCV* și imaginea micșorată cu 200 de pixeli pe lățime folosind metoda programării dinamice

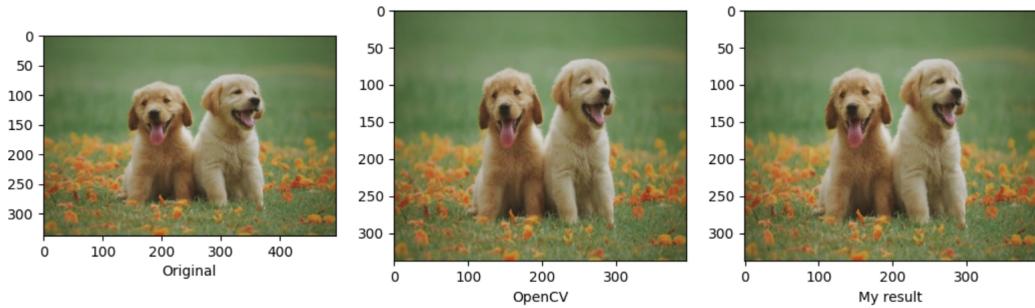


Figure 17: Imaginea originală, imaginea redimensionată folosind funcția *resize* din *OpenCV* și imaginea micșorată cu 100 de pixeli pe lățime eliminând drumuri alese în mod aleator

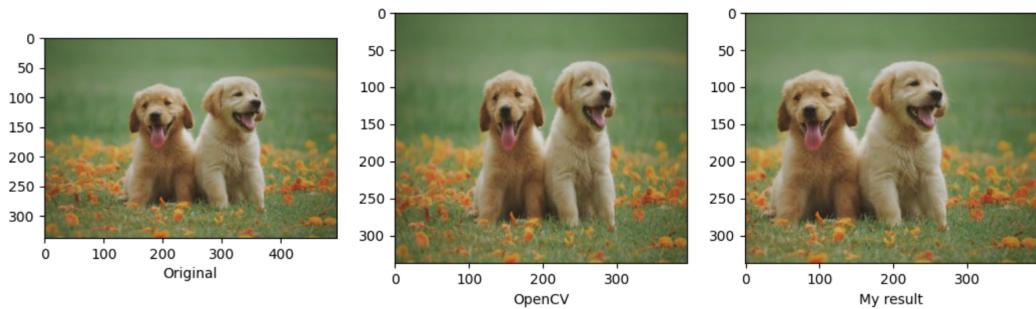


Figure 18: Imaginea originală, imaginea redimensionată folosind funcția *resize* din *OpenCV* și imaginea micșorată cu 100 de pixeli pe lățime folosind metoda greedy

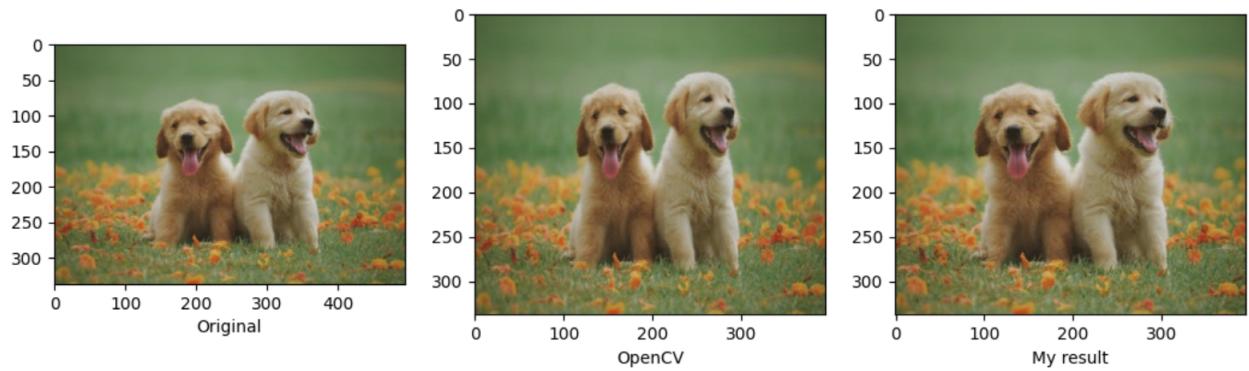


Figure 19: Imaginea originală, imaginea redimensionată folosind funcția *resize* din *OpenCV* și imaginea micșorată cu 200 de pixeli pe lățime folosind metoda programării dinamice