

## Inteligentă Artificială - Proiect 1

### Realizarea imaginilor mosaic

- (a) Mozaicurile obținute pentru imaginile din directorul '.../data/imaginiTest/' la rularea algoritmul vostru pentru diferite valori ale parametrului numarPieseMozaicOrizontala (100, 75, 50, 25) și criteriul distanței euclidiene dintre culorile medii comentând influența lor asupra rezultatelor obținute.



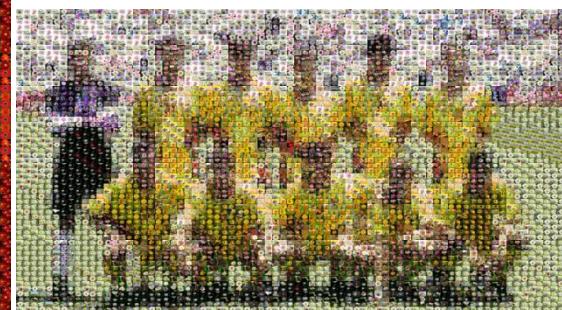
Adams – 100 piese orizontală



Ferrari – 25



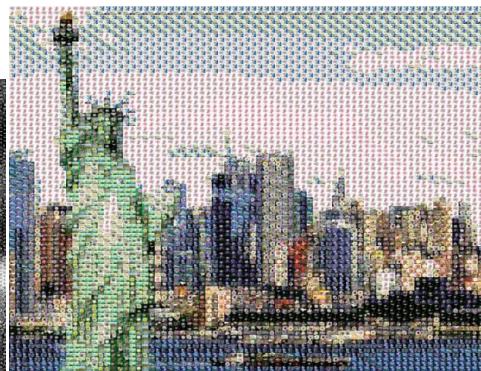
Tom și Jerry – 50



România – 75



Obama – 100



Liberty – 75

În ceea ce privește aspectul , atunci când punem mai multe piese pe orizontală crește rezoluția imaginii , aceasta devenind mai clară.

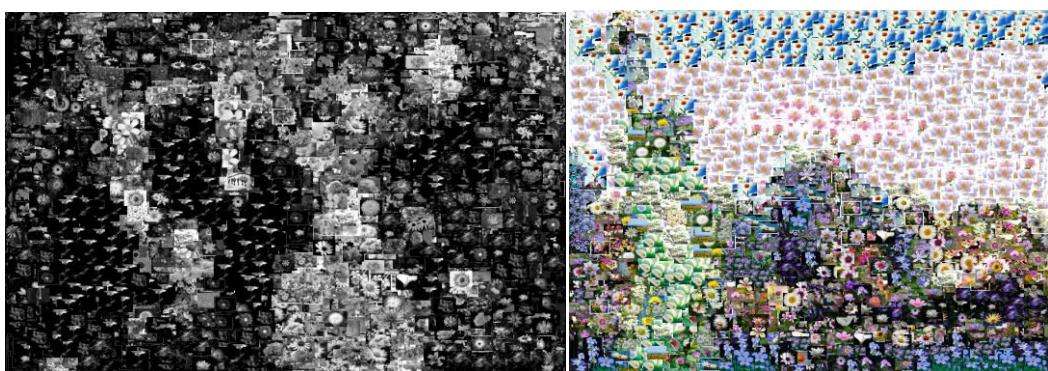
Întrucât calculăm distanța euclidiană medie dintre piese și patch-ul selecat din imaginea original , alegem piesa cea mai apropiată ca și culoare din toata colecția de patch-ul respectiv, rezultatul fiind acela că imaginea finală se aseamănă foarte mult cu imaginea originală.

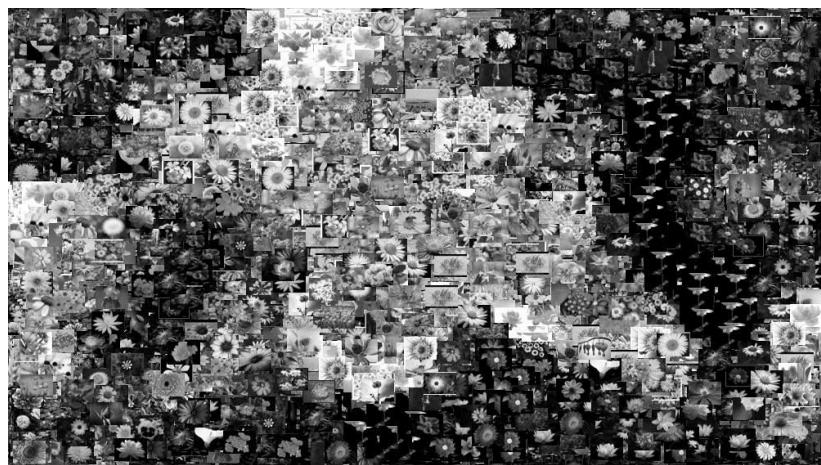
- (b) Mozaicurile obținute pentru imaginile din directorul '.../data/imaginiTest/' pentru modul de aranjare 'aleator' și criteriul distanței euclidiene dintre culorile medii. Pentru rezolvarea acestui punct trebuie să scrieți funcția `adaugaPieseMozaicModAleator.m`.



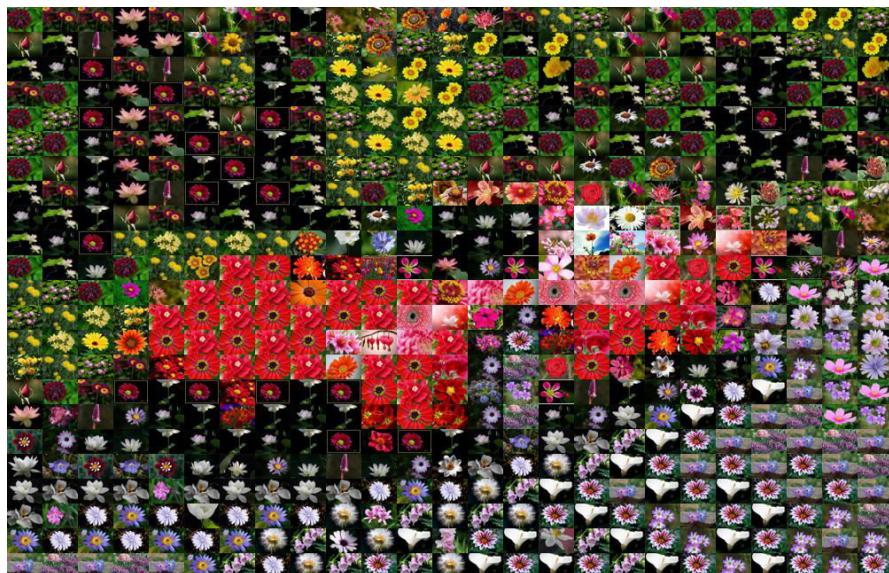
Aleator cu piese hexagonale

În funcția `adaugaPieseMozaicModAleator.m` am ales să folosesc o matrice de dimensiuni  $(h-H) \times (w-W)$  plină de zero-uri. Utilizând funcția `randi(n)` am ales un  $y$  random mai mic sau egal decât  $h-H$  și un  $x$  random mai mic sau egal decât  $w-W$ , iar dacă la poziția  $(y,x)$  în matrice am valoarea 0 , înseamnă că trebuie să pun un tile acolo (unde  $y$  și  $x$  semnifică poziția colțului din stânga sus al tile-ului) . După ce pun tile-ul , valoarea din  $(y,x)$  devine 1.

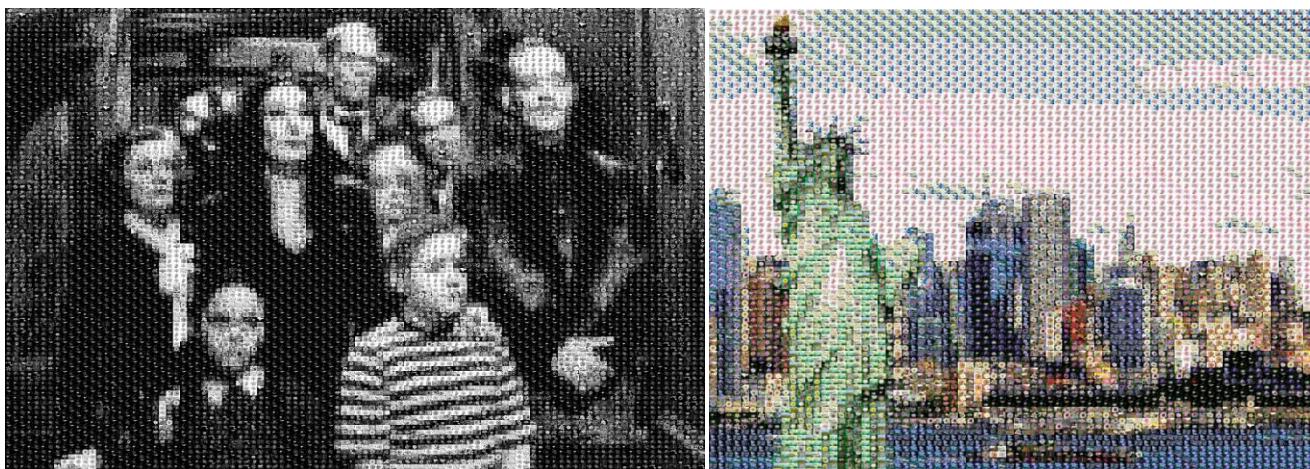




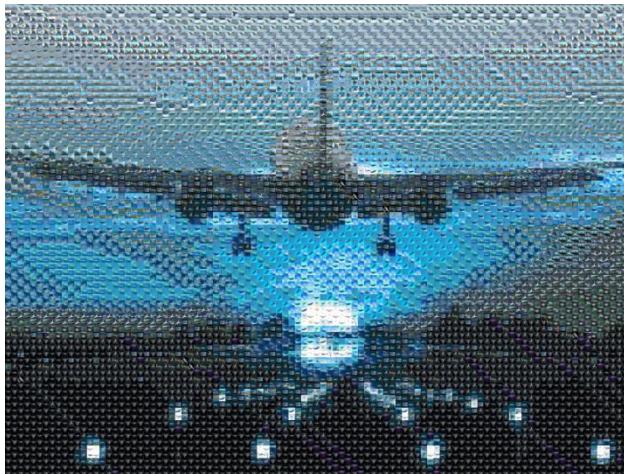
- (c) Mozaicurile obținute pentru imaginile din directorul '.../data/imaginiTest/' astfel: implementați o modificare a funcției adaugaPieseMozaicPeCaroaj.m astfel încât mozaicul obținut să aibă proprietatea că nu există două piese adiacente (stânga, dreapta, jos, sus) identice.



Modificarea pe care am făcut-o a fost să compare fiecare tile cu cel din stânga sa și de deasupra. În cazul în care sunt identice , caut următorul tile care se potrivește , cu ajutorul vectorului de distanțe.



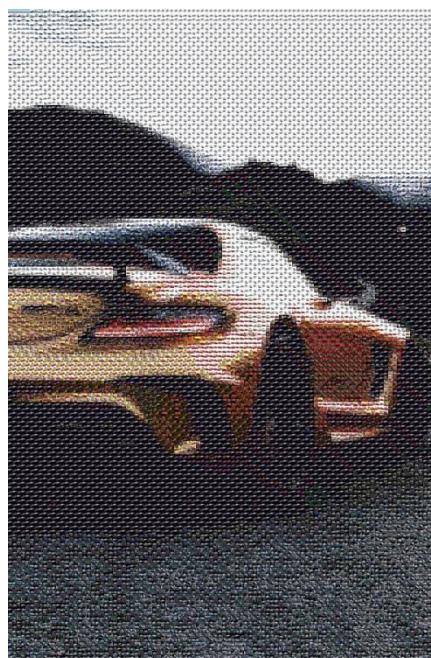
(d) Înlocuiți colecția furnizată de noi (cu cele 500 de piese ce reprezintă flori) cu o colecție proprie. O posibilitate ar fi să downloadați setul de date CIFAR – 10 de la adresa <https://www.cs.toronto.edu/~kriz/cifar.html>. CIFAR - 10 conține 60000 de imagini color de dimensiuni 32 x 32 pixeli cu obiecte din 10 clase: avion, automobil, etc. Realizați mozaicuri tematice, construind mozaicuri pentru imagini conținând obiecte din aceste clase cu piesele corespunzătoare: realizați un mosaic pentru o imagine cu un automobil folosind piese cu automobile. Includeți în pdf-ul vostru cel puțin 5 exemple de mozaicuri tematice.



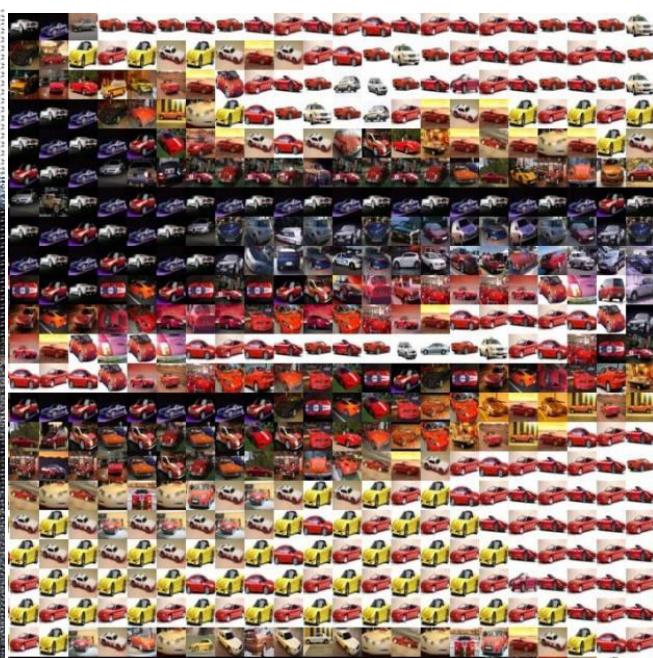
Avion



Zoom



Automobil



Zoom



Pasăre

Zoom



Câine

Zoom



Corabie

Zoom

## BONUS

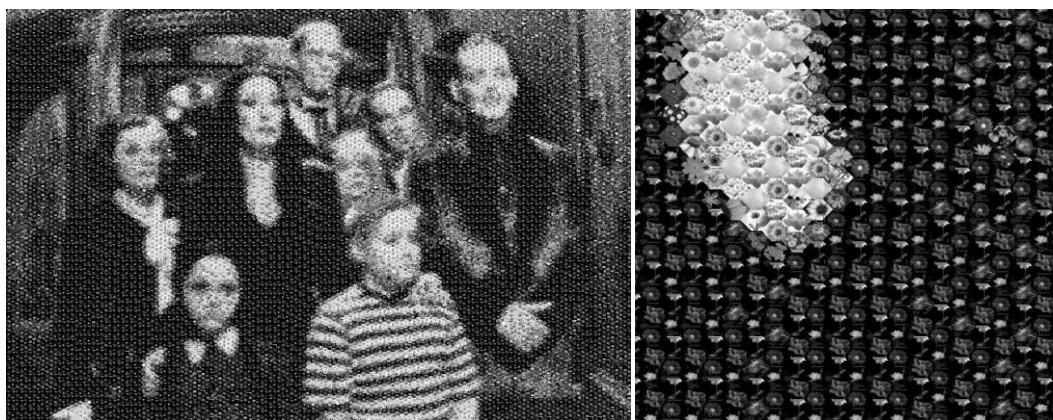
- (a) Modificați proiectul vostru astfel încât mozaicurile obținute să aibă piese hexagonale. Includeți în pdf-ul vostru mozaicurile obținute astfel pentru imaginile din directorul '.../data/imaginiTest/'.

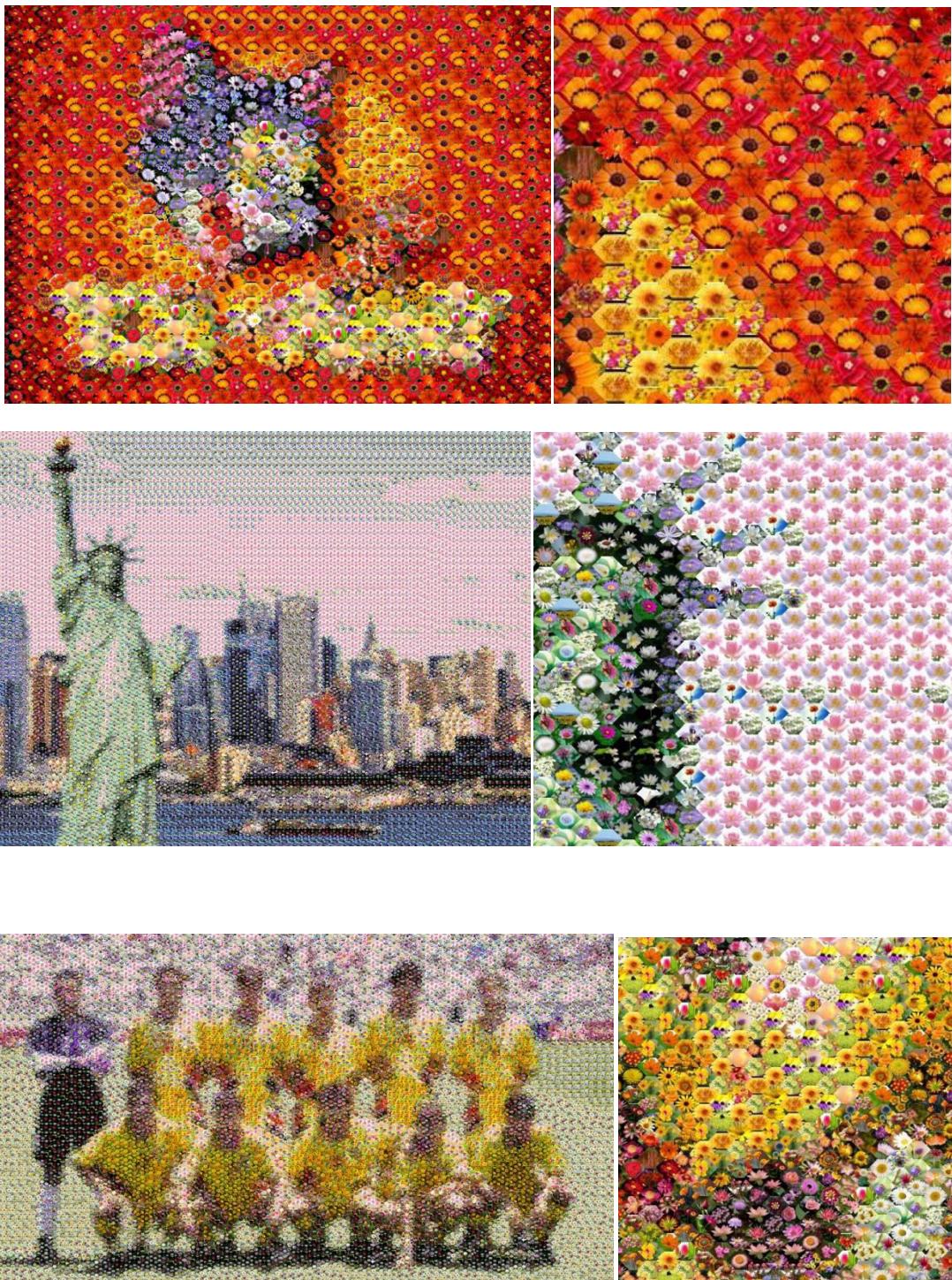


Am scris funcția `adaugaPieseHexagonale.m`, pentru distanță medie pe caroaj și pentru distanță medie aleatoriu. Am folosit 2 vectori în care am ținut minte coordonatele hexagonului din interiorul tile-ului de dimensiuni  $28 \times 40$ , apoi am folosit funcția `poly2mask` pentru a crea o mască, pe care mai apoi am aplicat-o pentru fiecare tile astfel încât rezultatul final să fie un hexagon. Pentru a da impresia de grid de tip honeycomb, tile-urile dreptunghulare au trebuit să fie suprapuse (aici am folosit variabilele `overlapH`, pentru overlap-ul pe înălțime și `overlapW` pentru cel pe lățime).

Programul cu piese hexagonale este scris strict pentru piese de dimensiuni  $28 \times 40$ . Dacă piesele sunt de alte dimensiuni, unele parametrii din interiorul codului trebuie să schimbeți. (precum `overlapH`, `overlapW`, vectorii `VectX` și `VectY`)

- (b) Modificați punctul anterior astfel încât mozaicurile obținute să aibă proprietatea că nu există două piese hexagonale adiacente (sus, jos, stânga sus, stânga jos, dreapta sus, dreapta jos) identice. Includeți în pdf-ul vostru mozaicurile obținute astfel pentru imaginile din directorul '`../data/imaginiTest/`'.





Pentru acest punct am folosit o matrice , vecini , în care am ținut minte indexul din params.pieseMozaic unde se află piesa curentă. Apoi am comparat acel index cu cel al vecinilor , iar dacă măcar unul este identic , continuă să caute alt tile , de un index diferit față de al vecinilor , folosind vectorul de distanțe.

Singura problemă cu această abordare este că , daca în colecția de piese există cel puțin 2 piese identice și cu indicii diferenți atunci există o sansă ca un hexagon să aibă , totuși , un vecin identic. Ideal , însă, ar fi să nu se repete tile-urile din colecție.