Raport z projektu z Algorytmów Ewolucyjnych

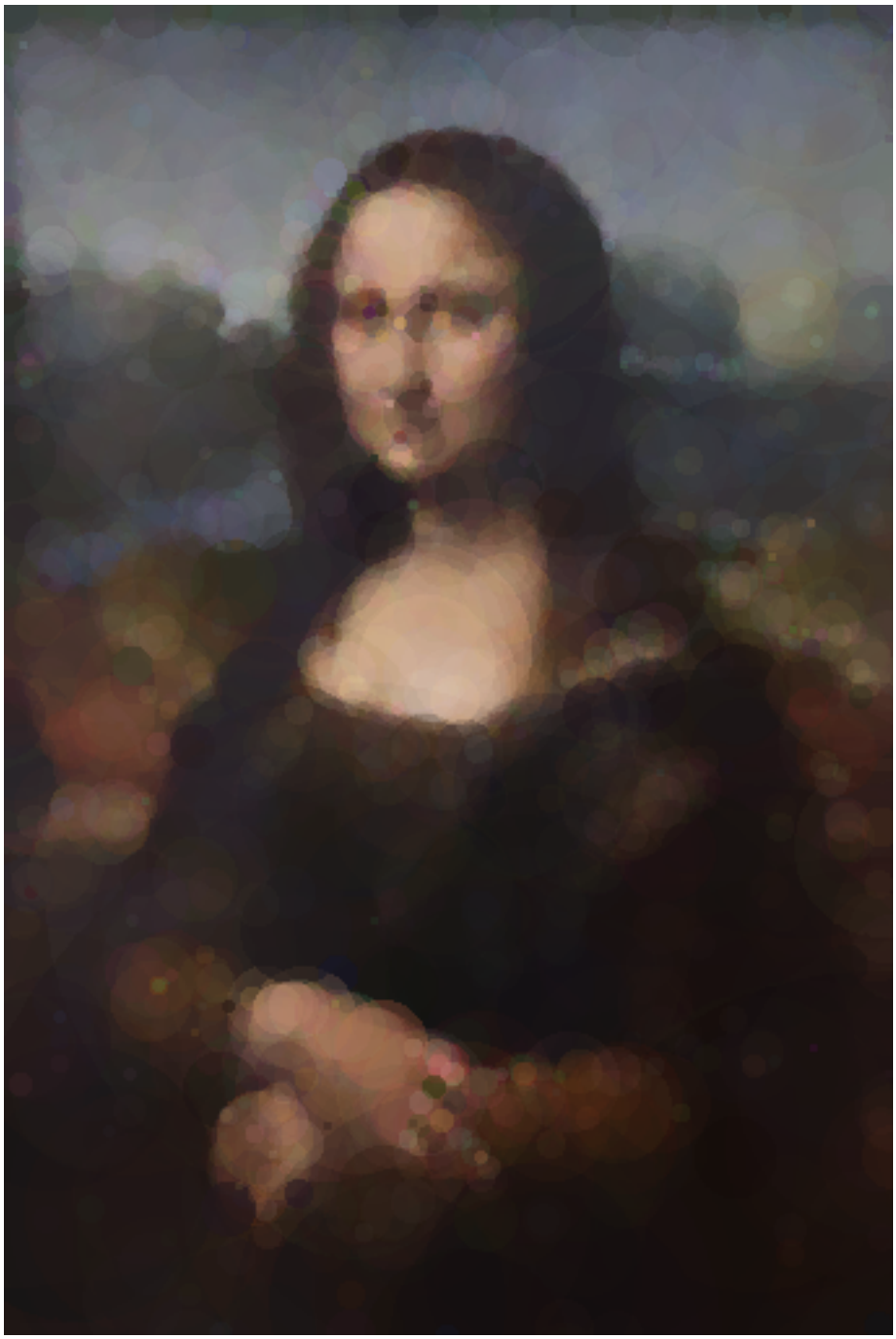
Łukasz Klasinski

11 lutego 2021

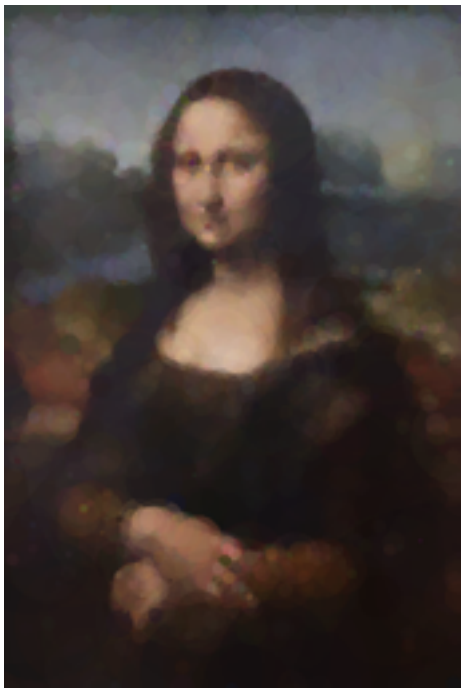
Opis projektu: Wykorzystanie algorytmu ewolucyjnego do rekonstrukcji zdjęć, gdzie głównym wzorcem będzie obraz `Mona Liza`. Sposobem rekonstrukcji będzie wyznaczanie kolejnych kropek, które pomalowane w odpowiedniej kolejności zrekonstruują obraz.

Rezultaty końcowe Obraz wzorcowy:

Najlepsze uzyskane przybliżenie po uzyciu 2000 elementów:



Pomniejszone dwukrotnie:



Rozwiazanie W celu uzyskania przybliżenia, został użyty nieco zmodyfikowany algorytm ES. Modyfikacja polegała na dodaniu ograniczeń na domenę wartości jakie mogą przyjmować poszczególne geny genotypu. Było to konieczne ze względu na to, że naturalnie genotyp kropki ma ograniczone wartości. Dodatkowo umożliwia to ustawianie tymczasowych ograniczeń (np. na wielkość kółek), dzięki czemu można uzyskać nieco lepsze odtworzenie. Sam genotyp ma następującą, (raczej nie wymagającą wjaśnienia) strukturę:

$G = \{pos_x, pos_y, radius, color_r, color_b, color_g, color_a\}$

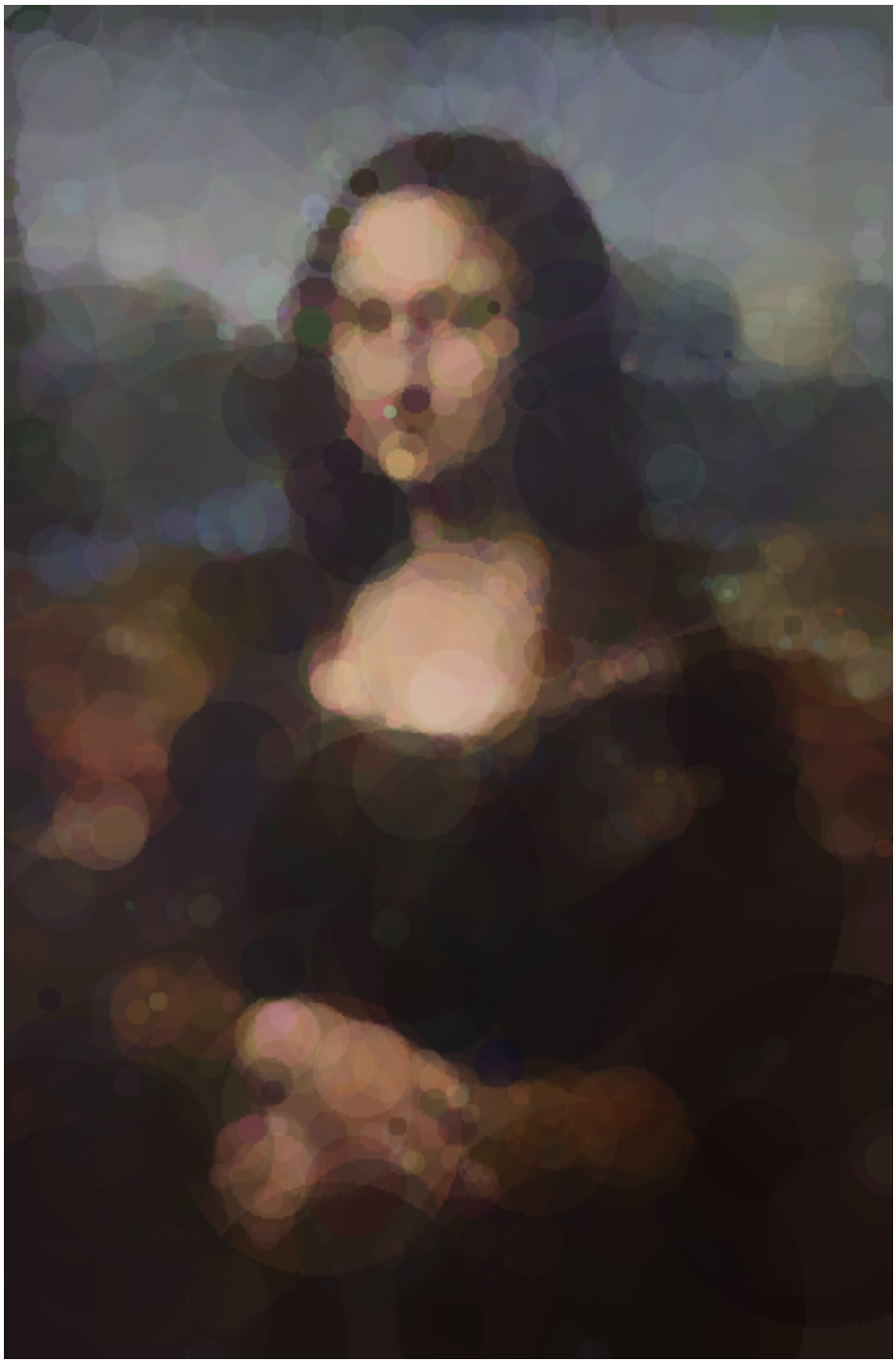
Domeny poszczególnych genów:

$$image_x = [0, len(image_x)]$$
$$image_y = [0, len(image_y)]$$
$$radius = [0, \frac{len(image_y)}{2}]$$
$$color = [0, 255]$$

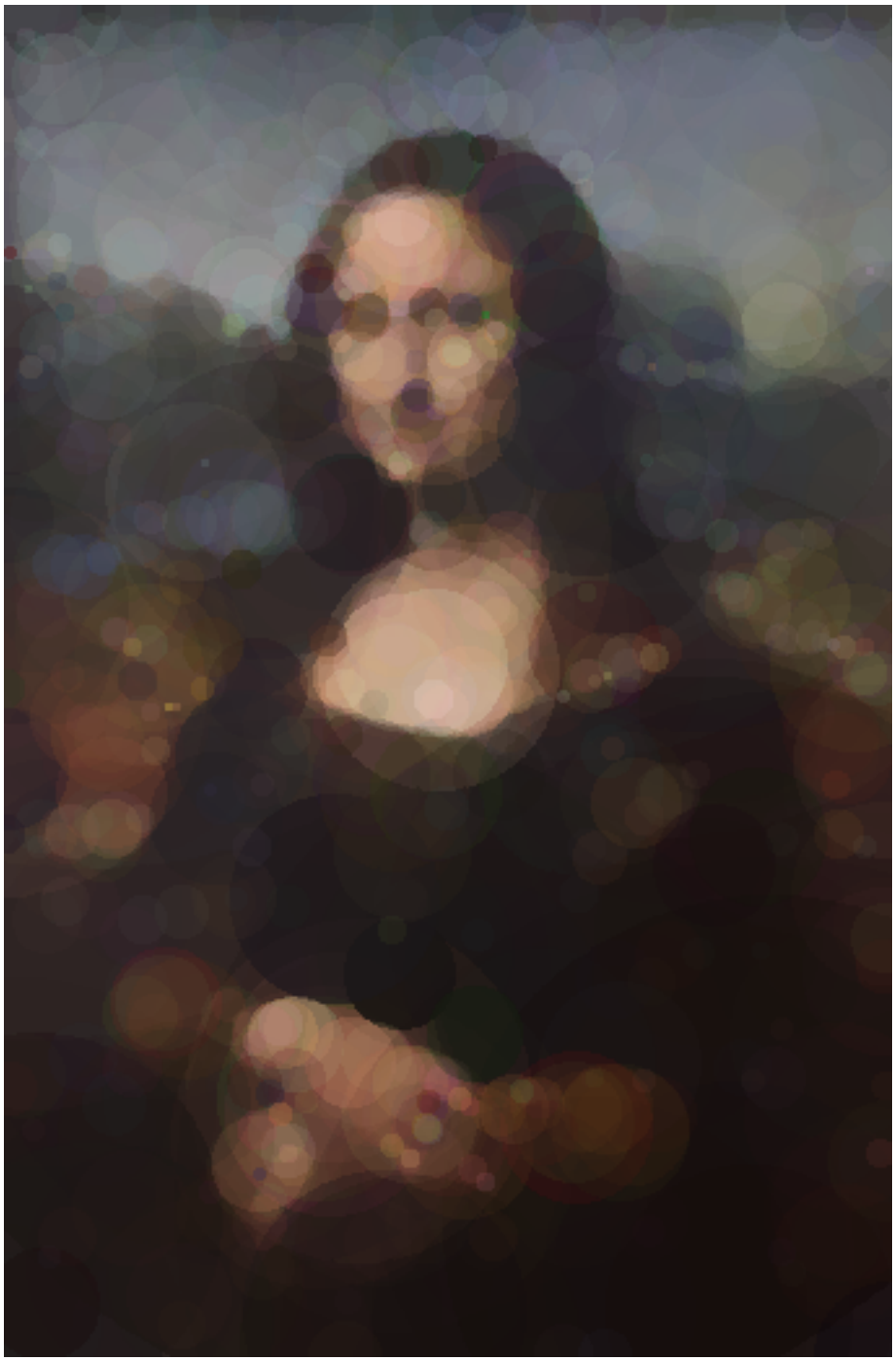
Jak widać każde kółko ma dodatkowy kanał alpha. Bez niego wyniki wymagały znacznie większej ilości kropek, szczególnie tych mniejszych, by lepiej oddać cieniowanie. Niestety przekłada się to na dłuższy czas ewaluacji populacji, bo nie wystarczy korzystać z czystego numpy'a, tylko potrzebne było zastosowanie biblioteki PIL do blendowania kolorów alpha.

Główny algorytm iteracyjnie wywołuje ES'a i z każdą iteracją dodaje kolejne kółko do rozwiązania o ile poprawi funkcję kosztu. Za funkcję kosztu została tutaj wybrana suma kwadratów różnicy między pikselami obrazu wzorcowego oraz rozwiązania. Jest ona o tyle przyjemna, że ma tylko jedno globalne minimum, do którego zawsze zbiegamy. Oczywiście najbardziej trywialnie byłoby robić kółka wielkości piksela i w taki sposób je wszystkie wypełnić dostając obraz wzrocowy. Oczywiście taka sytuacja jest niepożdana. Na szczęście w przypadku tej funkcji kosztów podejście `greedy` naturalnie stara się szukać jak największych kółek. Nie zawsze jednak dana iteracja (szczególnie na początku) zbiegnie do możliwie najbardziej dużego w danej iteracji kółka tylko czegoś pośredniego i nieoptymalnego. Dlatego zostało dodane wymuszenie na algorytmie minimalnej wielkości dla pierwszych kilkudziesięciu kółek, co poprawiło nieco wyniki(wizualnie). Przykładowo:

obraz bez restrykcji, 1000 kółek



obraz z restrykcjami, 1000 kółek



Poszczególne instancje ES uruchamiane są z następującymi hyperparametrami:

$\sigma = 20$ $N = 40$ $T = 500$

Sigma jest dosyć duża, jako że domena parametrów jest spora. Poza tym ze względu na specyfikę funkcji kosztu, bardziej opłaca się losować mniejszą ilość osobników (bo zawsze jest gdzieś sposobność na poprawę kosztu), ale za to z dużą ilością iteracji tak, aby dopasowały się one odpowiednią kolorystyką i wielkością do fragmentu obrazu, gdzie się znajdują. Dzięki temu, że osobniki mogą także zmieniać swoją pozycję, przy większej ilości iteracji mają one możliwość znalezienia najbardziej dogodnej pozycji.

Wnioski końcowe Ostatecznie jestem usatysfakcjonowany wynikami. Algorytm w około 1000 operacji uzyskuje obraz, który szczególnie po przeskalowaniu daje dobre przybliżenie oryginału. Jeśli rozwiązanie nas nie satysfakcjonuje, to można je rozszerzyć o kolejne punkty biorąc pod uwagę, że poźniej wymagane jest ich znacznie więcej aby było widać wyraźną porprawę (widać to w rozwiązaniu z 1000 i 2000 kółek). Prędkość zbiegania można by poprawić dodając do przybliżonego już obrazu inne, bardziej kanciaste figury (np. trójkąty) tak, aby nadać obrazowi bardziej "kanciaste" kształty. Poza tym, losowe wybieranie punktów na kandydatów można by zastapić losowaniem punktu startowego na wybranym fragmencie obrazu, ze względu obliczone na nich funkcje kosztu (czyli losować punkty nie na całym przedziale, ale na fragmencie obrazu który ma najmniejszą funkcję kosztu). Oczywiście wiąże się to z dodatkowym nakładem obliczeniowym ale powinien przyspieszyć zbieganie i tym samym zmniejszyć użyte ilości punktów.