Witold Bołt

Automaty komórkowe w grafice komputerowej

25 stycznia 2007

W tym krótkim dokumencie opisano dostarczony program *cell*: jego wymagania, sposób obsługi i zasadę działania.

1. Wymagania

Program *cell* został napisany w języku C# na platformę Microsoft .NET, w związku z tym do jego uruchomienia wymagane są:

— system z rodziny Microsoft Windows (testowano w Windows XP i Windows 2003 Server);
— .NET Platform Runtime 2.0 lub nowsze[1].

2. Obsługa

Program komunikuje się z użytkownikiem za pośrednictwem (niedopracowanego niestety) graficznego interfejsu użytkownika.

Pierwszym krokiem koniecznym do dalszego użytkowania programu jest załadowanie pliku graficznego, który będzie stanowić stan początkowy naszego automatu. Obsługiwane są pliki w popularnych formatach BMP, JPEG, PNG, ... Należy przy tym pamiętać, że przetwarzanie dużych obrazków może trwać BARDZO długo (do rozsądnej pracy zalecane jest korzystanie z obrazków o rozdzielczości nie większej niż 200×200 pikseli – jeśli oczekujemy zupełnie płynnej pracy powinniśmy używać obrazki w rozdzielczości poniżej 100×100 pikseli!).

Po załadowaniu, obrazek zostanie wyświetlony w głównym oknie programu. Domyślnie zostanie on rozciągnięty do rozmiaru okna (z zachowaniem proporcji). W menu *Widok* można zmienić sposób dopasowania rozmiaru obrazka do rozmiaru okna.

Gdy mamy załadowany obrazek, możemy spróbować zastosować automaty, dostępne w menu *Automat*. Wszystkie automaty przed rozpoczęciem pracy, pozwalają ustawić pewne parametry konfiguracyjne. Ich znacznie zostanie omówione w następnym rozdziale. Tutaj ograniczymy się o dwóch parametrach wspólnych dla wszystkich automatów: liczba powtórzeń i prob. Pierwszy z nich jest oczywisty – liczba powtórzeń wykonania automatu – po każdym pojedynczym przejściu efekt zostanie zaprezentowany w głównym oknie programu. Drugim parametrem jest natomiast prawdopodobieństwo zastosowania reguły automaty dla danego punktu. Domyślnie jest to 1.0.

[1] Do pobrania za darmo ze stron Microsoft.

3. Zasada działania

Wszystkie automaty działają na przestrzeni stanów wyznaczonej przez przestrzeń kolorów w formacie RGB – każdy punkt na siatce ma stan wyznaczony przez 3 liczby całkowite, każda z przedziału $[0, 255]$, odpowiadające kolejnym kanałom: czerwonemu (R), zielonemu (G) i niebieskiemu (B). Każda komórka ma więc 2^{24} możliwych stanów. Rozpatrywane tu automaty są więc bardzo specyficzne – z reguły rozpatruje się automaty o „niewielkiej" (w ogólno-subiektywnym sensie) liczbie stanów.

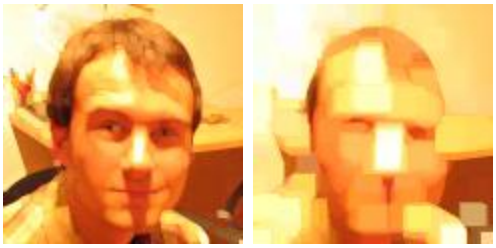
Wyniki większości automatów prezentuje na przykładzie powyższego obrazka testowego.

3.1. Automaty maksimum i minimum

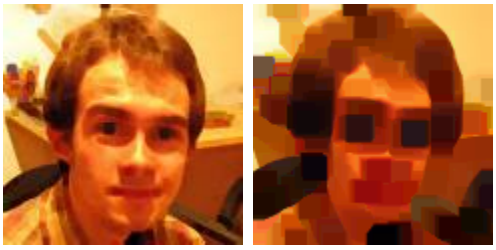
Automaty te mają reguły zdefiniowane na sąsiedztwie Moore'a. Oba z nich bazują na bardzo prostym pomyśle. Dla każdego punktu z sąsiedztwa obliczana jest jego długość, według wzoru, odpowiadającego normie euklidesowej:

$$len = \sqrt{r^2 + g^2 + b^2},$$

gdzie r, g, b oznaczają wartości kanałów koloru. W automacie maksimum każdy punkt dostaje kolor o największej długości, w automacie minimum – o najmniejszej. Oczywiście może się tak zdarzyć, że w jednym sąsiedztwie znajduje się kilka różnych kolorów o tej samej długości (leżących na jakiejś abstrakcyjnej sferze w przestrzeni kolorów) i w dodatku ich długość może być właśnie tą najmniejszą lub największą. Wówczas automat wybiera jeden losowo z jednostajnym rozkładem prawdopodobieństwa (nie ma możliwości kontrolowania tego zachowania przez użytkownika).



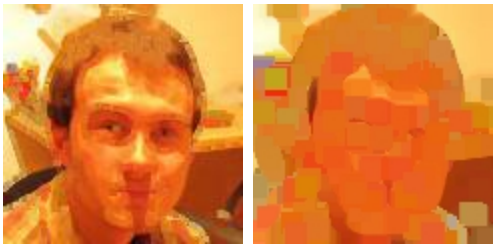
Przykład działania automatu maksimum: odpowiednio po 1 i 5 iteracji.



Przykład działania automatu minimum: odpowiednio po 1 i 5 iteracji.

3.2. Automat minimum offset

Automat ten działa bardzo podobnie do automatu minimum, z tą różnicą, że przyjmuje jeden dodatkowy parametr przesunięcia, (lub offsetu). W automacie tym „wygrywają" te komórki których długość jest najbliższa wartości przesunięcia, czyli takie dla których $|len - offset|$ jest najmniejsze. W przypadku niejednoznaczności stosowane są reguły z wcześniej opisywanych automatów.



Przykład działania: offset = 255, odpowiednio po 1 i 5 iteracji.

3.3. Automat border detect

Automat ten ma regułę zdefiniowaną na sąsiedztwie von Neumanna. Automat przyjmuje dodatkowy parametr ϵ (opisane w programie po prostu „epsilon"). Reguła tego automatu działa następująco:

1. dla każdego węzła z sąsiedztwa liczona jest jego długość len zgodnie ze wzorem podanym wcześniej, (długość węzła „środkowego" oznaczamy przez l)
2. jeśli dla każdego węzła (poza środkowym), wartość bezwzględna różnicy długości l i długości danego węzła jest większa od ϵ, to reguła zwraca wartość odpowiadającą kolorowi białemu, w przeciwnym wypadku kolorowi czarnemu.

Przykład działania automatu ($\epsilon = 17$).

3.4. Automat noise remove

Podobnie jak w poprzednim automacie, tu również definiujemy regułę na sąsiedztwie von Neumanna. Reguła zależna jest od dwóch parametrów δ i ϵ. Działanie reguły opiera się na metryce euklidesowej indukowanej wcześniej zdefiniowaną normą:

$$d[p^1, p^2] = \sqrt{(p_r^2 - p_r^1)^2 + (p_g^2 - p_g^1)^2 + (p_b^2 - p_b^1)^2},$$

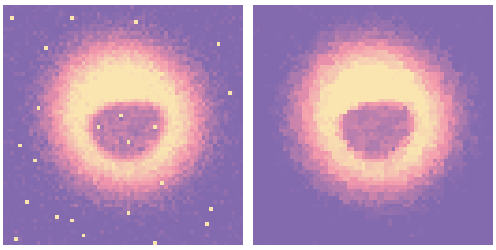
gdzie $p^i = [p_r^i, p_g^i, p_b^i]$ oznacza stan automatu.

1. Liczymy odległości (w sensie zdefiniowanej normy) od punktu środkowego do punktów sąsiednich. Jeśli maksimum tych odległości jest mniejsze bądź równe ϵ to zwracamy niezmieniony kolor środkowy.

2. W przeciwnym wypadku, szukamy największego (w sensie liczby elementów) podzbioru U punktów z całego sąsiedztwa (włączając w to punkt środkowy), który spełnia:

$$\forall_{p^1,p^2\in U}|d[p^1,B]-d[p^2,B]|\leqslant\delta,$$

   gdzie przez B oznaczamy kolor czarny (wartości $d[p,B]$ odpowiadają oczywiście wcześniej zdefiniowanej normie punktu p).
3. Zawsze istnieje przynajmniej jeden taki zbiór U. Może oczywiście istnieć ich więcej – wtedy wybieramy losowo (z równym prawdopodobieństwem) jeden taki zbiór (w krańcowe sytuacji może być on jedno punktowy).
4. Reguła zwraca kolor będący średnią arytmetyczną kolorów punktów z wybranego zbioru U.



Przykład działania automatu: po lewej obrazek oryginalny, po prawej potraktowany automatem (1 iteracja).