**Sprawozdanie z projektu I**

**Wojciech Sekta, 298273**

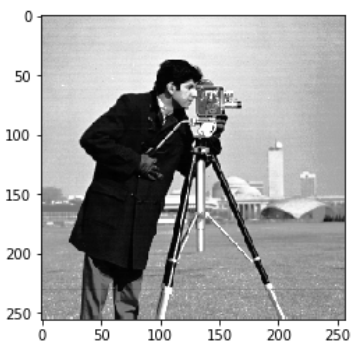
Pierwszy projekt przygotowałem jako bibliotekę w języku Python3, uruchamianą przez Jupyter Notebook (interaktywne środowisko), dla którego ciężko stwierdzić czy 7. założenie projektu jest w pełni spełnione. Jeżeli będzie stanowiło to problem, mogę zamienić rozwiązanie na wersję konsolową.

Obrazy na których były testowane algorytmy:

**Normalizacja obrazu według łamanej opisanej min. 3 punktami**

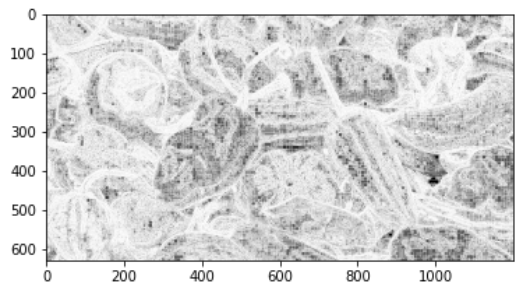
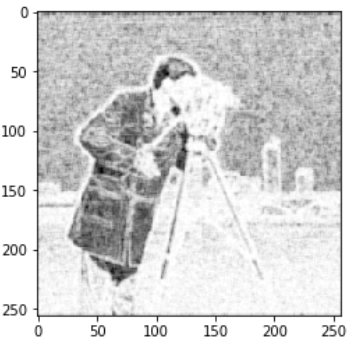
Funkcja przyjmuje jako argument obraz oraz listę tupli dwuelementowych w postaci: [(In1,Out1),(In2,Out2),…,(InN,OutN)]. Dodaje do tej listy element (0,0) oraz (256,255), a następnie sortuje po elementach InX. Następnie program bierze po dwa punkty, wyznacza z nich równanie prostej i podstawiając do tego równanai piksele, o intensywności mieszczącej się w zakresie <InX;InX+1), normalizuje obraz.



Normalizację powyższych obrazów wykonano dla punktów [(50,100),(150,150),(200,250)]

**Filtracja entropii**

Funkcja przyjmuje jako argumenty obraz i szerokość okna. Program konwolucyjnie wylicza wartość entropii a następnie powstały obraz normalizuje do przedziału <0;255>.



Wyliczone dla okna o szerokości 5 pikseli

**Otwarcie elementem linijnym**

## Funkcja przyjmuje obraz, długość elementu, oraz nachylenie. Na początku funkcja tworzy za pomocą algorytmu Bresenhama liniowy element strukturalny, następnie za pomocą tego elementu wykonuje na obrazie operacje erozji i dylatacji.

## 

Operację przeprowadzona dla elementu o długości 10 i nachyleniu 30 stopni

**Wypukłe otoczenie**

Funkcja przyjmuje jako argument obraz. Na początku tworzony jest następujący element strukturalny:

Następnie za pomocą tego elementu wykonywana jest operacja *Hit or Miss* (implementowana przy pomocy erozji). Następnie element strukturalny jest obracany o 45 stopni i powtarzana jest operacja aż do stanu w którym następna iteracja nic nie zmieni w obrazie.

