Robert Rozmus

7.01.2015

POBR – dokumentacja

Temat - wykrywanie logo *Lego*

Wykonanie – algorytm ogólny

1. Poprawa kontrastu.
2. Segmentacja.
3. Rozrost regionu.
4. Obliczanie 10 niezmienników momentowych.
5. Rozpoznanie liter na podstawie niezmienników.
6. Znalezienie napisu *Lego*.
7. Sprawdzenie, czy znaleziony napis *Lego* znajduje się na kolorze podobnym do czerwonego.

Opis poszczególnych kroków

1. Poprawa kontrastu – `contrast` = 88

```
factor = (259 * (contrast + 255)) / (255 * (259 - contrast))
colour = GetPixelColour(x, y)
newRed   = Truncate(factor * (Red(colour)   - 128) + 128)
newGreen = Truncate(factor * (Green(colour) - 128) + 128)
newBlue  = Truncate(factor * (Blue(colour)  - 128) + 128)
PutPixelColour(x, y) = RGB(newRed, newGreen, newBlue)
```

2. Segmentacja – próg = 210

   wartość piksela w skali szarości > próg -> biały

   wartość piksela w skali szarości <= próg -> czarny

3. Rozrost regionu – kolorowanie każdego białego obszaru na inny kolor.

Wykorzystanie pętli i zbioru:

```
Size size = processingImage.size();
Set<Point> allPoints = new HashSet<>();
allPoints.add(point);

while (!allPoints.isEmpty())
```

```
{
    Point p = allPoints.iterator().next();
    int x = p.x;
    int y = p.y;
    allPoints.remove(p);
    processingImage.put(x, y, currentSegmentColor);

    if (y > 0)
    {
       double[] topPoint = processingImage.get(x, y - 1);
       if (isPixelNotCheckedYetAndIsNotBlack(topPoint, currentSegmentColor))
       {
           allPoints.add(new Point(x, y - 1));
       }
    }
    if (x > 0)
    {
       double[] leftPoint = processingImage.get(x - 1, y);
       if (isPixelNotCheckedYetAndIsNotBlack(leftPoint, currentSegmentColor))
       {
           allPoints.add(new Point(x - 1, y));
       }
    }
    if (y < size.width - 1)
    {
       double[] bottomPoint = processingImage.get(x, y + 1);
       if (isPixelNotCheckedYetAndIsNotBlack(bottomPoint, currentSegmentColor))
       {
           allPoints.add(new Point(x, y + 1));
       }
    }
    if (x < size.height - 1)
    {
       double[] rightPoint = processingImage.get(x + 1, y);
       if (isPixelNotCheckedYetAndIsNotBlack(rightPoint, currentSegmentColor))
       {
           allPoints.add(new Point(x + 1, y));
       }
    }
}
```

4. Obliczanie 10 niezmienników momentowych – zgodnie ze wzorami podanymi na laboratorium dla każdego obszaru o powierzchni > 20 pikseli (min. wielkość litery).
5. Rozpoznanie liter na podstawie niezmienników

Na podstawie wielu badań sporządzonych w załączniku *momenty.xlsx* wybrano 4 niezmienniki momentowe, dla których przyjęto przedziały umożliwiające rozpoznawanie liter:

L	E	G	O
przyjęte wartości			
0.37> M1 > 0.3	0.36 >M1 > 0.27	0.35 > M1 > 0.25	0.33> M1 > 0.24

1.75>M6> 0.03	3.6 > M3 > 0.23	0.83 > M3 > 0.06	0.075 > M3 > 0.0015
	0.025> M6 > -0.035	0.35 > M4 > 0.03	0.06> M6 > 0.000003

6. Znalezienie napisu *Lego*:
    a. Sprawdzenie, czy litera E leży pomiędzy L i O
    b. Sprawdzenie, czy litera E leży obok L.
    c. Sprawdzenie, czy litera G leży pomiędzy E i O
    d. Sprawdzenie, czy litera G leży obok E.
    e. Sprawdzenie, czy litera O leży obok G.

    Czy litera A jest obok liter B i C sprawdzono algorytmem:
    - oblicz środki ciężkości A, B, C
    - oblicz odległość środka ciężkości A od prostej przechodzącej przez środki ciężkości B i C
    - sprawdź, czy odległość < 5

    Czy litera A jest obok B sprawdzono algorytmem:
    - oblicz przekątna prostokąta ograniczającego A
    - oblicz odległość euklidesową między środkami ciężkości A i B
    - sprawdź, czy odległość jest mniejsza od przekątnej

8. Sprawdzenie, czy znaleziony napis *Lego* znajduje się na kolorze podobnym do czerwonego - (r > 180, b i g < 80):
    - Oblicz wielkość *x* = odległości 3.5 razy mniejszej od przekątnej prostokąta ograniczającego cały napis *Lego*
    - Narysuj dwie równoległe linie próbek– jedna linia równoległa do linii łączącej środki ciężkości L i O o współczynniku b o *x* większym, druga o *x* mniejszym
    - Jeśli powyżej i poniżej napisu *Lego* znajdą się chociaż po 2 punkty -> czerwone tło

Wnioski

Poprawa kontrastu umożliwiła wykrycie napisy na brudnej koszulce – ostatnia strona w dokumentacji

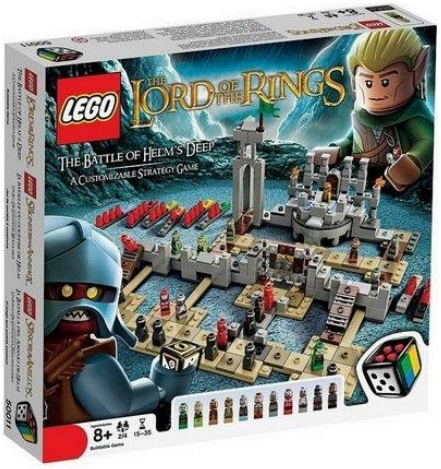
warunki poprawnego rozpoznania

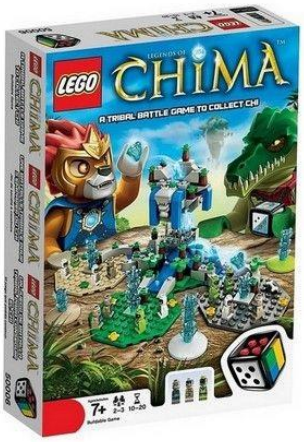
1. Każda litera musi składać się z pikseli podobnych do białego w ilości > 20.
2. Litery muszą być oddzielone o co najmniej 1 piksel na całym obwodzie o skali szarości

Testowane obrazy

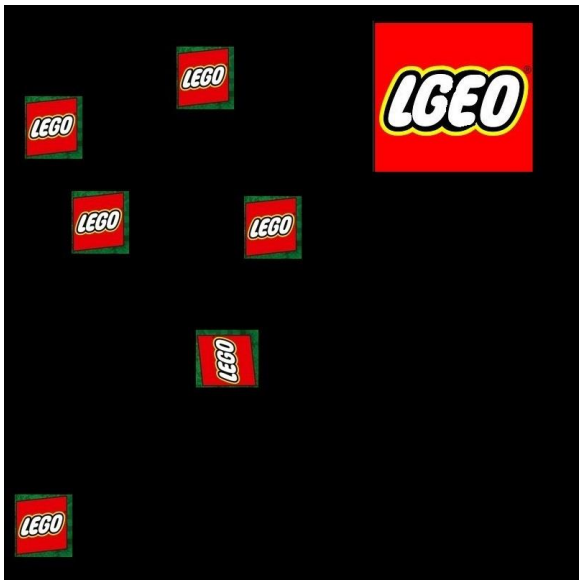
Przeprowadzono różnego rodzaju testy (wszystkie obrazy testowe w oryginalnej rozdzielczości umieszczono w folderze *obrazy*):

1. Typowe obrazy





2. Wiele logo na jednym obrazie + zmiana orientacji + zmiana kolejności liter

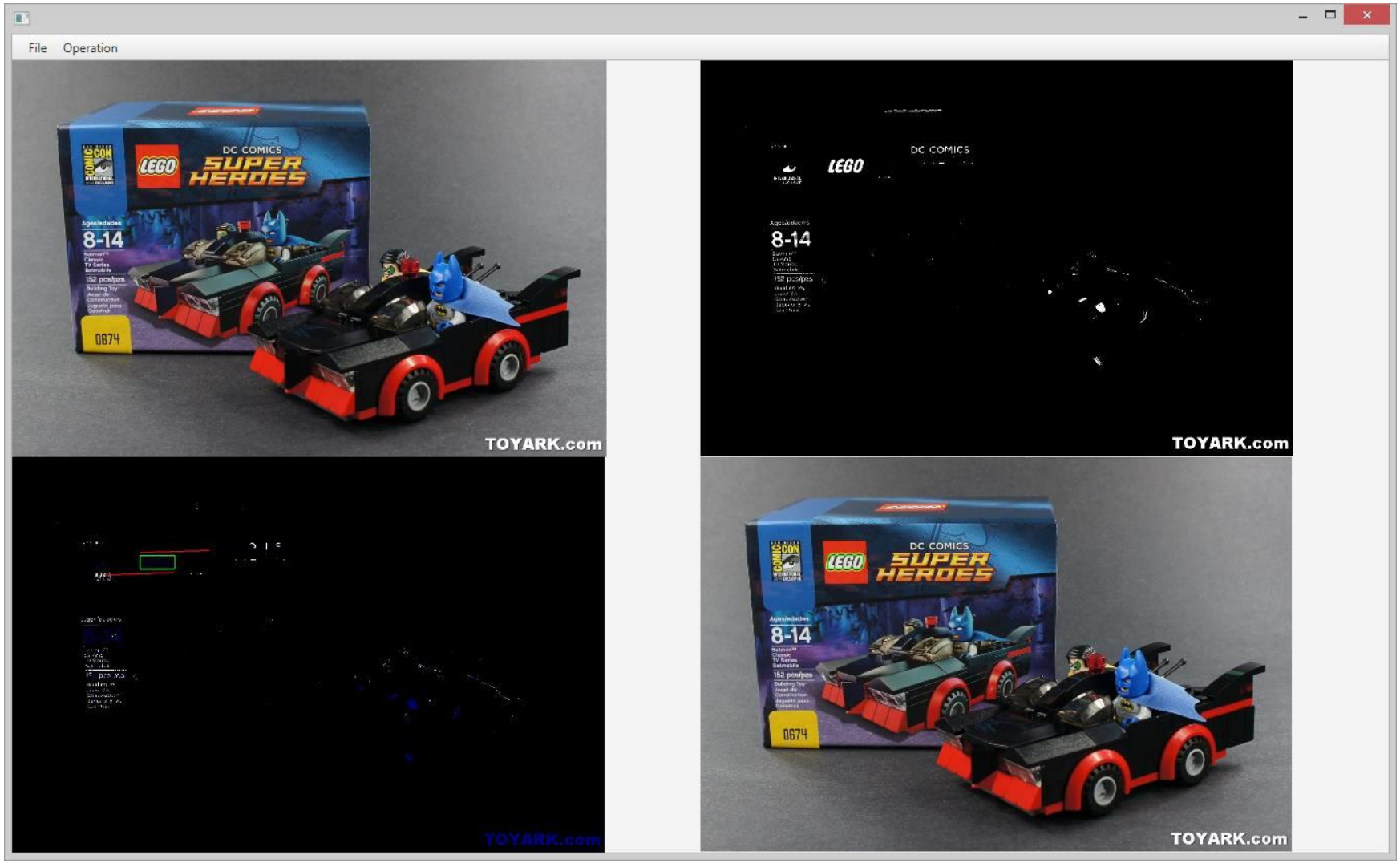


3. Zmiana kolejności liter

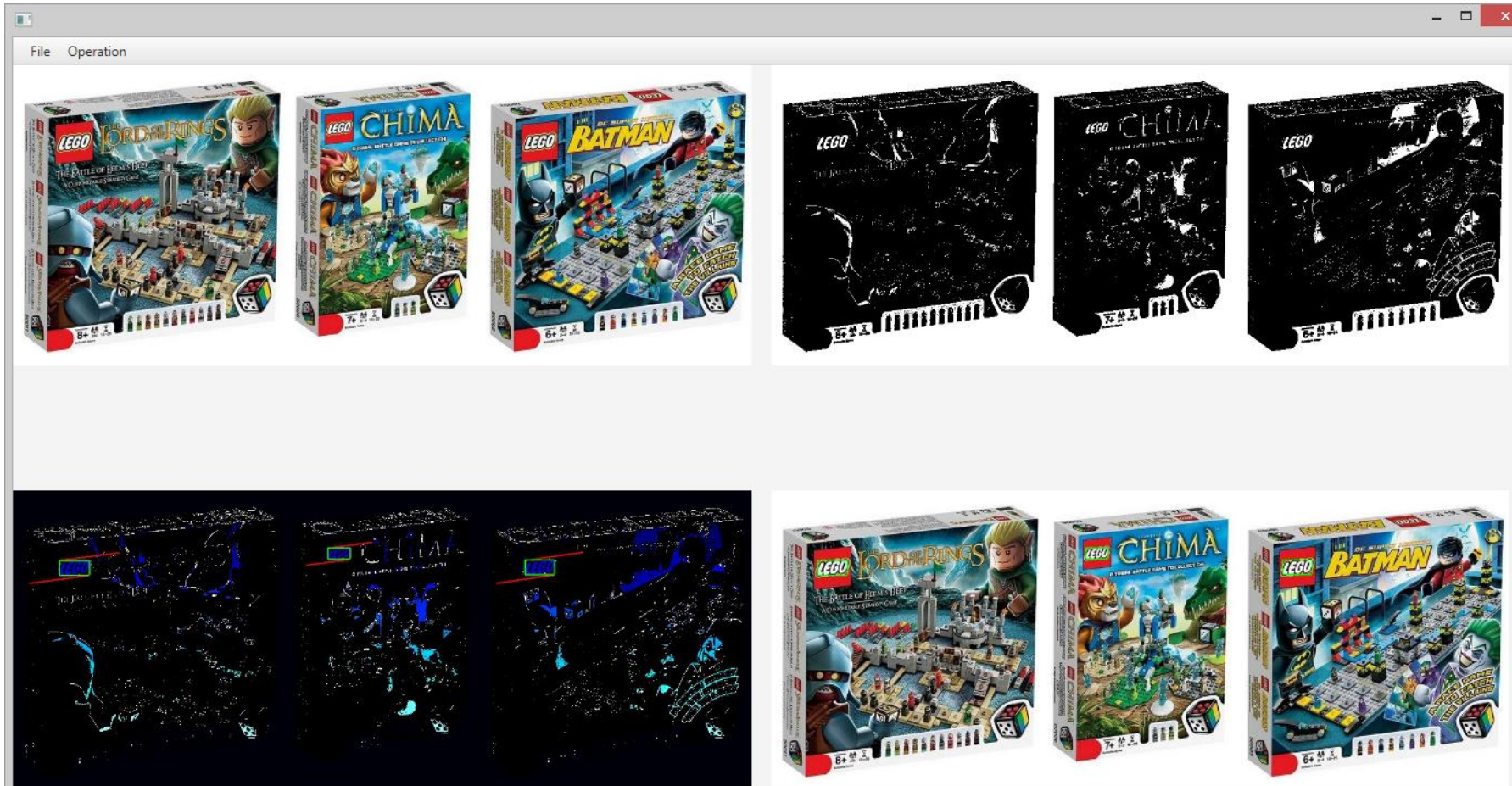
4. Zmiana koloru tła

5. Różne orientacje

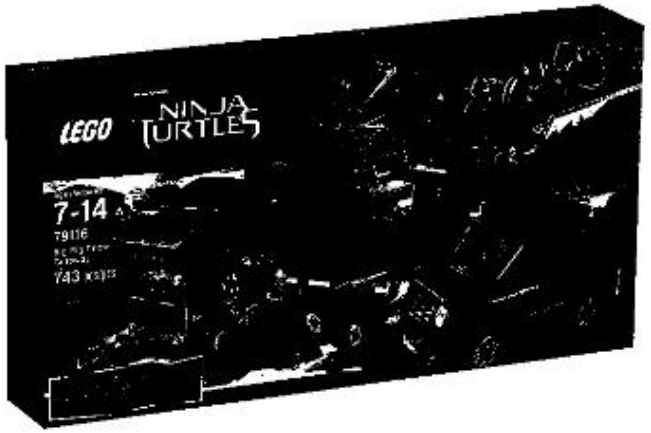
Program

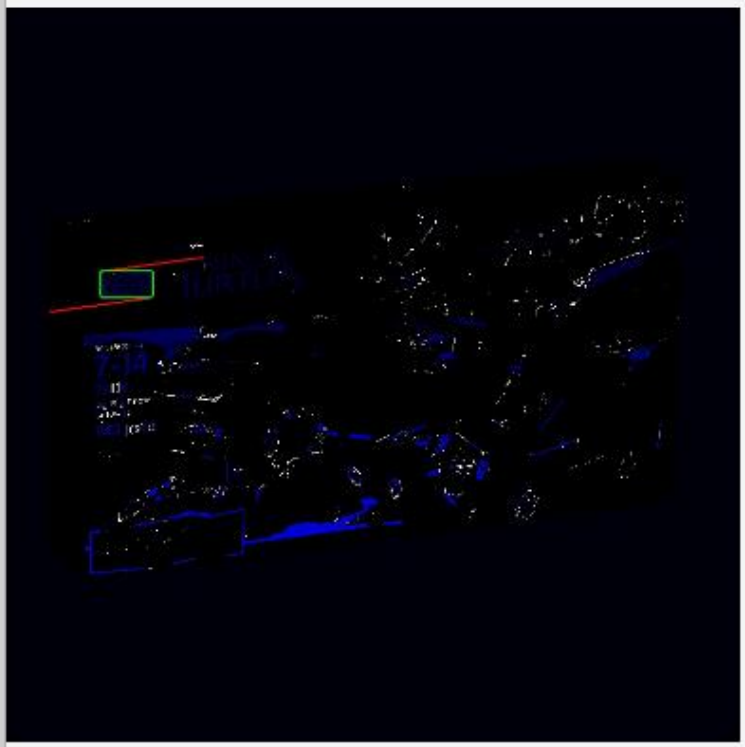

File
Operation
DC COMICS
SUPER HEROES
LEGO
8-14
0674
TOYARK.com
TOYARK.com
TOYARK.com
TOYARK.com


File
Operation
LEGO
BATMAN 3
BEYOND GOTHAM
LEGO
BATMAN 3
BEYOND GOTHAM
BEYOND GOTHAM
LEGO
BATMAN 3
BEYOND GOTHAM


File
Operation

File Operation






File
Operation
LEGO
LEGO
LEGO