Bartłomiej Niemienionek 69988
Artur Wojciechowski 66325

ROZPOZNAWANIE OBRAZÓW

Projekt

Dokument końcowy

Defektoskopia: Detekcja defektów w RTG odlewów

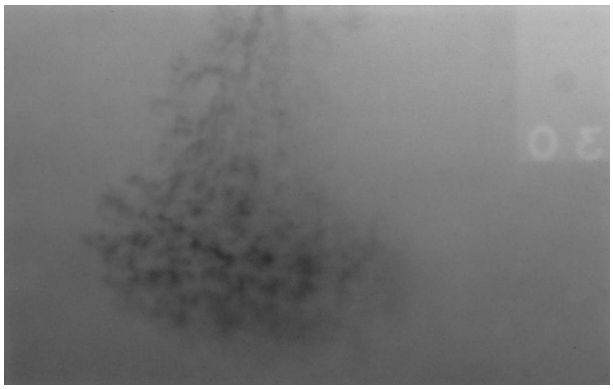
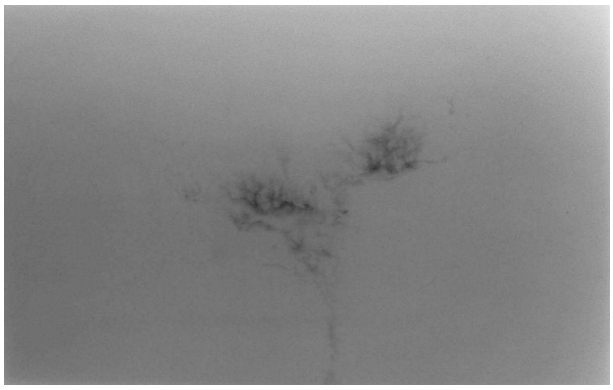
Spis treści:

1. Analiza wymagań

1.1. Cel projektu

a) Opis problemu

W procesie produkcji dużych bloków miedzianych mogą w nich powstać pewne anomalie (defekty), które klasyfikuje się na podstawie porównania zdjęć rentgenowskich tych bloków (w skali 1:1) ze zdjęciami prezentującymi określone standardy defektów.
Wyróżnia się trzy (b, c, d) klasy defektów, wśród których dodatkowo określa się pięciostopniową skalę wielkości uszkodzenia (w skali od 1 do 5).
Przykładowe zdjęcia przedstawiające określone wady defektów wyglądają następująco:

Wada C 5	Wada D 5
	

b) Ogólny cel projektu

Celem projektu prowadzonego w roku akademickim 2006/2007 była kontynuacja oraz poprawienie (właściwie przepisanie od nowa) projektu prowadzonego przez jednego ze studentów w roku poprzednim.
W ramach projektu zaproponowano metodę automatycznej klasyfikacji defektów powstałych na powierzchniach miedzianych bloków oraz stworzenie aplikacji, która ją wykorzystując pozwoli na rozpoznawanie standardowych anomalii oraz umożliwi zakwalifikowanie defektów niestandardowych (znajdujących się na rzeczywistych odlewach).

c) Dane

- wejściowe:
  W fazie uczenia zbiór obrazów reprezentujących standardowe klasy defektów.
  W późniejszym etapie działania aplikacji plik graficzny zawierający zdjęcie rentgenowskie rzeczywistego miedzianego bloku.

- wyjściowe:
  Prawdopodobieństwo przynależności defektu występującego na zdjęciu do odpowiednich klas.

1.2. Osadzenie projektu w dziedzinie rozpoznawania obrazów

W projekcie wykorzystano uczenie maszynowe w bardzo prostej formie. Jako zbiór uczący posłużył zbiór zdjęć prezentujących standardowe klasy i skale uszkodzeń (defektów).
Zastosowano podejście oparte na niebezpośrednich cechach obrazu, a mianowicie na macierzy (bądź zbiorze macierzy) współwystąpień.

1.3. Kryteria oceny wyników

Dokładność klasyfikatora definiuje się jako procent przykładów testowych poprawnie zaklasyfikowanych przez klasyfikator. Ważne jest, że dokładności klasyfikatora nie testujemy na zbiorze treningowym, a na zbiorze testowym. Ponieważ w takim przypadku otrzymane rezultaty byłyby nadmiernie optymistyczne.
Dodatkowo istotnymi wartościami pozwalającymi na określenie trafności klasyfikacji są:
- liczba przypadków, nienależących do danej klasy, błędnie do niej zaklasyfikowanych (False Positive)
- liczba przypadków, należących do danej klasy, błędnie zaklasyfikowanych do innej (False Negative)

1.4. Środowisko realizacji projektu

W realizacji projektu wykorzystano środowisko Visual Studio 2005 oraz język C#. Wstępnie do operacji dokonywanych na obrazach planowano wykorzystanie biblioteki OpenCV (używając odpowiedniego wrappera, przykładowo SharperCV). Jednakże po wstępnej analizie, która pozwoliła określenie, które funkcje tej biblioteki zostaną użyte (wyłącznie dostęp do poszczególnych punktów obrazu – przy obliczaniu macierzy współwystąpień oraz bardzo specyficzne (opisane w dalszej części dokumentu) przekształcenia obrazu), aby zaoszczędzić czas, zrezygnowano z poszukiwania możliwości podłączenia biblioteki OpenCV do projektu.

2. Opis metodologii

2.1. Wstępne przetwarzanie obrazu

Wstępna obróbka obrazu ma na celu wyrównanie jasności obrazu, jego dyskretyzację oraz uwydatnienie fragmentów zawierających defekty.
Preprocessing obrazu przebiega następującej kolejności:

1. Konwersja do skali szarości;
2. Zastosowanie rozmycia gaussowskiego i obliczenie różnicy pomiędzy obrazem rozmytym i oryginalnym;
3. Obliczenie histogramu;
4. Na podstawie wcześniej obliczonego histogramu wybierany jest punkt bieli (punkt czerni pozostaje bez zmian). W ustawieniach (white treshold) można określić ile procent najjaśniejszych pikseli ma zostać odciętych i uznanych za białe;
5. Przygotowywane są kopie obrazu o mniejszej rozdzielczości (każda kolejna kopia ma 2 razy mniejsza rozdzielczość - z 4 pikseli powstaje 1);
6. Wszystkie kopie obrazu zostają poddane dyskretyzacji. Wykorzystywany jest wcześniej obliczony punkt bieli. Obszar pomiędzy punktem bieli i punktem czerni jest dzielony na równoodległe przedziały;
7. W celu usunięcia szumów (wzmocnionych podczas obliczania różnicy między obrazem rozmytym i oryginalnym) dla każdego piksela obliczana jest mediana o promieniu 1;

Ze względu na specyfikę przeprowadzanych operacji (np. rozmycie o dużym promieniu – przykładowo 40 px) zostały one dodatkowo samodzielnie zaimplementowane, a nie wykorzystano możliwości bibliotek graficznych, a ich dokładniejszy opis znajduje się w punkcie 3. Opis implementacji.
Zoptymalizowane zostały wymagania pamięciowe związane z przechowywaniem obrazów, a mianowicie wykorzystano kanały obrazu do przechowywania dodatkowych informacji (kanał 0 – wartości po dyskretyzacji z przedziału od 0 do liczy poziomów dyskretyzacji, kanał 1 – wartości odpowiadające skali szarości obrazu, kanał 2 – wartości maski przypisanej, która pozwoli na pominięcie mało interesujących fragmentów obrazu podczas jego analizy). Efektem ubocznym takiego podejścia jest zmiana koloru obrazu – dominuje kolor czerwony.
Dodatkowo użytkownik może „ręcznie” wybrać interesujący go obszar, który ma być poddany analizie
Tak przygotowany obraz poddawany jest analizie w celu odnalezienia pożądanych cech charakterystycznych.

2.2. Analiza obrazu

W proponowanym podejściu wykorzystano macierz współwystąpień, która pozwala na przechowywanie informacji na temat struktury tekstury oraz umożliwia wyliczenie niektórych cech obrazu takich jak np.: maksimum, moment różnicowy elementy rzędu k, entropię.
Macierz współwystąpień definiuje się w następujący sposób:
Niech P będzie operatorem pozycji (relacją, w jakiej mogą się znaleźć dwa punkty obrazu).
Macierz A definiowana jest jako macierz kwadratowa o rozmiarach k x k (k - liczba poziomów jasności), której wartości elementów dla obrazu f zdefiniowane są następująco:

$$a_{i,j}=|\{p: f(p)=i \wedge P(p,r) \wedge f(r)=j\}|$$

gdzie p i r to punkty obrazu, natomiast zapis P(p, r) oznacza, że punkty p i r są w relacji przestrzennej P.
Macierz współwystąpień to macierz A znormalizowana wielkością obrazu:

$$c_{i,j}=\frac{a_{i,j}}{|D(f)|}$$

Jest to zatem estymata łącznego rozkładu prawdopodobieństwa napotkania w obrazie pary punktów o określonych jasnościach, pozostających w relacji P.
Dla zachowania większej zwięzłości opisu (zajętość pamięci) oraz zwiększenia reprezentatywności stosuje się dyskretyzację poziomów jasności.

Dla każdej kopii obrazu (związanej z różną rozdzielczością) obliczana jest macierz współwystąpień.

2.3.Wnioskowanie i klasyfikacja

Przeanalizowane obrazy, dla których można jednoznacznie określić przynależność do poszczególnych klas defektów, można dodawać do zbioru uczącego.
Zbiór uczący zapamiętywany jest w plikach.xml o następującej strukturze:

```
<Matrix>
  <Class name="B" />
  <Strenght value="1" />
  <Levels>
    <Level DescreteLevels="4">
      <CoocuranceMatrix>
        <value x="0" y="0" val="0,761057930432964" />
            <value ...
      </CoocuranceMatrix>
      <CoocuranceMatrix>
        <value x="0" y="0" val="0,800882411263137" />
            <value ...
      </CoocuranceMatrix>
      ...
    </Level>
    <Level DescreteLevels="6">
      <CoocuranceMatrix>
        <value x="0" y="0" val="0,495999362621186" />
            <value ...
      </CoocuranceMatrix>
      ...
    </Level>
    <Level DescreteLevels="8">
      <CoocuranceMatrix>
        <value x="0" y="0" val="0,0781346507144365" />
            <value ...
      </CoocuranceMatrix>
     ...
    </Level>
    <Level DescreteLevels="12">
      <CoocuranceMatrix>
        <value x="0" y="0" val="0,0781346507144365" />
            <value ...
```

```
      </CoocuranceMatrix>
      ...
    </Level>
    <Level DescreteLevels="16">
      <CoocuranceMatrix>
        <value x="0" y="0" val="0,0781346507144365" />
            <value ...
      </CoocuranceMatrix>
      ...
    </Level>
  </Levels>
</Matrix>
```

Pojedyncza macierz tworzy punkt w przestrzeni wielowymiarowej, następnie stosując klasyfikator NN (najbliższego sąsiada). Po porównaniu nowoobliczonej macierzy współwystąpień z odpowiednimi macierzami wchodzącymi w skład zbioru uczącego wybieranych jest n (domyślnie 4) najbliższych sąsiadów i wyniki dla każdej z macierzy są uśredniane.
W przypadku gdy dla danego obrazu posiadamy obliczone dane dla różnej liczby poziomów dyskretyzacji to wyniki o wyższej liczbie poziomów są bardziej istotne.

3. Opis implementacji

Ciakwymi elementami kodu aplikacji są niektóre fragmenty umożliwiające wstępne przetworzenie obrazu, jak również istotny jest sposób obliczania zbioru macierzy współwystąpień reprezentujących dany obraz.

a) Preprocessing obrazu

Poniżej przedstawiono dokładniejszy opis poszczególnych kroków przygotowania obrazu do analizy.

1 - Konwersja do skali szarości jest wykonywana standardowo.

2 - Zastosowanie rozmycia gaussowskiego i obliczenie różnicy pomiędzy obrazem rozmytym i oryginalnym.
Filtr gaussa zaimplementowano wykorzystując fakt, że kolejno zastosowane rozmycia poziome, a następnie pionowe daje ten sam efekt co standardowe rozmycie kwadratowe, uzyskuje się jednak znaczne przyspieszenie obliczeń co jest bardzo istone ze względu na fakt, iż promień rozmycia jest bardzo duży.

kod:

```
public static bool FastGaussianBlurAndDiff(Photo photo)
    {
        Bitmap b = photo.modified;
        BitmapData bmData = b.LockBits(new Rectangle(0, 0, b.Width, b.Height),
ImageLockMode.ReadWrite, b.PixelFormat);

        int stride = bmData.Stride;
        System.IntPtr Scan0 = bmData.Scan0;

        unsafe
        {
            byte* p = (byte*)(void*)Scan0;
            int nOffset = stride - b.Width * 3;
            int nWidth = b.Width;
            int nHeight = b.Height;

            FgbKernel gk = new FgbKernel(photo.workerSettings.blurr);

            int pixelAvg = (int)photo.avgFromSel();

            int pixelCount = b.Width * b.Height;
            int[] grey = new int[pixelCount];
            int[] grey2 = new int[pixelCount];

            int index = 0;
            for (int y = 0; y < nHeight; ++y)
            {
                for (int x = 0; x < nWidth; ++x)
                {
                    if (p[2] == 255)
                    {
                        grey[index] = (int)p[0];
                    }
                    else if (p[2] == 0)
                    {
                        //grey[index] = pixelAvg;
```

```
                    grey[index] = -1;
                }
                else
                {
                    //grey[index] = (p[0] * p[2] + pixelAvg * (255 -
p[2])) / 255;
                    grey[index] = -1;
                }
                index++;
                p += 3;
            }
            p += nOffset;
        }

        int gSum;
        int sum;
        int read;
        int start = 0;
        index = 0;
        for (int i = 0; i < nHeight; i++)
        {
            for (int j = 0; j < nWidth; j++)
            {
                gSum = sum = 0;
                read = index - gk.radius;

                for (int z = 0; z < gk.kernel.Length; z++)
                {
                    if (read >= start && read < start + nWidth &&
grey[read] >= 0)
                    {
                        gSum += gk.multable[z, grey[read]];

                        sum += gk.kernel[z];
                    }
                    ++read;
                }
                if (sum > 0)
                {
                    grey2[index] = (gSum / sum);
                }
                else
                {
                    grey2[index] = -1;
                }

                ++index;
            }
            start += nWidth;
        }

        int tempy;
        p = (byte*)(void*)Scan0;

        for (int i = 0; i < nHeight; i++)
        {
            int y = i - gk.radius;
            start = y * nWidth;
            for (int j = 0; j < nWidth; j++)
            {
                gSum = sum = 0;
```

```
                                read = start + j;
                                tempy = y;
                                for (int z = 0; z < gk.kernelWidth; z++)
                                {
                                    if (tempy >= 0 && tempy < nHeight &&
grey2[read]>=0)
                                    {
                                        gSum += gk.multable[z, grey2[read]];

                                        sum += gk.kernel[z];
                                    }
                                    read += nWidth;
                                    ++tempy;
                                }
                                if (sum > 0)
                                {
                                    p[1] = (byte)(gSum / sum);
                                    p[0] = (byte)((p[0] < p[1]) ? p[1] - p[0] : p[0] -
p[1]);
                                    //p[0] = (byte)((p[0] < wynik) ? wynik - p[0] : 0);

                                    //p[0] = p[1] = p[2] = wynik;
                                }
                                p += 3;
                            }
                            p += nOffset;
                        }

                    }

                    b.UnlockBits(bmData);

                    return true;
                }
            }
```

3 - Obliczenie histogramu przebiega standardowo.

4 - Na podstawie wcześniej obliczonego histogramu wybierany jest punkt bieli (punkt czerni pozostaje bez zmian). W ustawieniach (white treshold) można określić ile procent najjaśniejszych pikseli ma zostać odciętych i uznanych za białe.

kod:

```
public void firstPikDetection()
        {
            byte min = byte.MaxValue;
            byte max = byte.MinValue;

            sumSelHist = 0;
            for (int i = 0; i < 256; i++)
            {
                sumSelHist += selHistogram[i];
            {

            double highCount = (workerSettings.high * sumSelHist);

            double sumator = 0;
            int width = 10;
            double[] elements = new double[width];
            for (int i = 0; i < width; i++)
```

```
            }
                elements[i % width] = selHistogram[i];
                sumator += selHistogram[i];
            }
            for (int i = width; i < 256; i++)
            {
                sumator -= elements[i % width];
                elements[i % width] = selHistogram[i];
                sumator += selHistogram[i];
                if (sumator < highCount)
                {
                    max = (byte)(i - 10);
                    break;
                }
            }

            workerSettings.minGrayValue = 0;
            workerSettings.maxGrayValue = max;
        }
```

5 – W etapie przygotowywania dodatkowych kopii obrazu nie wykorzystuje się żadnych ciekawych metod.

6 - Wszystkie kopie obrazu zostają poddane dyskretyzacji. Wykorzystywany jest wcześniej obliczony punkt bieli. Obszar pomiędzy punktem bieli i punktem czerni jest dzielony na równoodległe przedziały.

kod:

```
public static bool DiscretizeBMP(Bitmap b, WorkerSettings ws)
        {
            BitmapData bmData = b.LockBits(new Rectangle(0, 0, b.Width, b.Height),
ImageLockMode.ReadWrite, b.PixelFormat);
            int stride = bmData.Stride;
            System.IntPtr Scan0 = bmData.Scan0;

            byte levels = (byte)(ws.descreetLevels - 1);
            byte min = ws.minGrayValue;
            byte max = ws.maxGrayValue;
            double step = ((double)(ws.maxGrayValue - ws.minGrayValue)) / levels;

            double perStep = 1.0 / step;
            step = 255.0 / levels;
            unsafe
            {
                int nOffset = stride - b.Width * 3;
                int nWidth = b.Width;
                int nHeight = b.Height;

                byte* p = (byte*)(void*)Scan0;

                for (int y = 0; y < nHeight; ++y)
                {
                    for (int x = 0; x < nWidth; ++x)
                    {
                        if (p[0] <= min)
                        {
                            p[0] = p[1] = 0;
                        }
                        else if (p[0] >= max)
```

```
                    }
                        p[0] = p[1] = levels;
                    }
                    else
                    {
                        p[0] = p[1] = (byte)((p[0] - min) * perStep);
                    }
                    //p[2] - maska
                    // p[1] = (byte)(p[0] * step);
                    p += 3;
                }
                p += nOffset;
            }
        }

        b.UnlockBits(bmData);

        return true;
    }
```

7 – Usuwanie szumów, wzmocnionych podczas wyznaczania różnicy pomiędzy obrazem rozmytym i oryginalnym, polega na obliczeniu dla każdego piksela mediany o promieniu 1.

kod:

```
public static bool MedianaOneLevel(Bitmap b, int poziomy)
        {
            BitmapData bmData = b.LockBits(new Rectangle(0, 0, b.Width, b.Height),
ImageLockMode.ReadWrite, b.PixelFormat);
            int stride = bmData.Stride;
            System.IntPtr Scan0 = bmData.Scan0;

            unsafe
            {
                int nOffset = stride - b.Width * 3;
                int nWidth = b.Width - 2;
                int nHeight = b.Height - 2;

                byte* p = (byte*)(void*)Scan0;

                byte* p0;

                int i;
                Directions direc = new Directions(nOffset, b.Width); //tabela z z
wartosciami o jakie trzeba przesunac wskaznik zeby odczytac danego sasiada
                Medianer med = new Medianer(poziomy);

                p = p + b.Width * 3 + nOffset; //pierwszy wiersz
                for (int y = 0; y < nHeight; ++y)
                {

                    p += 3; //pierwsza kolumna
                    for (int x = 0; x < nWidth; ++x)
                    {
                        if (p[2] == 255)
                        {
                            med.clean();
                            for (i = 0; i < 8; i++)
                            {
                                p0 = p + direc.ofsets[i];
```

```
                                        med.lista[p0[1]]++;
                                    }
                                    med.lista[p[1]]++;

                                    p[0] = med.median();
                                }
                                p += 3;
                            }
                            p += 3; //ostatnia kolumna

                            p += nOffset;
                        }
                    }

                    b.UnlockBits(bmData);

                    return true;
                }

            public class Medianer
            {
                public int[] lista;
                int poziomy;

                public Medianer(int poziomy)
                {
                    lista = new int[poziomy];
                    this.poziomy = poziomy;
                }

                public byte median()
                {
                    int max = 0;
                    int ile = 0;
                    for (int i = 0; i < poziomy; i++)
                    {
                        if (lista[i] > ile)
                        {
                            max = i;
                            ile = lista[i];
                        }
                    }
                    return (byte)max;
                }

                public void clean()
                {
                    for (int i = 0; i < poziomy; i++)
                    {
                        lista[i] = 0;
                    }
                }
            }
```

b) Obliczanie macierzy współwystąpień oraz wnioskowanie na jej podstawie

Dla każdej kopii obrazu, związanej z wieloma analizowanymi rozdzielczościami obliczana jest macierz współwystąpień wykorzystując następującą metodę:

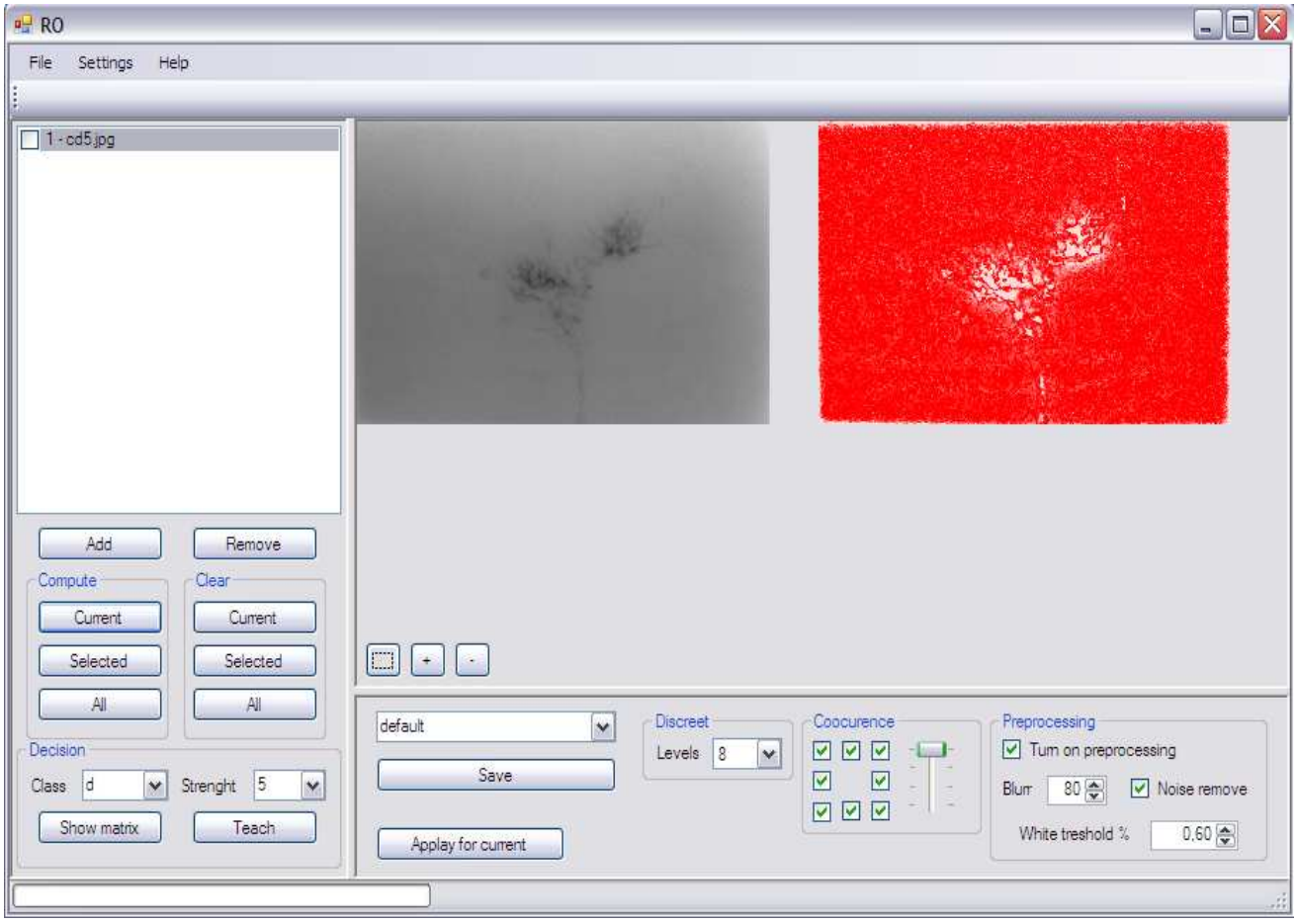
kod:

```
public static bool CoocurenceOneLevel(Bitmap b, WorkerSettings ws, int[,] matrix)
        {
        BitmapData bmData = b.LockBits(new Rectangle(0, 0, b.Width, b.Height),
ImageLockMode.ReadWrite, b.PixelFormat);
        int stride = bmData.Stride;
        System.IntPtr Scan0 = bmData.Scan0;


        //byte levels = (byte)photo.workerSettings.descreetLevels;
        unsafe
        {
            int nOffset = stride - b.Width * 3;
            int nWidth = b.Width - 2;
            int nHeight = b.Height - 2;

            byte* p = (byte*)(void*)Scan0;

            byte* p0;

            int i;
            Directions direc = new Directions(nOffset, b.Width, ws.coocurenceSettings);
              //tabela z z wartosciami o jakie trzeba przesunac wskaznik zeby
              odczytac wybranego sasiada

            p = p + b.Width * 3 + nOffset; //pierwszy wiersz
            for (int y = 0; y < nHeight; ++y)
            {

                p += 3; //pierwsza kolumna
                for (int x = 0; x < nWidth; ++x)
                {
                    for (i = 0; i < direc.count; i++)
                    {
                        if (p[2] ==255)
                        {
                            p0 = p + direc.ofsets[i];
                            matrix[p[0], p0[0]]++;
                        }
                    }

                    p += 3;
                }
                p += 3; //ostatnia kolumna

                p += nOffset;
            }
        }

        b.UnlockBits(bmData);

        return true;
    }
```

4. Interfejs użytkownika

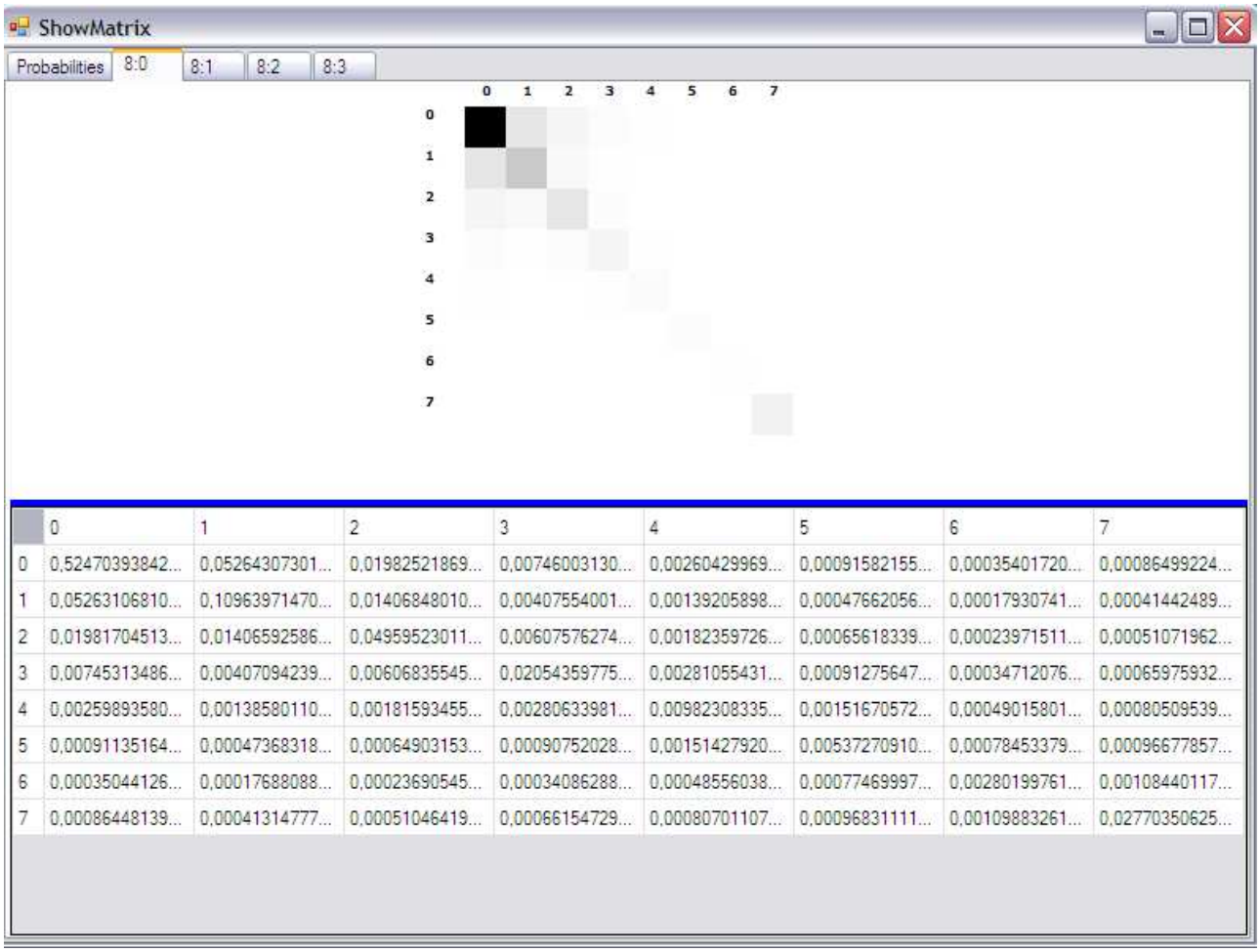
4.1. Główny interfejs programu i jego możliwości

Graficzny interfejs użytkownika (GUI) umożliwia m.in.:
- wczytywanie zdjęć oraz podgląd oryginalnego obrazu, jak również obrazu będącego efektem przeprowadzenia wstępnej obróbki
- wybór opcji wykorzystywanych podczas analizy obrazu, m.in.:
- wybór poziomów dyskretyzacji obrazu (4, 6, 8, 12, 16)
- wybór analizowanych sąsiadów dla poszczególnych pikseli podczas obliczana macierzy współwystąpień oraz określenie ich odległości od analizowanego piksela
- wybór opcji wykorzystywanych podczas wstępnej obróbki obrazu (poziom bieli, rozmiar rozmycia, czy redukcji szumu)
- uruchomienie analizy defektów dla obecnie wyświetlanego pliku, wszystkich wczytanych bądź dla zaznaczonej grupy plików
- zaklasyfikowanie analizowanego defektu do odpowiedniej klasy (opcja wykorzystywana będzie podczas tworzenia zbioru uczącego) bądź wskazanie najbardziej prawdopodobnej klasy dla badanego defektu.



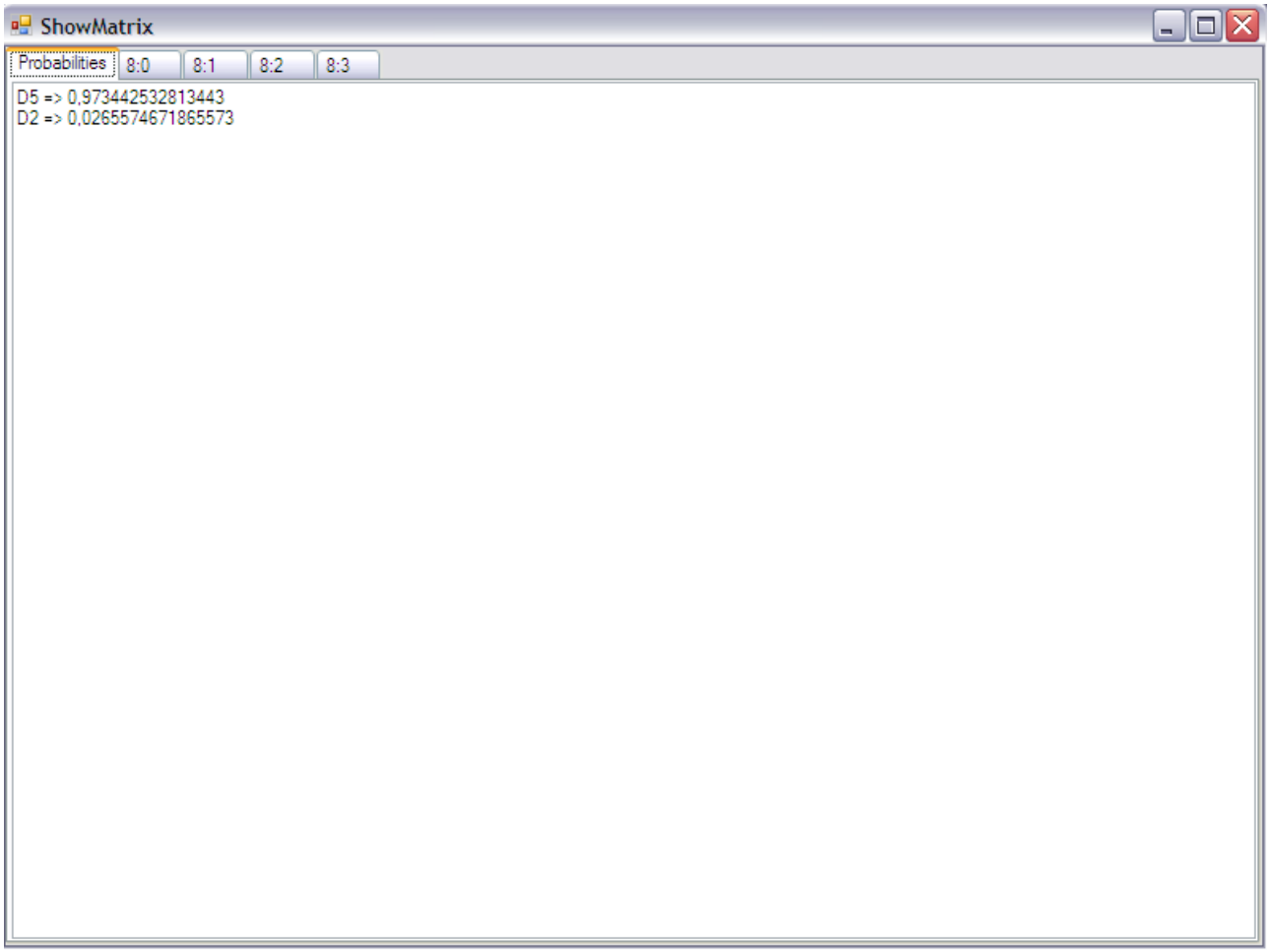
4.2. Wizualizacja macierzy współwystąpień

Dodatkową opcją, jest wizualizacja macierzy współwystąpień. Widoczne są zarówno poszczególne wartości, jak również ich graficzna interpretacja.



	0	1	2	3	4	5	6	7
0	0,52470393842...	0,05264307301...	0,01982521869...	0,00746003130...	0,00260429969...	0,00091582155...	0,00035401720...	0,00086499224...
1	0,05263106810...	0,10963971470...	0,01406848010...	0,00407554001...	0,00139205898...	0,00047662056...	0,00017930741...	0,00041442489...
2	0,01981704513...	0,01406592586...	0,04959523011...	0,00607576274...	0,00182359726...	0,00065618339...	0,00023971511...	0,00051071962...
3	0,00745313486...	0,00407094239...	0,00606835545...	0,02054359775...	0,00281055431...	0,00091275647...	0,00034712076...	0,00065975932...
4	0,00259893580...	0,00138580110...	0,00181593455...	0,00280633981...	0,00982308335...	0,00151670572...	0,00049015801...	0,00080509539...
5	0,00091135164...	0,00047368318...	0,00064903153...	0,00090752028...	0,00151427920...	0,00537270910...	0,00078453379...	0,00096677857...
6	0,00035044126...	0,00017688088...	0,00023690545...	0,00034086288...	0,00048556038...	0,00077469997...	0,00280199761...	0,00108440117...
7	0,00086448139...	0,00041314777...	0,00051046419...	0,00066154729...	0,00080701107...	0,00096831111...	0,00109883261...	0,02770350625...

Dodatkowo użytkownik może zapoznać się z prawdopodobieństwami przynależności do poszczególnych klas defektów.


ShowMatrix
Probabilities
8:0
8:1
8:2
8:3
D5 => 0,973442532813443
D2 => 0,0265574671865573

5. Opis przeprowadzonych testów oraz trafności klasyfikacji

Ze względu na bardzo małą liczbę zdjęć prezentujących poszczególne defekty niemożliwe było bardzo dogłębne przetestowanie proponowanej metody. Postanowiono modyfikować zdjęcia wchodzące w skład zbioru uczącego (obrót o pewien kąt, zmiana ROI, zmiana sąsiedztwa) i w ten sposób przeprowadzić testy jakości klasyfikacji.
Test 1:

Opis testu:
Zbiór testowy stanowią zdjęcia wchodzące w skład zbioru uczącego.

Otrzymane wyniki:

Poprawnie zaklasyfikowane	Błędnie zaklasyfikowane
100%	0%

Test 2:

Opis testu:
Zbiór testowy stanowią zdjęcia wchodzące w skład zbioru uczącego, dodatkowo znacznie ograniczono ROI.

Otrzymane wyniki:

Poprawnie zaklasyfikowane	Błędnie zaklasyfikowane
47%	53%

Test 3:

Opis testu:
Zbiór testowy stanowią zdjęcia wchodzące w skład zbioru uczącego, dodatkowo znacznie ograniczono ROI, dodatkowo wybrano jako sąsiedztwo punkty: NW,N,NE,E.

Otrzymane wyniki:

Poprawnie zaklasyfikowane	Błędnie zaklasyfikowane
47%	53%

Test 4:

Opis testu:
Zbiór testowy stanowią zdjęcia wchodzące w skład zbioru uczącego, dodatkowo znacznie ograniczono ROI, dodatkowo wybrano jako sąsiedztwo punkty: NW,N,NE,E.

Otrzymane wyniki:

Poprawnie zaklasyfikowane	Błędnie zaklasyfikowane
47%	53%

Test 5:

Opis testu:
Zbiór testowy stanowią zdjęcia wchodzące w skład zbioru uczącego obrócone o kąt od 0^o do 270^o losowo o kąt 90^o.

Otrzymane wyniki:

Poprawnie zaklasyfikowane	Błędnie zaklasyfikowane
100%	0%

Test 6:

Opis testu:
Zbiór testowy stanowią zdjęcia wchodzące w skład zbioru uczącego obrócone o kąt od 0^o do 270^o losowo o kąt 90^o, dodatkowo znacznie ograniczono ROI.

Otrzymane wyniki:

Poprawnie zaklasyfikowane	Błędnie zaklasyfikowane
67%	33%

Test 7:

Opis testu:
Na zdjęciach znacznie ograniczono ROI, dodatkowo w etapie przygotowania obrazu do analizy o fazę usuwania szumu.

Otrzymane wyniki:

Poprawnie zaklasyfikowane	Błędnie zaklasyfikowane
7%	93%

Test 8:

Opis testu:
Na zdjęciach znacznie ograniczono ROI, dodatkowo promień rozmycia równy 10.

Otrzymane wyniki:

Poprawnie zaklasyfikowane	Błędnie zaklasyfikowane
7%	93%

Test 9:

Opis testu:
Na zdjęciach znacznie ograniczono ROI, dodatkowo promień rozmycia równy 160.

Otrzymane wyniki:

Poprawnie zaklasyfikowane	Błędnie zaklasyfikowane
0%	100%

Test 10:

Opis testu:
Wartości promienia rozmycia 40 oraz 16 poziomów dyskretyzacji.

Otrzymane wyniki:

Poprawnie zaklasyfikowane	Błędnie zaklasyfikowane
14%	86%

Wady na zdjęciach mają charakter lokalny tzn. nie występują one na całym obrazie tylko na jego fragmencie, dlatego macierz współwystąpień ma różne wartości w zależności od tego ile procent analizowanego obszaru jest pokryte wada
Macierz współwystąpień dobrze opisuje tekstury natomiast w tym przypadku wady często mają charakter pojedynczych zaciemnień, kropek, albo kilku kresek, są to pojedyncze elementy.
Dodatkowym problemem jest mały zbiór uczący, dla każdej wady jest tylko jeden przykład. Gdyby wady miały charakter globalny tzn. cały obraz byłby pokryty jednakowymi zaburzeniami i tworzył teksturę wówczas z tych pojedynczych przykładów można by pobrać po kilka próbek z których każda byłaby tylko częścią obrazu, w tym przypadku możliwe jest tylko pobranie co najwyżej 2 próbek z każdego zdjęcia: jednej z całego zdjęcia, drugą natomiast z obszaru na którym w subiektywnej ocenie występuje wada (co jest dodatkowo utrudnione ze względu na fakt braku kontaktu ze specjalistą, który mógłby w tym przypadku znacznie pomóc).
Przykładowe zdjęcia nie były robione z myślą o obróbce cyfrowej, o wykorzystaniu ich do uczenia maszynowego, posiadają one dużo skaz takich jak nierównomierne oświetlenie, dodatkowe napisy, notatki, ręcznie rysowane kreski.

6. Wnioski na przyszłość

W dalszym etapie rozwoju aplikacji rozważyć można by następujące zagadnienia:
- wykrywanie „żeber” na zdjęciach – W rzeczywistych odlewach występują elementy ułożone prostopadle do płaszczyzny zdjęcia bloku. Utrudniają one analizę takich fotografii, ponieważ widoczne są jako bardzo mocno rozjaśnione obszary.
- wykrywanie zaburzeń obrazów – Zdjęcia, które były już analizowane „ręcznie” często zawierają dodatkowe notatki, linie które można by próbować wykrywać i pomijać w etapie analizy obrazu.
- obecnie uwydatnianie wad polega na obliczeniu różnicy pomiędzy obrazem mocno rozmytym (rozmycie o dużym promieniu) a oryginałem.
Metoda ta jest bardzo czuła na szumy dlatego można by spróbować ją zastąpić czymś w rodzaju lokalnej dyskretyzacji tzn.:
Dla każdego piksela należałoby obliczyć histogram jego otoczenia. Otoczenie to musi być większe od rozmiaru największej ciemnej plamy opisującej wadę. Na podstawie tak obliczonego histogramu można przeprowadzić dyskretyzację. W ten sposób dla każdego z punktów otrzymano by inne progi, czyli w obszarze przyciemnionym najjaśniejszy piksel który byłby dyskretyzowany do białego mógłby być znacznie ciemniejszy od najjaśniejszego piksela w innym rejonie. Kolejna cechą tego podejścia, byłoby to że duża dynamika jasności niektórych obszarów nie miałaby wpływu na obszary niewyraźne w których wady są słabo widoczne ponieważ tam punkty bieli i czerni byłyby znacznie bliżej siebie. Wadą tego podejścia w porównaniu do podejścia z rozmyciem jest jego wydajność, ponieważ podobnie jak przy filtrze rozmywającym dla każdego piksela trzeba przejrzeć jego sąsiadów, w przypadku lokalnej dyskretyzacji promień mógłby nawet być o połowę mniejszy, jednakże problemem jest to, że filtr rozmycia można zoptymalizować wykonując najpierw rozmycie pionowe, a następnie poziome, biorąc pod uwagę jak długo trwa obecnie rozmycie (i ile trwała wersja niezoptymalizowana) byłby to poważny problem.