Janne Koljonen
Vaasa yliopisto
2.3.2012
AUTO103 Signaalien käsittely
Harjoitus 5, tehtävä 6

Korrelaatio ImageJ:llä

Tehdään tällä kertaa Pearsonin korrelaatiokertoimella. Kahdeksanbittisessä harmaasävykuvassa kaikki arvot ovat välillä [0, 255] eli signaalin keskimääräinen taso on selvästi positiivinen. Pearsonin korrelaatiokertoimessa tason vaikutus eliminoidaan vähentämällä signaaleista keskiarvo eli lasketaan kovarianssin ja keskihajontojen tulon osamäärä:

$$\rho(j) = \frac{\sum_{n=0}^{N-1}\left[(x_1(n)-\bar{x}_1)(x_2(n)-\bar{x}_2)\right]}{\sqrt{\sum_{n=0}^{N-1}(x_1(n)-\bar{x}_1)^2 \cdot \sum_{n=0}^{N-1}(x_2(n)-\bar{x}_2)^2}},$$

jossa keskiarvo lasketaan:

$$\bar{x} = \frac{\sum_{n=0}^{N-1} x(n)}{N}.$$

Korrelaatiokertoimen laskenta voidaan laajentaa suoraviivaisesti useampiulotteisiin signaaleihin kuten 2-ulotteiseen kuvasignaaliin. Tehdään imageJ-pluginiin metodi, joka laskee kahden kuvan välisen korrelaation. Tämän jälkeen luodaan isommasta kuvasta pieniä osakuvia, joita verrataan korrelaatiolla mallikuvaan, jossa on viiden pikselin levyinen pystysuora valkoinen viiva mustalla taustalla. Esitetään korrelaatiot harmaasävykuvana.

ImageJ-pluginin koodi:

```
import ij.*;
import ij.process.*;
import ij.gui.*;
import java.awt.*;
import ij.plugin.filter.*;


// Värikanavien vaihto.
public class Matched_Filter_ implements PlugInFilter {
   ImagePlus imp;

   public int setup(String arg, ImagePlus imp) {
      this.imp = imp;
      return DOES_8G;
   }

   public void run(ImageProcessor ip) {
      // Kysytään kuvaoliolta kuvan leveyttä ja korkeutta
      int W=ip.getWidth();
      int H=ip.getHeight();

      // Apumuuttujat
      int[][] osakuva=new int[9][9]; // 9x9 pikselin kokoinen osakuva
      int[][] template=new int[9][9]; // 9x9 pikselin kokoinen sovitettu suodin eli mallikuva,
minkälaisia kuvioita korrelaatiolla haetaan.
      // Tehdään mallikuva, jossa 5 pikselin levyinen pystysuora valkoinen raita mustalla taustalla
      // Aluksi kaikki ovat 0, joten riittää lisätä valkoista.
      for (int y=0; y<9; y++)
      {
         for (int x=2; x<7; x++)
         {
            template[y][x]=255;
         }
```

```
      }
      // Toinen kuvaikkuna ja sen kuva tuloksille
      // Ks. http://rsbweb.nih.gov/ij/developer/api/ij/IJ.html
      // ja http://imagej.nih.gov/ij/developer/api/ij/ImagePlus.html
      ImagePlus kuva2=IJ.createImage("Korrelaatio", "8-bit", W, H, 1);
      kuva2.show();
      ImageProcessor ip2=kuva2.getProcessor();

      // Käydään jokainen pikseli reunoja lukuunottamatta läpi kahdella sisäkkäisellä for-silmukalla
      for(int y=4; y<H-4; y++)
      {
         for(int x=4; x<W-4; x++)
         {
            // Kunkin pikselin ympäriltä otetaan 9x9 pikselin näyte (osakuva)
            for (int dy=-4; dy<5; dy++)
            {
               for (int dx=-4; dx<5; dx++)
               {
                  osakuva[dy+4][dx+4] = ip.getPixel(x+dx, y+dy);
               }
            }

            // Lasketaan osakuvan ja templaten välinen korrelaatio
            double korrelaatio = corr(osakuva, template);

            // Skaalataan korrelaatio harmaasävyksi [0, 255] ja lisätään uuteen kuvaan
            ip2.putPixel(x,y, (int)(127*(korrelaatio+1)));
         }
      }
      // Päivitetään kuva näytölle
      kuva2.updateAndDraw();
   }

   // Kaksiulotteinen normalisoitu korrelaatio (Pearsonin korrelaatiokerroin)
   private double corr(int[][] kuva1, int[][] kuva2)
   {
      double keskiarvo1=0;
      double keskiarvo2=0;
      double kovarianssi=0;
      double varianssi1=0;
      double varianssi2=0;

      // Lasketaan keskiarvot signaaleista
      for (int y=0; y<kuva1.length; y++)
      {
         for (int x=0; x<kuva1[y].length; x++)
         {
            keskiarvo1 += kuva1[y][x];
            keskiarvo2 += kuva2[y][x];
         }

      }
      // Jaetaan pikseleiden lukumäärällä
      keskiarvo1 /= (kuva1.length*kuva1[0].length);
      keskiarvo2 /= (kuva1.length*kuva1[0].length);

      // Lasketaan kovarianssi ja varianssit
      for (int y=0; y<kuva1.length; y++)
      {
         for (int x=0; x<kuva1[y].length; x++)
         {
            kovarianssi += (kuva1[y][x]-keskiarvo1) * (kuva2[y][x]-keskiarvo2);
            varianssi1 += (kuva1[y][x]-keskiarvo1) * (kuva1[y][x]-keskiarvo1);
            varianssi2 += (kuva2[y][x]-keskiarvo2) * (kuva2[y][x]-keskiarvo2);

         }

      }

      // Lopuksi jaetaan kovarianssi keskihajontojen (eli varianssien neliöjuurien) tulolla
      return kovarianssi/Math.sqrt(varianssi1*varianssi2);
   }

}
```