Janne Koljonen
Vaasa yliopisto
15.3.2012
AUTO1030 Signaalien käsittely
Harjoitus 6, tehtävä 6

De-konvoluutio ImageJ:llä

De-konvoluution Java-koodi löytyy osoitteesta: http://rosettacode.org/wiki/Deconvolution/1D

Oetaan decony-metodi ja sovelletaan sitä ImageJ:llä kuvan dekonvoloimiseen. Asetetaan impulssivasteeksi yksinkertaisen keskiarvosuotimen kertoimet [0,2 0,2 0,2 0,2 0,2].

Eli nyt siis oletetaan, että kuva on sumentunut (blurred) keskiarvosuonta vastaavassa prosessissa. Dekonvoluutiolla selvitetään, mikä oli alkuperäinen kuva. Oletuksena on, että sumentumiseen ei liity kohinaa.

ImageJ-plugin.

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

// Yksiulotteinen dekonvoluutio kuvalle
public class Dekonvoluutio_ implements PlugInFilter
{
	ImagePlus imp;

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

	public void run(ImageProcessor ip)
	{
		// Kuvan leveys ja korkeus
		int W = ip.getWidth();
		int H = ip.getHeight();

		// Uusi kuva
		ImagePlus uusikuva = IJ.createImage("Dekonvoluoitu", "8-bit", W, H, 1);
		uusikuva.show();
		ImageProcessor ip2 = uusikuva.getProcessor();

		double[] impulssivaste = {0.2, 0.2, 0.2, 0.2, 0.2};

		// Käydään riveittäin läpi koko kuva.
		// Kukin rivi on alkuperäisen (tuntemattoman) kuvan ja impulssivasteen konvoluutio.
		// Lisätään rivin loppuun impulssivaste.length-1 nollaa.
		double[] osakuva = new double[W+impulssivaste.length-1];
		for (int y=0; y<H; y++)
		{
			// Kopioidaan rivi osakuva-muuttujaan (viimeiset jäävät nolliksi)
			for (int x=0; x<W; x++)
			{
				osakuva[x] = (double)ip.getPixel(x, y);
			}

			// Lasketaan dekonvoluutio.
			double[] dekonvoluoitu = deconv(impulssivaste, osakuva);

			// Tehdään uuteen kuvaan dekonvoluoitu rivi.
			for (int x=0; x<dekonvoluoitu.length; x++)
			{
				ip2.putPixel(x, y, Math.min(255, (int)dekonvoluoitu[x]));
			}
		}

		uusikuva.updateAndDraw();
	}
```

```
        public static double[] deconv(double[] f, double[] g)
        {
                double[] h = new double[g.length - f.length + 1];
                for (int n = 0; n < h.length; n++)
                {
                        h[n] = g[n];
                        int lower = Math.max(n - f.length + 1, 0);
                        for (int i = lower; i < n; i++)
                        {
                                h[n] -= h[i] * f[n-i];
                        }
                             h[n] /= f[0];
                        }
                return h;
            }
}
```

Testaaminen:

Dekonvoluoinnissa yllä käytetyllä algoritmilla on ongelmana, että kuvan reuna ei lähtökohtaisesti vastaa konvoluution tulosta ja reuna vaikuttaa koko dekonvoluutiotulokseen.

Esimerkki:

Tehtävässä 5 y oli:
```
double[] y = {4, 12, 28, 32, 28, 12, 0, -12, -28, -36, -36, -28, -12, -4, 0, 0, 0, 0, 0};
```

Poistetaan y:stä kolme ensimmäistä näytettä:
```
double[] y = {32, 28, 12, 0, -12, -28, -36, -36, -28, -12, -4, 0, 0, 0, 0, 0};
```

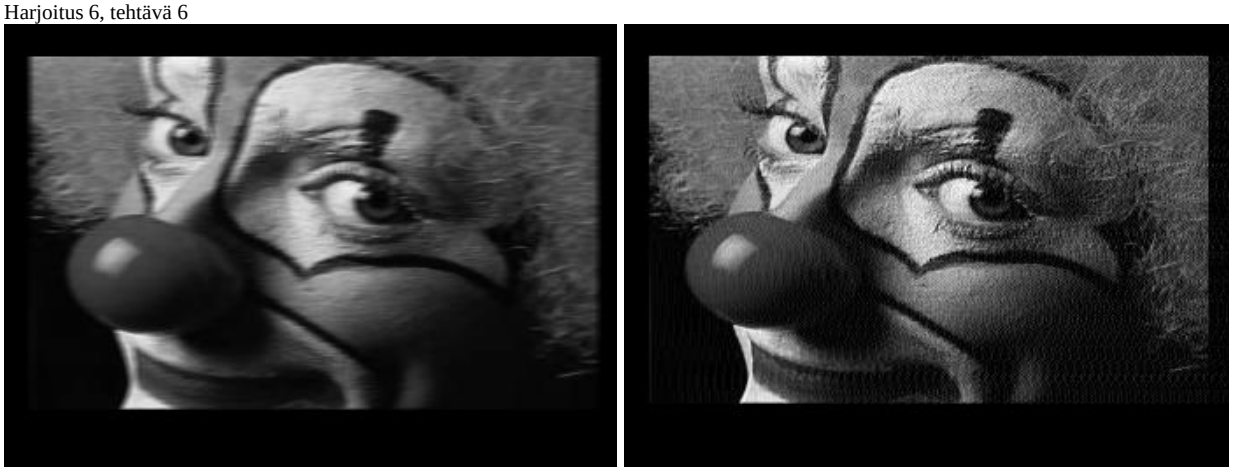
Verrataan dekonvoluutioita x1:n kanssa:
[1.0, 2.0, 4.0, 2.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[8.0, -1.0, -4.0, 5.0, -4.0, -8.0, 11.0]

Tulokset ovat aivan erit!

Testataan tämän takia dekonvoluutiota kuvilla, jotka on muodostettu oikeaoppimisesti konvoluutiolla:

1. Avataan kuva clown.jpg (File → Open Samples).
2. Muutetaan 8-bittiseksi.
3. Tehdään uusi kuva (File → New → Image), jonka pohjaväriksi valitaan Black, ja jonka koko on selvästi suurempi kuin pellekuvan.
4. Valitaan pellekuvasta haluttu alue hiirellä.
5. Kopioidaan valittu alue mustan kuvan keskelle.
6. Tehdään konvoluutio: Process → Filter → Convolve…: asetetaan kertoimiksi 1 1 1 1 1, joka normalisoituna vastaa dekonvoluution impulssivastetta.
7. Tehdään dekonvoluutio.

Janne Koljonen
Vaasa yliopisto
15.3.2012
AUTO1030 Signaalien käsittely
Harjoitus 6, tehtävä 6



Kuva. Konvolutoitu (vasen) ja dekonvoluoitu (oikea). Dekonvoluutio palautti kuvan terävyyden, mutta siinä näkyy pientä raidoitusta.