ROZ2 - Cv. 1 - Dekonvoluce

Adam Novozámský, novozamsky@utia.cas.cz
Ondřej Horáček, horacek@utia.cas.cz
Petra Bednaříková, bednarikova@utia.cas.cz

27. října 2011

Maska Gausiánu

1. naprogramujte generovaní masky 2D gausiánu
   - `G = gauss(N, sigma)`
2. funkce pro přidání bílého šumu o zadané SNR
   - `W = whiteNoise(I, SNR)`
3. funkce pro poškození obrázku rozmazáním a přidáním bílého šumu
   - `function D = demage(I, H, SNR)`
   - rozmažte obrázek a podívejte se na jeho spektrum
   - kombinujte rozostřením pohybem

Maska Gausiánu

1. naprogramujte generovaní masky 2D gausiánu
   - `G = gauss(N, sigma)`
2. funkce pro přidání bílého šumu o zadané SNR
   - `W = whiteNoise(I, SNR)`
3. funkce pro poškození obrázku rozmazáním a přidáním bílého šumu
   - `function D = demage(I, H, SNR)`
   - rozmažte obrázek a podívejte se na jeho spektrum
     - kruhem
     - pásmem
     - pohybem

Maska Gausiánu

1. naprogramujte generovaní masky 2D gausiánu
   - `G = gauss(N, sigma)`
2. funkce pro přidání bílého šumu o zadané SNR
   - `W = whiteNoise(I, SNR)`
3. funkce pro poškození obrázku rozmazáním a přidáním bílého šumu
   - `function D = demage(I, H, SNR)`
   - rozmažte obrázek a podívejte se na jeho spektrum
     - kruhem
     - pauzákem
     - pohybem

Maska Gausiánu

1. naprogramujte generovaní masky 2D gausiánu
   - `G = gauss(N, sigma)`
2. funkce pro přidání bílého šumu o zadané SNR
   - `W = whiteNoise(I, SNR)`
3. funkce pro poškození obrázku rozmazáním a přidáním bílého šumu
   - `function D = demage(I, H, SNR)`
   - rozmažte obrázek a podívejte se na jeho spektrum
     - kruhem
     - pohybem

Maska Gausiánu

1. naprogramujte generovaní masky 2D gausiánu
   - `G = gauss(N, sigma)`
2. funkce pro přidání bílého šumu o zadané SNR
   - `W = whiteNoise(I, SNR)`
3. funkce pro poškození obrázku rozmazáním a přidáním bílého šumu
   - `function D = demage(I, H, SNR)`
   - rozmažte obrázek a podívejte se na jeho spektrum
     - kruhem
     - gausiánem
     - pohybem

Maska Gausiánu

1. naprogramujte generovaní masky 2D gausiánu
   - `G = gauss(N, sigma)`
2. funkce pro přidání bílého šumu o zadané SNR
   - `W = whiteNoise(I, SNR)`
3. funkce pro poškození obrázku rozmazáním a přidáním bílého šumu
   - `function D = demage(I, H, SNR)`
   - rozmažte obrázek a podívejte se na jeho spektrum
     - kruhem
     - gausiánem
     - pohybem

Maska Gausiánu

1. naprogramujte generovaní masky 2D gausiánu
   - `G = gauss(N, sigma)`
2. funkce pro přidání bílého šumu o zadané SNR
   - `W = whiteNoise(I, SNR)`
3. funkce pro poškození obrázku rozmazáním a přidáním bílého šumu
   - `function D = demage(I, H, SNR)`
   - rozmažte obrázek a podívejte se na jeho spektrum
     - kruhem
     - gausiánem
     - pohybem

Maska Gausiánu

1. naprogramujte generovaní masky 2D gausiánu
   - `G = gauss(N, sigma)`
2. funkce pro přidání bílého šumu o zadané SNR
   - `W = whiteNoise(I, SNR)`
3. funkce pro poškození obrázku rozmazáním a přidáním bílého šumu
   - `function D = demage(I, H, SNR)`
   - rozmažte obrázek a podívejte se na jeho spektrum
     - kruhem
     - gausiánem
     - pohybem

Maska Gausiánu

1. naprogramujte generovaní masky 2D gausiánu
   - `G = gauss(N, sigma)`
2. funkce pro přidání bílého šumu o zadané SNR
   - `W = whiteNoise(I, SNR)`
3. funkce pro poškození obrázku rozmazáním a přidáním bílého šumu
   - `function D = demage(I, H, SNR)`
   - rozmažte obrázek a podívejte se na jeho spektrum
     - kruhem
     - gausiánem
     - pohybem

Maska Gausiánu

1. naprogramujte generovaní masky 2D gausiánu
   - `G = gauss(N, sigma)`
2. funkce pro přidání bílého šumu o zadané SNR
   - `W = whiteNoise(I, SNR)`
3. funkce pro poškození obrázku rozmazáním a přidáním bílého šumu
   - `function D = demage(I, H, SNR)`
   - rozmažte obrázek a podívejte se na jeho spektrum
     - kruhem
     - gausiánem
     - pohybem

Inverzní filtr

1. naprogramujte inverzní filtr a vyzkoušejte na rozmazaných a zašuměných obrázcích
   - `h = inverse(g, h)`
2. zjistěte, jak velký vliv má šum na jeho účinnost

Inverzní filtr

1. naprogramujte inverzní filtr a vyzkoušejte na rozmazaných a zašuměných obrázcích
   - `h = inverse(g, h)`
2. zjistěte, jak velký vliv má šum na jeho účinnost

Inverzní filtr

1. naprogramujte inverzní filtr a vyzkoušejte na rozmazaných a zašuměných obrázcích
   - `h = inverse(g, h)`
2. zjistěte, jak velký vliv má šum na jeho účinnost

Wienerův filtr

1. naprogramujte Wienerův filtr a vyzkoušejte na rozmazaných a zašuměných obrázcích
   - `h = wiener(g, h, konst)`
2. zjistěte, jak velký vliv má šum na jeho účinnost

Wienerův filtr

1. naprogramujte Wienerův filtr a vyzkoušejte na rozmazaných a zašuměných obrázcích
   - `h = wiener(g, h, konst)`
2. zjistěte, jak velký vliv má šum na jeho účinnost

Wienerův filtr

1. naprogramujte Wienerův filtr a vyzkoušejte na rozmazaných a zašuměných obrázcích
   - `h = wiener(g, h, konst)`
2. zjistěte, jak velký vliv má šum na jeho účinnost

Rozmazání

1. Obrázky v balíku `yiXX.pgm`
   - `i = 1,2,3 - typ poškození`
   - `XX ... SNR`
2. Prozkoumejte logiku těchto příkazů:
   - ```
     m1 = log(abs(fft2(f).^2));
     m2 = real(fft2(m1));
     mi = min(m2(:));
     m3 = m2 < 0.9*mi;
     ```
3. kód vyzkoušejte na jednotlivé obrázky, určete typ poškození a obrázky opravte
     ```

Rozmazání

1. Obrázky v balíku `yiXX.pgm`
   - `i = 1,2,3 - typ poškození`
   - `XX ... SNR`
2. Prozkoumejte logiku těchto příkazů:
   - ```
     m1 = log(abs(fft2(f).^2));
     m2 = real(fft2(m1));
     mi = min(m2(:));
     m3 = m2 < 0.9*mi;
     ```
3. kód vyzkoušejte na jednotlivé obrázky, určete typ poškození a obrázky opravte
     ```

Rozmazání

1. Obrázky v balíku `yiXX.pgm`
   - `i = 1,2,3 - typ poškození`
   - `XX ... SNR`
2. Prozkoumejte logiku těchto příkazů:
   - ```
     m1 = log(abs(fft2(f).^2));
     m2 = real(fft2(m1));
     mi = min(m2(:));
     m3 = m2 < 0.9*mi;
     ```
3. kód vyzkoušejte na jednotlivé obrázky, určete typ poškození a obrázky opravte
     ```

Rozmazání

1. Obrázky v balíku `yiXX.pgm`
   - `i = 1,2,3 - typ poškození`
   - `XX ... SNR`
2. Prozkoumejte logiku těchto příkazů:
   - ```
     m1 = log(abs(fft2(f).^2));
     m2 = real(fft2(m1));
     mi = min(m2(:));
     m3 = m2 < 0.9*mi;
     ```
3. kód vyzkoušejte na jednotlivé obrázky, určete typ poškození a obrázky opravte
     ```

Rozmazání

1. Obrázky v balíku `yiXX.pgm`
   - `i = 1,2,3 - typ poškození`
   - `XX ... SNR`
2. Prozkoumejte logiku těchto příkazů:
   - ```
     m1 = log(abs(fft2(f).^2));
     m2 = real(fft2(m1));
     mi = min(m2(:));
     m3 = m2 < 0.9*mi;
     ```
3. kód vyzkoušejte na jednotlivé obrázky, určete typ poškození a obrázky opravte
     ```

Rozmazání

1. Obrázky v balíku `yiXX.pgm`
   - `i = 1,2,3 - typ poškození`
   - `XX ... SNR`
2. Prozkoumejte logiku těchto příkazů:
   - ```
     m1 = log(abs(fft2(f).^2));
     m2 = real(fft2(m1));
     mi = min(m2(:));
     m3 = m2 < 0.9*mi;
     ```
3. kód vyzkoušejte na jednotlivé obrázky, určete typ poškození a obrázky opravte
     ```

ROZ2 - Cv. 1 - Dekonvoluce

Adam Novozámský, novozamsky@utia.cas.cz
Ondřej Horáček, horacek@utia.cas.cz
Petra Bednaříková, bednarikova@utia.cas.cz

27. října 2011