Uniwersytet Wrocławski | Institute Informatyki

Computational Photography
Winter 2015-16

LAB 3: DUE 9 DECEMBER 2015

Task 1: Deconvolution Gaussian Prior (40 pts)

Implement deconvolution with Gaussian Prior in the frequency domain. Ignore the artefacts at the boundaries. You have the paper here *http://groups.csail.mit.edu/graphics/CodedAperture/.*

You also have some details about implementation in here

http://groups.csail.mit.edu/graphics/CodedAperture/SparseDeconv-LevinEtAl07.pdf

You have to implement section 2. The formula 8.

If you have problems, or doubts, you can check their matlab implementation. Check the section 5 of the previous document.

http://groups.csail.mit.edu/graphics/CodedAperture/DeconvolutionCode.html

DATA: *http://groups.csail.mit.edu/graphics/CodedAperture/CodedAperture-Data-LevinEtAl.zip*
The filter files are *.mat. In python you can use **scipy.io**. In C++ **matio**. I only have checked the python one. Please read the README files. There are 7 filters per scale in the x axis.

Task 2: Depth Estimation (40)

Given the result of task 1 for all the scales, compute a depth map with the best result for each pixel. In the paper they use some weights (λ) learned from a set of images, which we don't have, and therefore ignore, so your depth might not be as good as theirs. You can check section 4 if you need more details.

The structure of the algorithm is the following.

- Deblur the entire image for each of scaled kernels.
- Compute reconstruction error $\boldsymbol{e_k}$ for each of the scaled images.

$$e_k = y - f_k \otimes x_k$$

- For small windows around every pixel, compute the error of that pixel,

$$\mathrm{E}_k \sim \sum_{j \in W_i} e_k(j)^2$$

- Select the minimum and create a depth map with the selected depths.

$$d(i) = \mathrm{argmin}_k \mathrm{E}_k(y(i))$$

Task 3: Image Reconstruction (20)

Simply take the values for $\boldsymbol{x_k}$ from the selected ***k = d(i)*** *to create a final deblurred image.*

Extra:

You do not need do do the regularisation part. But, if you want to play with graphcuts, and other MRF based optimization, there is some C++ implementation here

http://vision.middlebury.edu/MRF/code/

and here

http://vision.csd.uwo.ca/code/.

I have not tried the second.

Deliverables

Code and 1 final clock image, and depth map.

README file.

- How long did the assignment take?
- Issues and descriptions of your partial solution (for partial credit)
- Any extras?
- Collaboration acknowledgment.
- What was most unclear/difficult?
- What was most exciting?