Mathematisch- Naturwissenschaftliche
Fakultät

Fachbereich Informatik
Arbeitsbereich Visual Computing

Eberhard Karls
Universität
Tübingen

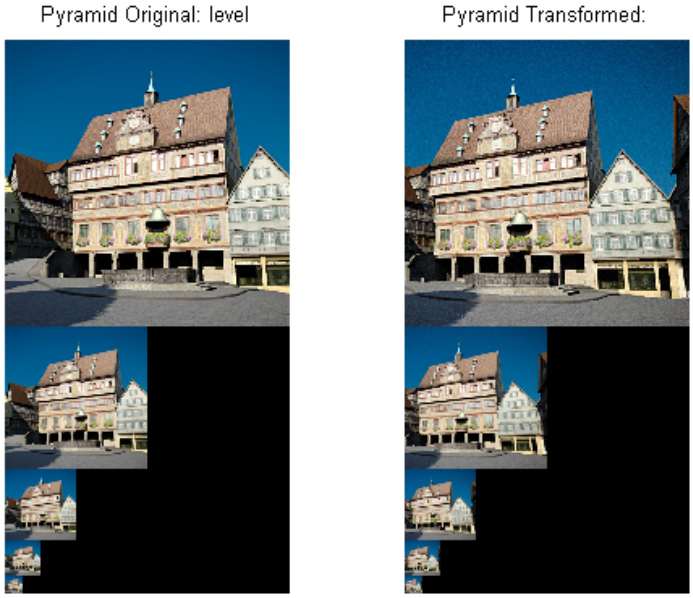
Bildverarbeitung I (Prof. Schilling)
WS2017/2018

Assignment 6, Due: January 25th, 2018

Exercise 13: Multiscale Image Alignment [5 points]

In this exercise you are going to align two images efficiently using a recursive multiscale approach to calculate the necessary transformation (in this case only translation).

a) Implement a function `my_gaussianPyramid` that generates a Gaussian pyramid for a given image and returns all of the pyramid levels in a cell array (e.g., `cell(1,levels)`) and the number of levels. Plot all of the levels using a function `my_plotImagePyramids`. The function should plot the pyramids of the original and transformed image next to each other (HINT: use one big image for each pyramid that includes the levels between `startlevel` and `endlevel`). You may use `imfilter` and `fspecial` for this exercise.

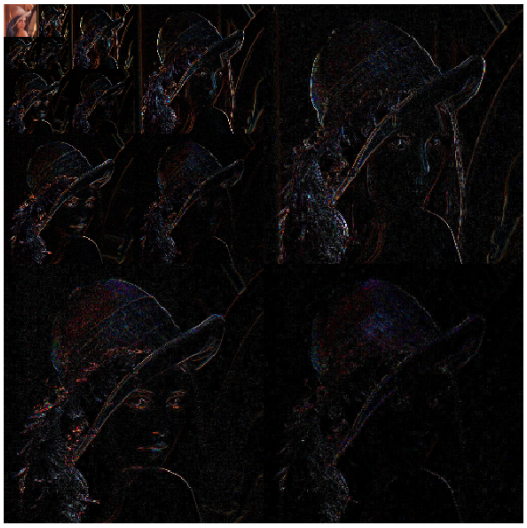


b) Implement a function `my_calculateImageTranslation` that recursively calculates the absolute translation between two images using their image pyramids. You should always start at the lowest level and use an appropriate error metric (e.g., sum of differences). Implement this metric in a function `my_imageDifference` that compares an image with another image translated by tx and ty and returns the normalized error.

c) Implement a function `my_blendResults` that overlays the two images (using the calculated transformation on one of the images) in order to validate your results visually.

Exercise 14: 2D Fast Wavelet Transformation [10 bonus points]

- Implement a function `my_fwt` that uses low-pass filtering, differencing and scaling to recursively calculate a discrete wavelet transformation for a given image. Use cell arrays to return the levels of each of the four subimages. Also return the number of levels. Plot your result as depicted in the figure below with a function `my_plotWavelets`. Use a larger image to include all of the subimages, again.



- Implement a function `my_ifwt` that calculates the inverse wavelet transformation. The function's first parameter `img_l_l` should be the horizontally and vertically low-pass filtered image at the starting level `startlevel`. The function's other parameters should be in line with the return values of your implementation of `my_fwt`.
- A simple way to denoise an image using wavelets is the so called soft thresholding (or shrinkage): the operator $D(U, \lambda) = sgn(U)max(0, \|U\| - \lambda)$ is applied to each of the high frequency subimages (`img_h_h`, `img_h_l`, `img_l_h`). Write a function `my_waveletDenoise` that implements this approach. Find appropriate values for λ. You might want to experiment with different values for λ for different levels to achieve the best possible results.