COMP6223 Computer Vision (MSc)

Coursework 1: Image Filtering and Hybrid Images

This coursework is only for students registered on the COMP6223 module

Brief

Due date: Thursday 8th November, 16:00.
Sample images: hybrid-images.zip
Handin: 1819/COMP6223/1/
Required files: report.pdf; code.zip
Credit: 15% of overall module mark

Overview

The goal of this assignment is to write a basic image convolution function and use it to create hybrid images using a simplified version of the SIGGRAPH 2006 paper by Oliva, Torralba, and Schyns. Hybrid images are static images that change in interpretation as a function of the viewing distance. The basic idea is that high frequency tends to dominate perception when it is available, but, at a distance, only the low frequency (smooth) part of the signal can be seen. By blending the high frequency portion of one image with the low-frequency portion of another, you get a hybrid image that leads to different interpretations at different distances. An example of a hybrid image is shown below.

Example hybrid image. Look at image from very close, then from far away.

Details

This project is intended to familiarise you with image filtering and the implementation of a convolution function in a language of your choice. Once you have created an image convolution function, it is relatively straightforward to construct hybrid images.

Template convolution. Template convolution is a fundamental image processing tool. Mark has covered convolution in detail in the lectures. See section 3.4.1 of Mark's book (Third Edition) ("Template Convolution") and the lecture materials for more information.

For this assignment, we want you to *hand-code* your own convolution operator function using a programming language or environment of your choice (Matlab, C, C++, etc). You should not make use of any built-in functions or libraries available to you for performing the convolution.

Your implementation must support arbitrary shaped kernels, as long as both dimensions are odd (e.g. 7x9 kernels but not 4x5 kernels). The border pixels should be set to 0. The implementation must also support convolution of both grey-scale and colour images. Note that colour convolution is achieved by applying the convolution operator to each of the colour bands separately (i.e. treating each band as an independent grey-level image).

Make sure that you implement the convolution operator and not a different (but similar) operator. Check that your implementation works correctly for non-symmetric kernels.

Hybrid Images. A hybrid image is the sum of a low-pass filtered version of the one image and a high-

pass filtered version of a second image. There is a free parameter, which can be tuned for each image pair, which controls *how much* high frequency to remove from the first image and how much low frequency to leave in the second image. This is called the "cutoff-frequency". In the paper it is suggested to use two cutoff-frequencies (one tuned for each image) and you are free to try that, as well.

Low pass filtering (removing all the high frequencies) can be achieved by convolving the image with a Gaussian filter. The cutoff-frequency is controlled by changing the standard deviation, sigma, of the Gaussian filter used in constructing the hybrid images. You will need to implement a function to generate a 2D Gaussian convolution kernel of `size*size` pixels. The `size` is width and height of the filter in pixels. It is standard practice for this to be set as a function of the sigma value as follows:

```
int size = (int) (8.0f * sigma + 1.0f); // (this implies the window is +/- 4 sigmas
from the centre of the Gaussian)
if (size % 2 == 0) size++; // size must be odd
```

High pass filtering (removing all the low frequencies) can be most easily achieved by subtracting a low-pass version of an image from itself.

We have provided you with 5 pairs of aligned images (in the hybrid-images.zip file) which can be merged reasonably well into hybrid images. The alignment is important because it affects the perceptual grouping (read the paper for details). We encourage you to create additional examples (e.g. change of expression, morph between different objects, change over time, etc.). See the hybrid images project page for some inspiration.

For the example shown at the top of the page, the two original images look like this:

The low-pass (blurred) and high-pass versions of these images look like this:

The high frequency image is actually zero-mean with negative values so it is visualised by adding 0.5 to every pixel in each colour channel. In the resulting visualisation, bright values are positive and dark values are negative.

Adding the high and low frequencies together gives you the image at the top of this page. If you're having trouble seeing the multiple interpretations of the image, a useful way to visualise the effect is by progressively down-sampling the hybrid image as is done below:

You should implement a function to create visualisations like the above to include in your report.

Restrictions

You must not use any built-in or library functions for implementing the convolution. For example, use of the following Matlab functions are forbidden: `imfilter()`, `filter2()`, `conv2()`, `nlfilter()`, `colfilt()`. If you're using OpenCV, then use of `filter2D` and other provided convolution functions is forbidden. The same applies for other libraries. You are allowed to use a Fourier transform operator, should you so wish (e.g Matlab's FFT, or (FFTW)[http://www.fftw.org/]).

The report

You need to prepare a short report (target length is ~2 sides of A4, although there won't be penalties for exceeding this). In the report you need to describe your convolution and hybrid images algorithms (in particular, please include your code for the convolution implementation) and any decisions you made to write your algorithms in a particular way. Then you should show and discuss the results of your algorithm, showing the results of your hybrid images algorithm (showing the image at a range of scales to show the effect) and show some of the intermediate images in the hybrid image pipeline (e.g. the low and high frequency images). Also, discuss anything extra you did. Feel free to add any other information you feel is relevant.

What to hand in

You need to submit to ECS Handin the following items:

- The report (as a PDF document)
- Your code enclosed in a zip file (include everything we would need to build and run it, including instructions)

Marking and feedback

Marks will be awarded for:

- Successful implementation of the convolution and hybrid images algorithms.
- Providing a good demonstration of your hybrid images algorithm.
- Effective use of your chosen implementation language.
- Well structured and commented code.
- Evidence of professionalism in implementation and reporting.
- Quality and contents of the report.

Standard ECS late submission penalties apply.

Individual feedback will be given covering the above points.

Useful links

- SIGGRAPH Hybrid Images Paper
- The Hybrid Images project page
- **Using Matlab:**
- Image processing toolbox tutorials
- **Libraries for C and C++ programmers**
- OpenCV
- VLFeat
- **Libraries for Java programmers**
- OpenIMAJ (note: this is what the COMP3204 students are using)
- BoofCV

Questions

If you have any problems/questions then email or speak to Jon in his office.