Laboratory 3:
Exemplar-based methods and applications

Artificial Vision, Spring 2025

Overview

Laboratory 3 covers two image processing methods, related by a common idea. The first part is a usual programming assignment, and the second one involves working with an on-line demo of one of the methods. The main goals of Laboratory 3 are to practice on:

- texture synthesis.
- exemplar-based image in-painting.

First of all, we must download the codes and supplementary materials from the link *Laboratory 3 Materials* on Aula Global of *Artificial Vision*.

FAQs:

- **What do I need to send?** A report with your work, including missing codes, an explanation where you discuss your work, as well as your results and conclusions. Extra codes and analysis are also welcome.
- **How?** Once your work is finished, you should upload it to the *Laboratory 3 Submission* on the Aula Global, in a single zip file. The submitted zip file has to contain your report and your source code (and all files needed to run it, such as images, other dependencies, etc). **Be clear and concise.** Your final score is not directly proportional to the number of pages in your report. This work should be done by groups of two people.
- **When?** Anytime until 13 May 2025 (08:29 UTC+2). After that, nothing will be received.

  Within 24 hours prior to the deadline, no comments will be answered.

References

- Slides on exemplar-based methods and applications of the course on *Artificial Vision*.
- [1] Original work of Efros and Leung on texture synthesis, available on Aula Global of *Artificial Vision*.
- [Inpainting] Description of the exemplar-based inpainting method, focused on numerical implementation. Available on Aula Global of *Artificial Vision*.
- On-line demo of the exemplar-based in-painting method. IPOL

Introduction

Laboratory 3 consists of two parts. In the first one, you will implement pieces of the code for texture synthesis, following the approach presented in the highly valued paper of Efros and Leung [1]. This work is usually acknowledged as the origin for the whole family of exemplar-based methods. In the second one, you will get familiar with IPOL Journal (Image Processing On-Line). You will use one of the on-line demos of IPOL for a hands-on study of the variational method for exemplar-based image in-painting.

1 Texture Synthesis by Non-parametric Sampling

In the *Laboratory 3 Materials* you will find MATLAB scripts, implementing the texture synthesis method proposed by [1]. Some parts of the code are missing, so your task is to finish the code and test the method. Folder *data* contains several examples of input texture samples, however, fill free to test the method using your own samples.

The code is organized as follows:

- **main.m**: script that runs the method and shows the results.
- **synthesize_texture.m**: function that takes a sample and creates a new texture of a desired size from that sample.
- **extract_patches.m**: function that extracts all possible square patches from a given sample. This part will be implemented by you.
- **compute_patch_distances.m**: function that computes distances between a given patch and all patches in a given list of patches. Again, this part will be implemented by you.
- **build_color_map.m**: function that creates a map where each pixel position is encoded by its own unique color.
- **gaussian_kernel_2d.m**: function that builds a 2D discrete isotropic Gaussian kernel.

Next, we present the tasks you have to perform [M - Mandatory, O - Optional]:

M Make sure you understand the code provided to you, then implement functions **extract_patches.m** and **compute_patch_distances.m**. Pay attention to the dimensions of *patch_list*.

Depending on your implementation of **compute_patch_distances.m** function, the method may take less or more time to finish. You may use smaller output texture size for debugging, for instance, [30 30].

M Test the complete method on a couple of different samples. Try to set different values of the patch size and the tolerance and explain the effect of it. Show examples of both success and failure cases.

You can use samples that are provided to you or any other samples of your choice. For testing the complete method, set the size of the output texture to [128 128] or bigger.

2 Exemplar-base Image In-painting

Navigate to the IPOL Journal web-page at www.ipol.im. IPOL contains articles about a variety of image processing and computer vision methods. Each of the articles is supplemented by the source code and the on-line demo, which you can use to test any method of interest by yourself. Articles are categorized into several topics. Open the "Inpainting" topic and then select the article called "Variational Framework for Non-Local In-painting". There are three tabs at the top of the article's page: article, demo and archive. Switch to the "demo" tab. Here you can choose one of the predefined examples or upload your own images and masks.

When you use your own images, there are two options, how to specify a region to be in-painted (*in-painting domain*). You can either upload an image together with a black and white mask (white color in the mask represents the in-painting domain). Or, you can upload an image alone and then draw the in-painting domain in the demo, using brushes and other tools. The first option is preferred for the session.

After selecting or uploading an image you get to the next page, where you can adjust parameters and run the method. The demo has nine parameters that can be adjusted. Only four main of them are shown by default and the rest are hidden under the "Advanced parameters" link.

Next, we present the tasks you have to perform [M - Mandatory, O - Optional]:

M Choose your own image (not one of the examples) and decide, which region of that image you would like to inpaint. Prepare a mask in any painting software (e.g., in GIMP) or even by hand. Use the demo to achieve a natural looking in-painting result. You might need to try different values of parameters. Show examples of both success and failure cases, and explain the results.

M Show the effect of changing the parameters of the method. Since there are many parameters that can be adjusted, focus primarily on *method*, *patch_size* and S (# scales). For this assignment you can use one of the pre-defined examples. How is the solution affected by the parameter tuning?

O Based on your observations in the previous point, formulate a set rules that should help to adjust parameters for different in-painting problems (different images, different in-painting domains, and so on).