In-painting with Savvy

Seijin Kobayashi, Frank Fu Lanke, Norman Juchler
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—A critical part of scientific discovery is the communication of research findings to peers or the general public. Mastery of the process of scientific communication improves the visibility and impact of research. While this guide is a necessary tool for learning how to write in a manner suitable for publication at a scientific venue, it is by no means sufficient, on its own, to make its reader an accomplished writer. We also describe the rules for submission in the computational intelligence laboratory. This guide should be a starting point for further development of writing skills.**

I. Introduction

Inpainting is an image processing technique used to infer the value of unknown pixels using the information apparent in the original image. One application for this method is image restoration where single pixels or larger scratches are missing, e.g. due to defective image acquisition or the age-driven deterioration of video footage. Another very common application is the removal of certain parts of an image, like watermarks or user-selected objects in the image. Figure 1 shows examples for input images that can be treated by inpainting methods. Throughout this paper we assume that it is known beforehand which pixels have to be reconstructed. The extraction of defective regions in the image is not discussed here.

Existing work. There exist many different techniques for inpainting, sometimes very specific to a certain application. For example noise reduction or recovering has been treated by various approaches using filter theory, Bayesian framework [TODO REFS]. Others [TODO REF] apply texture-synthesis approaches where the objective is to remove (potentially large) objects from an image by mimicking the surrounding region of the to-be-removed objected. Very popular in the image processing community is the exemplar-based method described by Criminisi et al., [REF CRIMINISI], an onion-peel approach where local gradients steer the fill-in procedure, which helps to preserve edges in the c. The correct reconstruction of the occluded region is of lesser importance in this approach and it impresses with a good overall visual perception.

Our approach. The method presented here is based on sparse coding of image patches. This technique can be compared with an interpolation scheme that takes into account . The results achieved with this method a

The novelty of our patch-based method consists of the combination of the following components:

- Intelligent use of neighboring patches to support the sparse encoding of a patch.
- Learning a dictionary
- Redundant reconstruction of missing pixels by overlapping patches and

Structure of the text. In section II some background information about the proposed methods is presented. Section IV presents our implementation which is then compared in the subsequent section **??**.

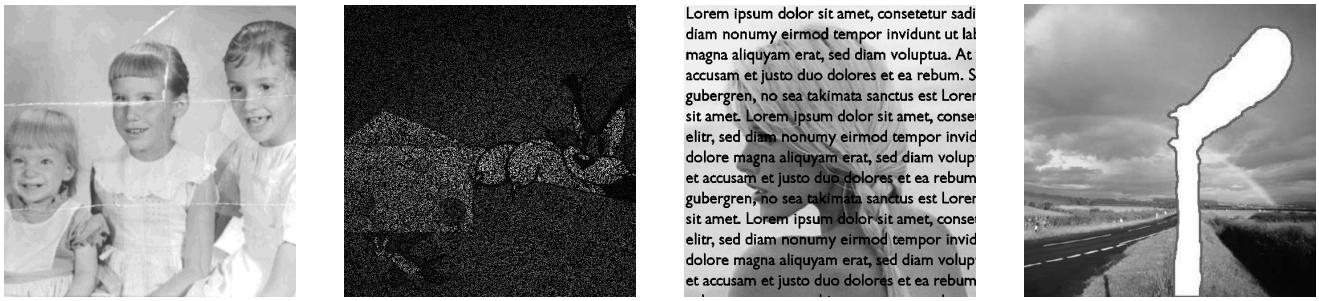
II. Methods (The Structure of a Paper)

In general the inpainting problem can be seen as a Maximum Likelihood Estimation (MLE) problem where the objective is to fill the missing pixels with most likely values, given the observed data. We represent the known values of the image through sparse encoding and during the process, infer pixel values at the locations of the unknown pixels, using the characteristics of the encoding basis. Popular bases include Discrete Fourier Transforms (DCT), Haar wavelets etc. which demonstrate favorable qualities for image encoding as they exhibit characteristics similar to generic image features. To achieve the sparse encoding we use the matching pursuit algorithm which serves to meet the following criterion (insert equation line 36 lecture 9).

Inpainting through sparse coding relies on infering unknown values based on the impact the known pixels incur on the chosen basis. Therefore, for a given genre of images it is advantageous to utilize a custom dictionary which can easily encode the given genre's typical image characteristics. For example in (Elad, Querre, Donoho), curvelets were demonstrated to be specifically efficient at encoding cartoon images.

A. Dictionary learning

Another direction in which we investigated is the construction of the dictionary we would use for our sparse coding. The nature of a dictionary not only determines the quality of the inpainting result, but also the speed of the inpainting process, depending on the size of the dictionary as well as the degree of sparseness reachable with it. The choice of dictionary was particularly of interest in our case to improve the running time. The approach we adopted for the reconstruction of the image would a priori assure a good quality of the reconstruction, however on the expense of the time necessary for the reconstruction, hence the need of a dictionary which would limit this trade-off without affecting



(a) Image restoration (b) Noise removal (c) Removal of watermarks (d) Object removal

Figure 1: Sample applications for inpainting. [TODO: REF OF IMAGE SOURCES]

too much the improved quality of the reconstruction. With this objective in mind, we built a under-complete dictionary on the space of patches of size 16x16, composed of 64 atoms (25 percent of the total dimension), by applying the approximate K-SVD algorithm on a training data customized for our need. We also paid particular attention to the initialization of the learning process, since this method is known to be initialization-sensitive [TODO REF]. In the rest of this section we will refer to the space of patches of size 16x16 as E.

K-SVD algorithm. The algorithm takes a training set of patches X, and learn a dictionary U, starting from an initialized value of U and iteratively learning to fit to X in a more efficient way. For all dictionaries, we used this algorithm with 15 iterations, using the same training data set.

Training Data set. The dictionary being under-complete, loss of information will inevitably occur. The atoms will hence need to be very efficient in reconstructing the image without losing important information. In other words, the atoms need to be able to describe characteristics of patches that have a high-variability among the kind of patches that we want to reconstruct. The aspects of patches with low-variability can be easily estimated by the mean, with a minimal loss of information. To achieve this, we first gathered patches of size 16x16 extracted from 36 pictures of size 512x512, consisting of photography of cities, nature and animals. We then applied principal component analysis (PCA) on these patches to deduce the most significant subspace of E of dimension 64, i.e. the subspace of E that include the highest variability of value among patches of real-world photography. We then projected the whole training set as well as our learning process onto that subspace. That way, we ensure that the learned 64 atoms stay in this a priori significant subspace, without getting lost into the less-meaningful directions of E, allowing them to efficiently reconstruct a patch.

Initialization. The algorithm we chose is sensitive to the choice of the initial candidate, it optimizes it locally and greedily until no progress is possible. We implemented 2 main different ways to initialize the learning process:

- From a known dictionary: in this method, we initialize the dictionary on a known dictionary, such as a 64-atom DCT (Discrete Cosine Transform) dictionary, or the 64 significant vectors obtained by applying PCA to the training set.
- k-means initialization: in this method, we initialize the dictionary based on the training data set. We first center and normalize the training data, after which we apply the K-means algorithm with K=64 based on the euclidean norm, and use the normalized 64 centroids as initializing atoms. To improve the robustness of the K-mean algorithm, we applied the algorithm 10 times and chose the result which had the smallest mean pairwise-distance with other clustering.

A randomly-generated atom initialization have also been implemented to serve as a reference for studying the efficiency of other initialization methods.

Lastly, to compare the efficiency of the overall dictionary performance, we set the baseline reference as the standard complete DCT dictionary.

TODO: Subsection “Background” to introduce roughly the concepts of sparse coding.

TOOD: Present our method.

TODO: Present baseline method and Criminisi method. (My suggestion is to compare the results of our algorithm with the class winner. They probably will like this approach. We probably won’t beat the performance of the Criminisi approach in general, but we might be better in some cases, furthermore we will be faster for sure...

Scientific papers usually begin with the description of the problem, justifying why the problem is interesting. Most importantly, it argues that the problem is still unsolved, or that the current solutions are unsatisfactory. This leads to the main gist of the paper, which is “the idea”. The authors then show evidence, using derivations or experiments, that the idea works. Since science does not occur in a vacuum, a proper comparison to the current state of the art is often part of the results. Following these ideas, papers usually have the following structure:

Abstract
: Short description of the whole paper, to help the reader decide whether to read it.

Introduction
: Describe your problem and state your contributions.

Models and Methods
: Describe your idea and how it was implemented to solve the problem. Survey the related work, giving credit where credit is due.

Results
: Show evidence to support your claims made in the introduction.

Discussion
: Discuss the strengths and weaknesses of your approach, based on the results. Point out the implications of your novel idea on the application concerned.

Summary
: Summarize your contributions in light of the new results.

III. Tips for Good Writing

The ideas for good writing have come from [1], [2], [3].

A. Getting Help

One should try to get a draft read by as many friendly people as possible. And remember to treat your test readers with respect. If they are unable to understand something in your paper, then it is highly likely that your reviewers will not understand it either. Therefore, do not be defensive about the criticisms you get, but use it as an opportunity to improve the paper. Before your submit your friends to the pain of reading your draft, please *use a spell checker*.

B. Abstract

The abstract should really be written last, along with the title of the paper. The four points that should be covered [2]:

1) State the problem.
2) Say why it is an interesting problem.
3) Say what your solution achieves.
4) Say what follows from your solution.

C. Figures and Tables

Use examples and illustrations to clarify ideas and results. For example, by comparing Figure 2 and Figure 3, we can see the two different situations where Fourier and wavelet basis perform well.

D. Models and Methods

The models and methods section should describe what was done to answer the research question, describe how it was done, justify the experimental design, and explain how the results were analyzed.

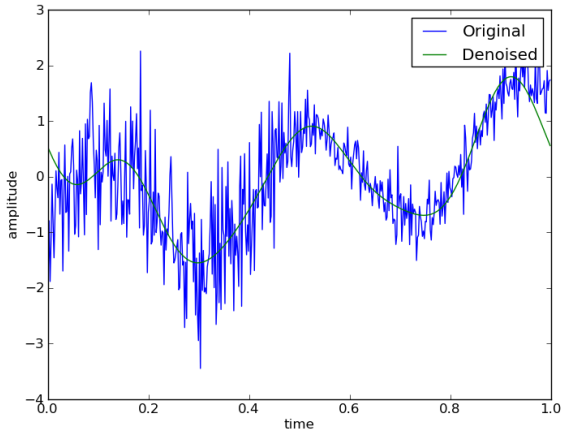


Figure 2: Signal compression and denoising using the Fourier basis.

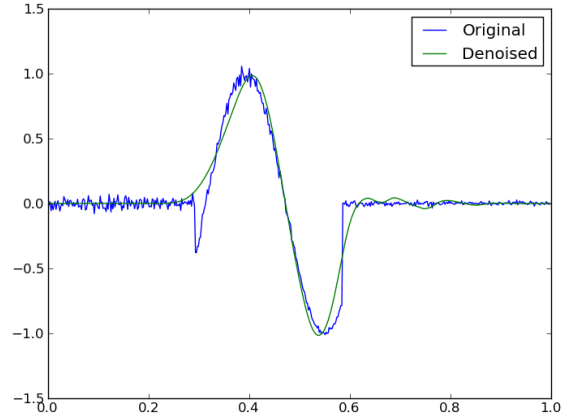


Figure 3: Signal compression and denoising using the Daubechies wavelet basis.

The model refers to the underlying mathematical model or structure which you use to describe your problem, or that your solution is based on. The methods on the other hand, are the algorithms used to solve the problem. In some cases, the suggested method directly solves the problem, without having it stated in terms of an underlying model. Generally though it is a better practice to have the model figured out and stated clearly, rather than presenting a method without specifying the model. In this case, the method can be more easily evaluated in the task of fitting the given data to the underlying model.

The methods part of this section, is not a step-by-step, directive, protocol as you might see in your lab manual, but detailed enough such that an interested reader can reproduce your work [3], [4].

The methods section of a research paper provides the information by which a study's validity is judged. Therefore, it requires a clear and precise description of how an experiment was done, and the rationale for why specific experimental procedures were chosen. It is usually helpful to structure the methods section by [5]:

1) Layout the model you used to describe the problem or the solution.
2) Describing the algorithms used in the study, briefly including details such as hyperparameter values (e.g. thresholds), and preprocessing steps (e.g. normalizing the data to have mean value of zero).
3) Explaining how the materials were prepared, for example the images used and their resolution.
4) Describing the research protocol, for example which examples were used for estimating the parameters (training) and which were used for computing performance.
5) Explaining how measurements were made and what calculations were performed. Do not reproduce the full source code in the paper, but explain the key steps.

E. Results

Organize the results section based on the sequence of table and figures you include. Prepare the tables and figures as soon as all the data are analyzed and arrange them in the sequence that best presents your findings in a logical way. A good strategy is to note, on a draft of each table or figure, the one or two key results you want to address in the text portion of the results. The information from the figures is summarized in Table I.

When reporting computational or measurement results, always report the mean (average value) along with a measure of variability (standard deviation(s) or standard error of the mean).

IV. Results (Tips for Good Software)

TODO: implementation details

TODO: details about benchmarking. How did we compare results? Which data sets did we use.

TODO: idea: use existign datasets for inpainting. Maybe we can found some in the web.

There is a lot of literature (for example [6] and [7]) on how to write software. It is not the intention of this section to replace software engineering courses. However, in the interests of reproducible research [8], there are a few guidelines to make your reader happy:

- Have a `README` file that (at least) describes what your software does, and which commands to run to obtain results. Also mention anything special that needs to be set up, such as toolboxes[1].
- A list of authors and contributors can be included in a file called `AUTHORS`, acknowledging any help that you may have obtained. For small projects, this information is often also included in the `README`.
- Use meaningful filenames, and not `temp1.m`, `temp2.m`. The code should also unzip into a sub-directory.
- Document your code. Each file should at least have a short description about its reason for existence. Non obvious steps in the code should be commented.
- Describe how the results presented in your paper can potentially be reproduced.

V. Discussion (Computational Intelligence Laboratory Requirements)

TODO: discuss the results. Why is our method worse/better than the other one etc.

TODO: outlook. What could be done in order to improve the method.

Your semester project is a group effort. It consists of four parts:

1) The programming assignments you solve during the semester.
2) Developing a novel solution for one of the assignments, e.g. by combining methods from previous programming assignments into a novel solution.
3) Comparing your novel solution to previous assignments.
4) Writing up your findings in a short scientific paper.

A. Developing a Novel Solution

As your final programming assignment, you develop a novel solution to one of the four application problems. You are free to exploit any idea you have, provided it is not identical to any other group submission or existing Matlab implementation of an algorithm on the internet[2].

Two examples for developing a novel solution:

- You implemented a collaborative filtering algorithm based on dimension reduction as part of an assignment. Now you apply dimension reduction to inpainting.
- You implemented both a clustering and a sparse coding algorithm for image compression. Now you combine both techniques into a novel compression method.

B. Comparison to Baselines

You compare your novel algorithm to *at least two baseline algorithms*. For the baselines, you can use the implementations you developed as part of the programming assignments.

[1]For those who are particularly interested, other common structures can be found at http://en.wikipedia.org/wiki/README and http://www.gnu.org/software/womb/gnits/.

[2]http://www.ethz.ch/students/semester/plagiarism_s_en.pdf

Basis	Support	Suitable signals	Unsuitable signals
Fourier	global	sine like	localized
wavelet	local	localized	sine like

Table I: Characteristics of Fourier and wavelet basis.

C. Write Up

The submission must be in PDF form, using the LaTeX template corresponding to the IEEE style of publication. Refer to Section V-D for more information about preparing your document. The document should be a maximum of **4 pages**.

D. LaTeX Primer

LaTeX is one of the most commonly used document preparation systems for scientific journals and conferences. It is based on the idea that authors should be able to focus on the content of what they are writing without being distracted by its visual presentation. The source of this file can be used as a starting point for how to use the different commands in LaTeX. We are using an IEEE style for this course.

1) Installation: There are various different packages available for processing LaTeX documents. On Windows, use the MikTeX package (http://miktex.org/), and on OSX use MacTeX (http://www.tug.org/mactex/2009/). Alternatively, on OSX, you can install the `tetex` package via Fink[3] or Macports[4].

2) Compiling LaTeX: Your directory should contain at least 4 files, in addition to image files. Images should be in `.png`, `.jpg` or `.pdf` format.

- IEEEtran.cls
- IEEEtran.bst
- groupXX-submission.tex
- groupXX-literature.bib

Note that you should replace groupXX with your chosen group name. Then, from the command line, type:

```
$ pdflatex groupXX-submission
$ bibtex groupXX-literature
$ pdflatex groupXX-submission
$ pdflatex groupXX-submission
```

This should give you a PDF document `groupXX-submission.pdf`.

3) Equations: There are three types of equations available: inline equations, for example $y = mx+c$, which appear in the text, unnumbered equations

$$y = mx + c,$$

which are presented on a line on its own, and numbered equations

$$y = mx + c \tag{1}$$

which you can refer to at a later point (Equation (1)).

[3]http://www.finkproject.org/

[4]http://www.macports.org/

4) Tables and Figures: Tables and figures are "floating" objects, which means that the text can flow around it. Note that `figure*` and `table*` cause the corresponding figure or table to span both columns.

E. Grading

There are two different types of grading criteria applied to your project, with the corresponding weights shown in brackets.

Competitive

The following criteria is scored based on your rank in comparison with the rest of the class.

- time taken for computation (10%)
- average rank for all other criteria relevant to the task, for example reconstruction error and sparsity (20%)

The ranks will then be converted on a linear scale into a grade between 4 and 6.

Non-competitive

The following criteria is scored based on an evaluation by the teaching assistants.

- quality of paper (30%)
- quality of implementation (20%)
- creativity of solution (20%)

F. Submission System

The deadline for submitting your project report is Friday, 22 June 2012. You need to submit:

- PDF of paper.
- Archive (`.tar.gz` or `.zip`) of software. Please do not forget to include author information in the source archive.

Important: Please check the submission instructions on the webpage as it is the most updated instructions.

VI. Summary

The aim of a scientific paper is to convey the idea or discovery of the researcher to the minds of the readers. The associated software package provides the relevant details, which are often only briefly explained in the paper, such that the research can be reproduced. To write good papers, identify your key idea, make your contributions explicit, and use examples and illustrations to describe the problems and solutions.

Appendix

Acknowledgements

The author thanks Christian Sigg for his careful reading and helpful suggestions.

SEIJIN'S SNIPPETS

A. Dictionary learning

Another direction in which we investigated is the construction of the dictionary we would use for our sparse coding. The nature of a dictionary not only determines the quality of the inpainting result, but also the speed of the inpainting process, depending on the size of the dictionary as well as the degree of sparseness reachable with it. The choice of dictionary was particularly of interest in our case to improve the running time. The approach we adopted for the reconstruction of the image would a priori assure a good quality of the reconstruction, however on the expense of the time necessary for the reconstruction, hence the need of a dictionary which would limit this trade-off without affecting too much the improved quality of the reconstruction. With this objective in mind, we built a under-complete dictionary on the space of patches of size 16x16, composed of 64 atoms (25 percent of the total dimension), by applying the approximate K-SVD algorithm on a training data customized for our need. We also paid particular attention to the initialization of the learning process, since this method is known to be initialization-sensitive [TODO REF]. In the rest of this section we will refer to the space of patches of size 16x16 as E. **K-SVD algorithm.** The algorithm takes a training set of patches X, and learn a dictionary U, starting from an initialized value of U and iteratively learning to fit to X in a more efficient way. For all dictionaries, we used this algorithm with 15 iterations, using the same training data set. **Training Data set.** The dictionary being under-complete, loss of information will inevitably occur. The atoms will hence need to be very efficient in reconstructing the image without losing important information. In other words, the atoms need to be able to describe characteristics of patches that have a high-variability among the kind of patches that we want to reconstruct. The aspects of patches with low-variability can be easily estimated by the mean, with a minimal loss of information. To achieve this, we first gathered patches of size 16x16 extracted from 36 pictures of size 512x512, consisting of photography of cities, nature and animals. We then applied principal component analysis (PCA) on these patches to deduce the most significant subspace of E of dimension 64, i.e. the subspace of E that include the highest variability of value among patches of real-world photography. We then projected the whole training set as well as our learning process onto that subspace. That way, we ensure that the learned 64 atoms stay in this a priori significant subspace, without getting lost into the less-meaningful directions of E, allowing them to efficiently reconstruct a patch. **Initialization.** The algorithm we chose is sensitive to the choice of the initial candidate, it optimizes it locally and greedily until no progress is possible. We implemented 2 main different ways to initialize the learning process:

- From a known dictionary: in this method, we initialize the dictionary on a known dictionary, such as a 64-atom DCT (Discrete Cosine Transform) dictionary, or the 64 significant vectors obtained by applying PCA to the training set.
- k-means initialization: in this method, we initialize the dictionary based on the training data set. We first center and normalize the training data, after which we apply the K-means algorithm with K=64 based on the euclidean norm, and use the normalized 64 centroids as initializing atoms. To improve the robustness of the K-mean algorithm, we applied the algorithm 10 times and chose the result which had the smallest mean pairwise-distance with other clustering.

A randomly-generated atom initialization have also been implemented to serve as a reference for studying the efficiency of other initialization methods.

Lastly, to compare the efficiency of the overall dictionary performance, we set the baseline reference as the standard complete DCT dictionary.

FRANKS'S SNIPPETS

Here goes Franks's text.

Formulation as a sparse coding problem:

In general the inpainting problem can be seen as a Maximum Likelihood Estimation (MLE) problem where the objective is to fill the missing pixels with most likely values, given the observed data. We represent the known values of the image through sparse enconding and during the process, infer pixel values at the locations of the unknown pixels, using the characteristics of the encoding basis. Popular bases include Discrete Fourier Transforms (DCT), Haar wavelets etc. which demonstrate favorable qualities for image encoding as they exhibit characteristics similar to generic image features. To achieve the sparse encoding we use the matching pursuit algorithm which serves to meet the following criterion (insert equation line 36 lecture 9).

Inpainting through sparse coding relies on infering unknown values based on the impact the known pixels incur on the chosen basis. Therefore, for a given genre of images it is advantageous to utilize a custom dictionary which can easily encode the given genre's typical image characteristics. For example in (Elad, Querre, Donoho), curvelets were demonstrated to be specifically efficient at encoding cartoon images.

NORMAN'S SNIPPETS

Here goes Norman's text.

REFERENCES

[1] Editorial, "Scientific writing 101," *Nature Structural & Molecular Biology*, vol. 17, p. 139, 2010.

[2] S. P. Jones, "How to write a great research paper," 2008, microsoft Research Cambridge.

[3] G. Anderson, "How to write a paper in scientific journal style and format," 2004, http://abacus.bates.edu/ ganderso/biology/resources/writing/HTWtoc.html.

[4] J. B. Buckheit and D. L. Donoho, "Wavelab and reproducible research," Stanford University, Tech. Rep., 2009.

[5] R. H. Kallet, "How to write the methods section of a research paper," *Respiratory Care*, vol. 49, no. 10, pp. 1229–1232, 2004.

[6] A. Hunt and D. Thomas, *The Pragmatic Programmer*. Addison Wesley, 1999.

[7] J. Spolsky, *Joel on Software: And on Diverse & Occasionally Related Matters That Will Prove of Interest etc..: And on Diverse and Occasionally Related Matters ... or Ill-Luck, Work with Them in Some Capacity*. APRESS, 2004.

[8] M. Schwab, M. Karrenbach, and J. Claerbout, "Making scientific computations reproducible," *Computing in Science and Engg.*, vol. 2, no. 6, pp. 61–67, 2000.