MATLAB-Mini-Project
Sampling Signals with Finite Rate of Innovation with an Application to Image Super-Resolution

Pier Luigi Dragotti, Jesse Berent, Loïc Baboulaz
Communications and Signal Processing Group
Department of Electrical and Electronic Engineering
Imperial College London

The coursework consists of a Matlab Miniproject. It counts 25% of the final mark. The data on which the numerical experiments will be performed is available on the course website. Students should produce a written final report. The report should contain, maybe in appendix, all the Matlab codes written during the project. We expect the code to be properly written and well documented.

The written final report must be submitted through the blackboard system by 5pm on Monday 27th of March 2017.

1 Preliminaries

The classic sampling theorem states that any bandlimited function $x(t)$ such that $X(\omega) = 0 \ \forall \ |\omega| > \omega_{max}$ can be exactly recovered from its samples given that the rate $2\pi/T$ is greater or equal to twice the highest frequency component. The continuous time signal is recovered with $x(t) = \sum_{n\in\mathbb{Z}} y[n]\text{sinc}(t/T - n)$ where $\text{sinc}(t) = \sin(\pi t)/\pi t$ and $y[n] = x(nT)$. A fun extension of this theorem has recently been developed, where it is made possible to sample signals that are not bandlimited but completely determined by a finite number of parameters. Such signals are called Finite Rate of Innovation (FRI) signals. In particular, it is made possible to sample and exactly recover streams of Diracs, differentiated Diracs and piecewise polynomials using a smoothing kernel $\varphi(t)$ that satisfies the Strang-Fix conditions. In this case, the samples are given by $y[n] = \langle x(t), \varphi(t-n)\rangle$. The main aim of this project is to make the student familiar with these recently developed sampling theorems. Notice that this project refers mainly to Chapter 5 of the lecture notes.

Throughout the project we assume that kernels of finite support L are such that $\varphi(t) \neq 0$ for $t \in [0, L]$. In order to simulate a continuous time signal in Matlab, we create a signal of length 2048 and use a sampling period

of $T = 64$. In this way, we have a signal ranging from $[0, 32[$ with a time resolution of $1/64$. This setup is valid for all the exercises.

2 Strang-Fix conditions

One of the main conditions on the sampling kernel $\varphi(t)$ is that it should be able to reproduce polynomials. Therefore, there exists coefficients $c_{m,n}$ such that

$$\sum_{n\in\mathbb{Z}} c_{m,n}\varphi(t-n) = t^m \quad m = 0, 1, \ldots, N. \tag{1}$$

In our context, we assume that $\{\varphi(t-n)\}_{n\in\mathbb{Z}}$ spans a Riesz space. Therefore there exists a dual-basis $\{\tilde{\varphi}(t-n)\}_{n\in\mathbb{Z}}$ that satisfies $\langle\varphi(t), \tilde{\varphi}(t-n)\rangle = \delta_0$. The coefficients $c_{m,n}$ can thus be found by applying the inner product with the dual-basis $\tilde{\varphi}(t-n)$ on both sides of equation (1). It results that $c_{m,n} = \langle t^m, \tilde{\varphi}(t-n)\rangle$. Notice that a scaling function that generates wavelets with $N+1$ vanishing moments is able to reproduce polynomials of degree N. It can therefore be used in our case as a sampling kernel.

Exercise 1 *The Daubechies scaling function. Which Daubechies scaling function should be used in order to reproduce polynomials of maximum degree 3? Compute the coefficients to reproduce polynomials of degree 0 to 3 for $n = 0, \ldots, 32 - L$ where L is the support of the kernel. Plot the results with the shifted kernels and the reproduced polynomials as in Figure 5.2 of the lecture notes.*

Guideline:

- *Choose the right Daubechies scaling function. For example, assume it is 'dB2'.*
- *To design the $\varphi(t)$ with a resolution $\frac{1}{2^J} = \frac{1}{64}$, do as follows:*

```
> phi = zeros(1,2048);
> [phi_T, psi_T, xval] = wavefun('dB2',6);
> phi(1:length(phi_T))=phi_T;
```

- *The function* `phi` *is in fact $\varphi(t)$. The shifted version $\varphi(t-T)$ is obtained by shifting* `phi` *by 64 points.*

Exercise 2 (optional) *The B-Spline scaling function. Which B-Spline scaling function should be used in order to reproduce polynomials of maximum degree 3? Find the dual-basis of the scaling function and compute the coefficients to reproduce polynomials of degree 0 to 3 for $n = 0, \ldots, 32 - L$ where L is the support of the kernel. Plot the results with the shifted kernels and the reproduced polynomials as in the previous exercise.*

Hint:

- *Use the Daubechies formula for spectral factorization in order to find the dual-basis.*

3 The annihilating filter method

Assume we observe a discrete signal $\tau[m]$ of the form $\tau[m] = \sum_{k=0}^{K-1} a_k t_k^m$ and assume we want to retrieve the locations t_k and the weights a_k. One efficient way of determining these parameters is through the annihilating filter. Consider the discrete filter $h[m]$ given by z-transform

$$H(z) = \prod_{k=0}^{K-1} (1 - t_k z^{-1}). \tag{2}$$

Clearly, the convolution between h and τ will result in a zero output as the t_ks are the roots of the filter (for more details, we refer to Section 5.2.1 of the lecture notes). Therefore h is called annihilating filter of τ. In practice, the filter can be found by posing $h[0] = 1$ and solving the following system of equations

$$\begin{bmatrix} \tau[K-1] & \tau[K-2] & \dots & \tau[0] \\ \tau[K] & \tau[K-1] & \dots & \tau[1] \\ \vdots & \vdots & \ddots & \vdots \\ \tau[N-1] & \tau[N-2] & \dots & \tau[N-K] \end{bmatrix} \begin{pmatrix} h[1] \\ h[2] \\ \vdots \\ h[K] \end{pmatrix} = - \begin{pmatrix} \tau[K] \\ \tau[K+1] \\ \vdots \\ \tau[N] \end{pmatrix}.$$

The knowledge of the filter is sufficient to determine the t_ks as they are the roots of the polynomial in (2). The weights a_k can be determined using K samples of $\tau[m]$. These form a classic Vandermonde system

$$\begin{bmatrix} 1 & 1 & \dots & 1 \\ t_0 & t_1 & \dots & t_{K-1} \\ \vdots & \vdots & \ddots & \vdots \\ t_0^{K-1} & t_1^{K-1} & \dots & t_{K-1}^{K-1} \end{bmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{K-1} \end{pmatrix} = \begin{pmatrix} \tau[0] \\ \tau[1] \\ \vdots \\ \tau[K-1] \end{pmatrix}$$

which has unique solution given that the t_ks are distinct.

Exercise 3 *Write a program that finds the annihilating filter $h[m]$ given a signal $\tau[m]$. Apply your program to the signal $\tau[m]$ available on the course website (m-file: tau.mat). Retrieve the locations t_k and the amplitudes a_k. Remark: we have set $K = 2$.*

4 Sampling Diracs

All the necessary pieces for implementing an end-to-end algorithm capable of recovering sampled streams of Diracs have been written. Indeed, consider a signal of type $x(t) = \sum_{k=0}^{K-1} a_k \delta(t - t_k)$. Then the locations t_k and the amplitudes a_k can be recovered from the samples by applying the annihilating

filter method to

$$\begin{aligned} s[m] &= \sum_n c_{m,n} y[n] \\ &= \langle x(t), \sum_n c_{m,n} \varphi(t-n) \rangle \\ &= \int_{-\infty}^{\infty} x(t) t^m dt \\ &= \sum_{k=0}^{K-1} a_k t_k^m \end{aligned}$$

where the last equality derives from the fact that $x(t)$ is a stream of Diracs and $\int_{-\infty}^{\infty} f(t)\delta(t-t_k)dt = f(t_k)$.

Exercise 4 *Create a stream of Diracs where $K = 2$. Sample the signal with an appropriate Daubechies scaling function using a sampling period of $T = 64$. Reconstruct the continuous time Diracs from the samples.*

Exercise 5 *We have sampled a stream of Diracs ($K = 2, T = 64$) with a 'db4' Daubechies scaling function. The observed samples are available on the course website on blackboard (m-file: samples.mat). Use the reconstruction algorithm to exactly recover the locations and the amplitudes of all the Diracs.*

5 Reconstruction in the presence of noise

The methods developed so far are not resilient to noise, so we want to use total-least squares and the Cadzow routine to make them more resilient to noise.

Exercise 6 *Using the routine you developed in Exercise 4, create a stream of Diracs where $K = 2$. Sample the signal with an appropriate Daubechies scaling function that can reproduce polynomial up to degree $N-1$ and $N > 2K$ using a sampling period of $T = 64$. Compute the moments $s[m] = \sum_n c_{m,n} y[n]$ with $m = 0, 1, .., N-1$ and $N > 2K$. Add zero-mean Gaussian noise to the moments $s[m]$ leading to $\hat{s}[m] = s[m] + \epsilon[m]$. Use the command* `randn` *to generate Gaussian noise. Choose different values for the variance σ^2.*

Exercise 7 *Given the noisy moments you have generated, develop the TLS and Cadzow routine described in the lecture notes in order to retrieve reliably the two Diracs that you have sampled. Compare the results with what you would obtain if you were not using the Cadzow routine. Try different values of N.*

6 Application: image super-resolution

You have just implemented a new sampling algorithm capable of perfectly recovering certain classes of signals that have been sampled with a rate below the Nyquist rate. This new scheme has a wide variety of applications ranging from wideband communications to image super-resolution. It is this last application that we are now asking you to implement.

Suppose that we have access to N low-resolution cameras that are all pointing to the same fixed scene. The positions of each camera is *unknown*. The goal of image super-resolution is to generate a single high-resolution image, called the super-resolved image, by fusing the set of low-resolution images acquired by these cameras. The super-resolved image (SR) has therefore more pixels than any of the low-resolution image (LR), but more importantly, it has also more details (*i.e.* it is not a simple interpolation). This is made possible because each camera observes the scene from a different viewpoint.

Image super-resolution can usually be decomposed into two main steps (see Figure 1):

1. Image registration: the geometric transformations between the LR images are estimated so that they can be overlaid accurately.

2. Image fusion and restoration: the data from each LR image is fused on a single high-resolution grid and missing data is estimated. Restoration finally enhances the resulting image by removing any blur and noise.

In the next exercise, you are asked to write in Matlab the image registration function of an image super-resolution algorithm. The approach used to register the LR images is based on the notion of continuous moments and is explained next. The fusion/restoration step is already written for you.

Suppose that the i-th camera observes the scene:

$$f_i(x,y) = f_1(x + dx_i, y + dy_i) \quad i = 1, \ldots, N,$$

where $f_1(x,y)$ is the scene observed by the first camera taken as the reference. We therefore have $(dx_1, dy_1)^T = (0,0)^T$. We have implicitly assumed here that the geometric transformations between the views are 2-D translations. At each camera, the point spread function $\varphi(x,y)$ of the lens is modeled by a 2-D cubic B-spline and blurs the observed scene $f_i(x,y)$ (see Figure 2). B-splines are functions with compact support and satisfy Strang-Fix conditions. Therefore, similarly to Equation (1), there exists a set of coefficients $\left\{c_{m,n}^{(p,q)}\right\}$ so that:

$$\sum_{m\in\mathbb{Z}}\sum_{n\in\mathbb{Z}} c_{m,n}^{(p,q)} \varphi(x-m, y-n) = x^p y^q,$$

The blurred scene $f_i(x,y) * \varphi(-x,-y)$ is then uniformly sampled by the CCD array which acts as an analog to discrete converter. The output discrete image of the i-th camera has then the samples $S_{m,n}^{(i)}$:

$$S_{m,n}^{(i)} = \langle f_i(x,y), \varphi(x-m, y-n) \rangle$$

By using a similar development as written in Section 4, it can be shown that:

$$\begin{aligned} m_{p,q} &= \int\int f(x,y)\, x^p y^q dx dy \quad (3) \\ &= \sum_m \sum_n c_{m,n}^{(p,q)} S_{m,n} \quad (4) \end{aligned}$$

Equation (3) is the definition of the geometric moments $m_{p,q}$ of order $(p+q)$, $p,q \in \mathbb{N}$, of a 2-D continuous function $f(x,y)$. Equation (4) shows that it is possible to retrieve the exact moments of the observed continuous scene from its blurred and sampled version by computing a linear combination with the proper coefficients. Knowing those moments, we can compute the barycenter of $f(x,y)$:

$$(\overline{x}, \overline{y})^T = \left(\frac{m_{1,0}}{m_{0,0}}, \frac{m_{0,1}}{m_{0,0}} \right)^T$$

If $f_i(x,y) = f_1(x + dx_i, y + dy_i)$, then $(dx_i, dy_i)^T$ can be retrieved from the difference between the barycenter of f_1 and the barycenter of f_i:

$$(dx_i, dy_i)^T = (\overline{x}_i, \overline{y}_i)^T - (\overline{x}_1, \overline{y}_1)^T$$

Exercise 8 *In this exercise, you have access to* $N = 40$ *low-resolution color images of* 64x64 *pixels. The goal is to generate a super-resolved color image of* 512x512 *pixels. The images observe the same scene and have been taken from different viewpoints. We assume that the geometric transformations between the views are 2-D translations. Write a program that registers each image with respect to the first image and which fits into the skeleton of the super-resolution algorithm provided.*

Guidelines and tips:

1. *Download and unzip the file* Tiger.zip. *You should have the following:*
   - Main_SR.m*: this file is the main program.*
   - ImageRegistration.m*: this is where you should write your code.*
   - ImageFusion.m*: this function fuses your LR images and takes as input the output of your registration function. You should* ***not*** *need to modify the code here.*
   - LR_Tiger_xx.tif*: this is the set of* 40 *LR images where* xx *goes from* 01 *to* 40*. Use* LR_Tiger_01.tif *as your image of reference for registration. Each LR image is a color image and has* 64x64 *pixels in each of the three layers of color (R,G,B).*
   - HR_Tiger_01.tif*: this* 512x512 *color image is your 'Holy Grail' and is only used to compute the PSNR of your SR image.*
   - PolynomialReproduction_coef.mat*: you will use this data to compute the continuous moments from the samples of your LR images. It contains three matrices of size* 64x64*:* Coef_0_0, Coef_1_0 *and* Coef_0_1*.* Coef_0_0 *is used to compute the continuous moment* $m_{0,0}$*,* Coef_1_0 *is for* $m_{1,0}$ *and* Coef_0_1 *is for* $m_{0,1}$*. They are loaded at the beginning of* ImageRegistration.m *so that you can directly use the matrices when you need to.*
   - 2DBspline.mat*: this data is the 2-D cubic B-spline used to model the point-spread function of the cameras. It is used during the restoration process and you don't need to use it.*

2. *Consider first the Red layer of each LR images, compute the continuous moments and find the 2-D translations. Then only, consider the Green layer, and then the Blue one. For each LR image, you should then have retrieved three similar 2-D translations.*

3. *The output of your registration function should be formatted the following way to fit the super-resolution algorithm:* Tx_RGB *and* Ty_RGB *are two* 40x3 *matrices. They correspond respectively to the horizontal and vertical shifts. The row of each matrix corresponds to the image number and the column specify the layer of color:* 1 =*Red,* 2 =*Green,* 3 =*Blue. By definition,* $\text{Tx\_RGB}(1,:) = \text{Ty\_RGB}(1,:) = (0 \quad 0 \quad 0)$*. Then* $\text{Tx\_RGB}(i,:) = (dx_i^{(R)} \quad dx_i^{(G)} \quad dx_i^{(B)})$ *and so on...*

4. *In order to reduce the noise created by the background of the image and get accurate moments, it is required that you select a threshold for each layer and set to 0 all the samples that are below it.*

5. *When reading an image in Matlab, samples values are integers between 0 and 255. Rescale them between 0 and 1.*

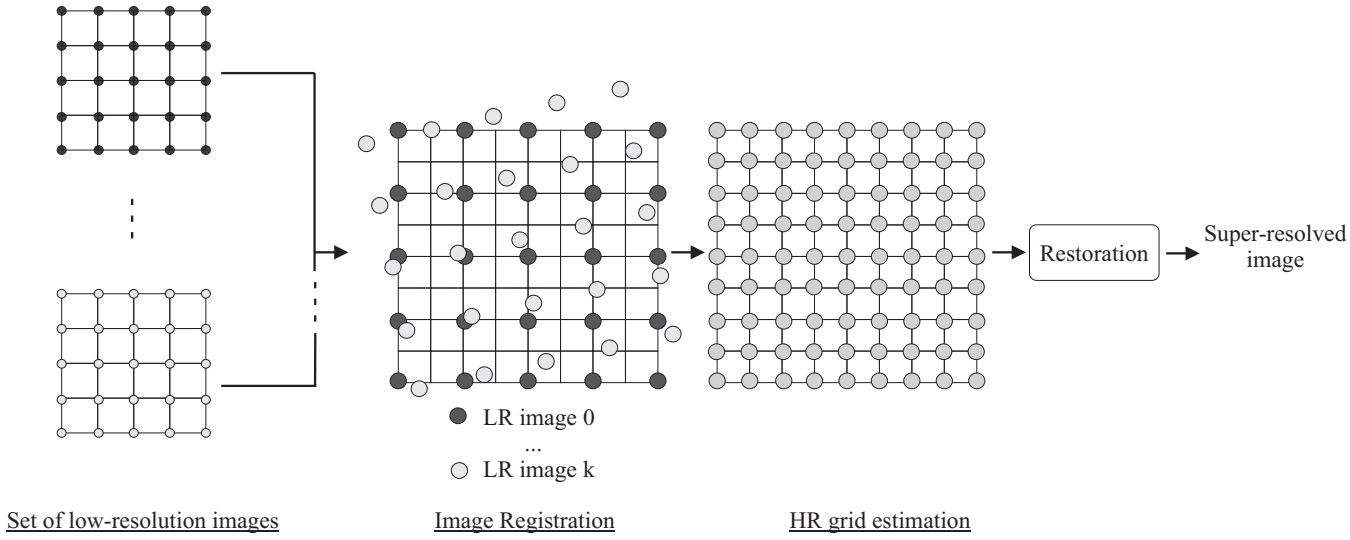
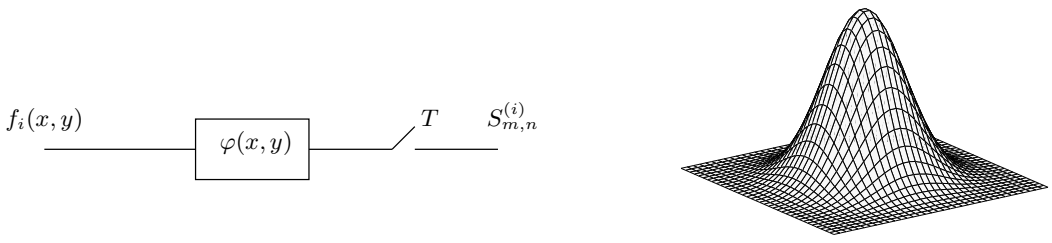


Figure 1: Image super-resolution principles



(a) Acquisition model.

(b) 2-D B-spline $\varphi(x,y)$.

Figure 2: Model for each of the N cameras.