Improved Resolution in Infrared Imaging Using Randomly Shifted Images

Graduation Report

Author: C. L. Luengo Hendriks
Mentor: Dr. Ir. L. J. van Vliet

Pattern Recognition Group
Laboratory of Applied Physics
Faculty of Applied Sciences
Delft University of Technology
Lorentzweg 1, 2628 CJ Delft
The Netherlands

Date: November 1998

Abstract

This report explores the possibility to reconstruct a properly sampled image from an undersampled image sequence. The proposed method aims at undoing the effects of aliasing and removing noise. The idea is to use multiple, randomly shifted, aliased frames, and merge them into a single, high-resolution image. This implies exchanging temporal resolution for spatial resolution.

The research was focused on images acquired with an infrared imager with a small fill factor (the detector elements are small compared to the pixel pitch). However, the results are applicable to images from many sources, and the algorithms were kept as general as possible.

The task has two parts: image registration, which is the estimation of shifts between two images, and image interpolation (or data fusion), the calculation of the high-resolution pixels.

We have implemented and tested some registration algorithms, which are capable of estimating shifts between two images with sub-pixel accuracy. We have also implemented and tested two interpolation algorithms that are capable of interpolating non-uniformly sampled, noisy data, as well as two other algorithms that can compute a high-resolution image from many low-resolution ones in a different manner. The registration algorithms, together with the interpolation algorithms, can successfully reconstruct a high-resolution image from multiple, randomly shifted, aliased frames. This report presents these methods, as well as their accuracy and computational cost, and some of the results obtained using these methods on real-world infrared images.

Outline

1. Introduction

The human visual system seems capable of combining information in a video sequence such that the spatial resolution appears to be much higher than that of a single frame. This is possible since each frame contains a slightly different view of the scene, and thus contains unique information. This report explores the possibilities to do this digitally. In short, we are looking for an algorithm that can translate a sequence of undersampled images like the two on the left of figure 1 into the one on the right.

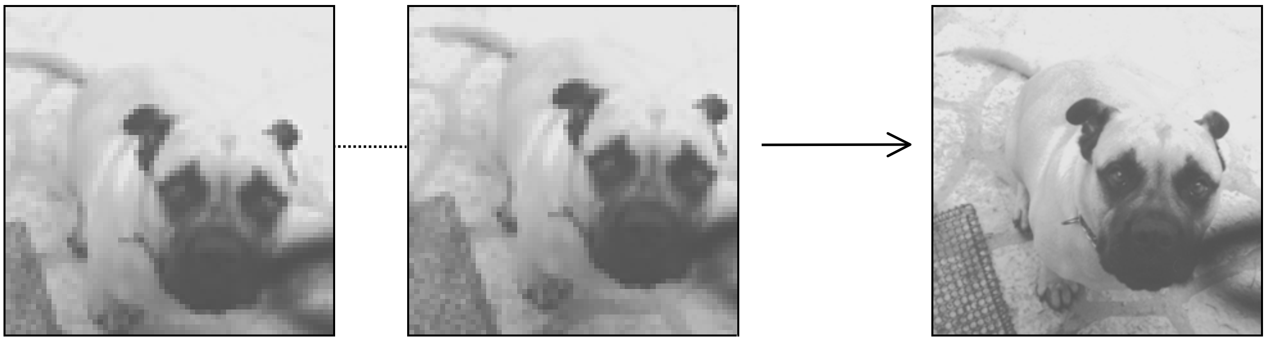


Figure 1: From a sequence of randomly shifted, undersampled images one can obtain a higher resolution image. Ideally, the algorithm should reveal the structure of the aliased object on the bottom left of the image.

Images acquired with an infrared imager are usually undersampled [5] due to a very low pixel density, which causes aliasing. A single frame contains only part of the information from the scene. We will argue that under certain circumstances, a sequence of these frames contains all the information required to reconstruct that scene. One of that requirements is that each image has a random, small (sub-pixel) displacement in relation to the next one. This can be accomplished by mounting the camera on a less than stable platform, or by inducing vibration on purpose. This sub-pixel displacement causes each frame to add new information to the sequence. Taking a couple of frames together, a single, higher-resolution image can be calculated. This calculation will reduce (or in some cases completely eliminate) the effects of aliasing.

The majority of the proposed techniques apply an iterative restoration scheme which produces good results, albeit at the expense of a huge, image-dependent computational load. An iterative algorithm is one in which the output is fed to the input again and again until a satisfactory result is produced. The aim is to construct an algorithm that can be used in real-time applications, and such an iterative algorithm does not fit in that constraint. The ultimate goal is to make a box that can be connected between an infrared camera and a monitor, and that improves the spatial resolution of the camera, in exchange of temporal resolution (less frames). This, of course, needs a very fast algorithm, preferably implemented in hardware, that produces an output image in the same amount of time it takes the camera to generate the input images. Our goal for this project is to find a suitable algorithm.

There are two parts to this restoration problem: first the shifts between the images must be estimated, and then the high resolution image must be calculated. The first part is called image registration, and the second part image interpolation.

Image registration is applicable to many areas of image processing, and is therefore amply discussed in the literature. The most popular method is block matching, extendible to hierarchical block matching to detect sub-pixel shifts. It is a simple method, in which the shift is found by comparing the two images with each possible shift and selecting the best match. It is, however, a brute-force approach that requires a lot of computations and almost no understanding of image processing. Two other, more elegant, approaches were also found, both applicable in real-time.

Image interpolation is the combination of the data in many frames. We could also call it image fusion. No satisfactory fusion algorithms were found in the literature. Some articles [1,4] propose interpolation using nearest neighbor interpolation; this is the most elementary interpolation method, and produces poor results. Most other articles discussing superresolution [5,6,12,14] describe iterative solutions. We tried some other interpolation methods, and compared them to the iterative solutions.

The general idea of superresolution has many more applications. For example, noise reduction and resolution improvement is useful in echography (ultrasonic imaging, used primarily for medical purposes, but also found in engineering applications like metal fatigue detection). A physician using such an apparatus moves the sensor around the area he is interested in; the motion provides him with a sequence of images that his brain can extract the relevant information from. However, a hard copy provides only a single frame, out of which it is difficult to make much sense.

The next chapter gives a formal definition of the images used, how they were acquired, some of their properties, and the constraints of the problem. Chapter 3 gives two solutions to the first part of the problem: estimating the shifts between the images (image registration). And the following chapter provides some solutions to the other half: using the images and the estimated shifts to find the high resolution image (image interpolation). Both chapters contain descriptions and results of some experiments we conducted to test the performance of these methods. Chapter 5 gives a short summary of the results in the previous two chapters to select an algorithm, and applies this algorithm to some real IR image sequences. And finally, the last chapter contains a summary of the conclusions found in this report.

A lot of questions have remained unanswered in the course of this research, and some new ones arose. This topic proved to be complex enough for a follow-up study. We compiled a list of ideas, presented at the end of this report, that we did not get to explore.

This project was done in cooperation with TNO-FEL. dr K. Schutte initiated this project; he and dr ir A.L.D. Beckers are our contacts at TNO-FEL.

I would like to thank dr ir L.J. van Vliet, prof. dr I.T. Young, dr K. Schutte, dr ir A.L.D. Beckers and ir G.M.P. van Kempen.

2. The Input Images

2.1. Infrared sensor

The infrared imager used in this research (CMT.FPA, manufactured by AIM) has a focal plane detector array that operates in the 7.7 to 10.1 micron wavelength region [15]. This array is made out of small, isolated, light-sensitive elements that do not cover the entire pixel (the fill factor is small, see figure 2).

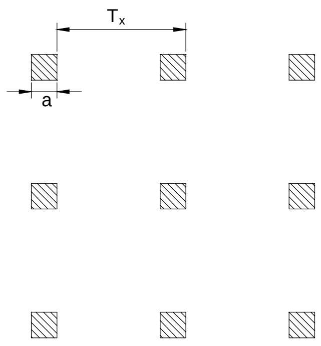


Figure 2: Small detector elements in focal plane array (example fill factor is $(1/4)^2$).

Large detector elements blur the image they acquire because they integrate the source image over the detector area [19]. This blurring is shown in figure 3 as a reduced response to the higher frequencies. In contrast, ideal point sampling produces a flat frequency response. Of course, the smaller the detector elements are, the flatter this response will be. We will assume the detector elements in this configuration small enough to regard it as a point-sampling system.

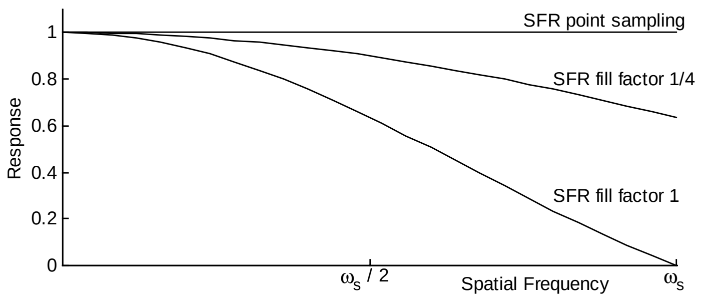


Figure 3: Spatial frequency response (SFR) of a detector array (ω_s is the sampling frequency). The ideal sampling is point sampling, which yields a constant response. An array that uses all the photons it receives has a fill factor of 1, and, as seen in this figure, has a reduced response to higher frequencies.

The camera optics act as a lowpass filter which limits the bandwidth of the acquired image [3]. For a correct sampling of the image in a camera, the detector array has to be sufficiently dense so it can sample this image (as projected on the detector plane) satisfying the Nyquist sampling criterion. If the detector array is not dense enough, the image will be corrupted by aliasing. This is indeed the case with the sensor we used for this project. It has 128 x 128 detector elements, which, in combination with the lens used, appear not to be nearly enough to provide a correctly sampled image. Thus we get severely aliased images.

2.2. Aliasing

Aliasing is always a tricky property to deal with. Any processing of aliased images must be done considering how and when the aliasing was introduced. In this case, we must remember that the aliasing was done *after* the image was shifted. This is important in both image registration and estimation.

Shifting an image produces a phase shift in the frequency domain [11]. Aliasing causes high frequencies to be folded over to lower frequencies (and its energy to be identified as energy of a lower frequency, hence the name); this folding is additive. It is easy to see that changing the phase of a couple of complex values and then adding them together gives a different result than adding them first and then changing the phase. For registration algorithms this is a problem. However, one algorithm uses this property to reconstruct the original scene using the fast Fourier transform.

2.3. Noise

Noise in an image is measured by its signal to noise ratio (SNR). It is defined by

$$SNR = 10 \log \frac{\mu^2}{\sigma^2} \quad , \tag{1}$$

when expressed in decibel (dB), where μ is the average signal strength or signal level, and σ^2 is the noise variance. There are four noise sources which exert an influence on these images [19]:

- Photon noise: unavoidable, caused by the quantum nature of light. The number of photons that arrive at a detector element is Poisson distributed, with expectancy equal to the intensity of the light, and variance equal to the expectancy. That means that photon noise is not independent of the signal, not additive, and not normally distributed.
- Thermal noise: noise produced by spontaneous creation of electron-hole pairs in the detector elements, due to thermal vibration. It is also Poisson distributed, and depends strongly on the temperature. Infrared detectors are usually cooled to very low temperatures (with liquid nitrogen, about 77 K) for proper functioning. As a side-effect, this reduces thermal noise.

- Readout noise: caused by the camera's electronics, this component is additive and normally distributed. It increases as the readout rate increases, but it is small in state-of-the-art components.
- Quantization noise: produced by the quantization of the pixel amplitudes into a finite number of discrete levels. As each pixel has a 14 bit gray value, this noise has a SNR of 95dB and can be ignored.

Photon noise is the only component that cannot be avoided, and is therefore usually the one that limits the SNR of a state-of-the-art camera. In CCD technology, the well capacity (the maximum number of electrons that can be stored in a detector element before readout) will determine the upper bound for the SNR. Infrared cameras, however, do not have comparable hardware, and SNR has no default definition for these machines.

Research at TNO-FEL [15] indicates a response of 88.1 mV/K in the 298 K to 303 K temperature region, and a variance of 3.641 mV. This gives a minimal sensitivity of 41 mK. This is the minimal temperature difference that can be measured in the given temperature range; it will be different for other temperatures. The minimal sensitivity is a way of expressing the SNR. The region of interest in the images used in chapter 5 have a temperature difference of is approximately 5°C. This means that the SNR in that region is about 42dB.

2.4. Calibration

There exists a difference in bias and gain among detector elements. This causes the images to look noisy, as each pixel behaves differently under the same light intensity. These characteristics also change in time, as they depend on the temperature of the detectors themselves. The simplest correction method needs two reference images, and does not allow for time-varying parameters. The two reference images are a dark image, *B(x,y)*, taken with the shutter closed, and a blank field image, *W(x,y)*, an evenly illuminated exposure. In the case of IR cameras, *B* is an image of a uniform cold plate, and *W* that of a uniform warm one. The corrected image is then calculated with

$$C(x,y) = K \cdot \frac{I(x,y) - B(x,y)}{W(x,y) - B(x,y)} \quad , \tag{2}$$

I(x,y) being the original input image and *K* a scaling factor. This also allows for temperature measurements if the temperature of the reference images are known.

The camera we used to record the test images has such a correction build-in.

2.5. Motion

The camera is usually placed on a stable platform, but the cooling equipment introduces motion by vibration. The motion can also come from the vibrations of the vehicle the camera might be stationed on, or they might be induced on purpose. We will use this motion in our resolution improving algorithm. We assume for now that all motion can be modeled as space-invariant translations in the image plane. This includes the assumption that all objects are at the same distance from the camera, and that the scene does not change during acquisition.

Let’s define the optical axis of the camera as *z* (see figure 4). The two other axis, both in the image plane, are *x* and *y* (*x* is horizontal, *y* is vertical). A rotation about *z* will rotate the acquired image, whereas a small rotation about *x* or *y* can be approximated by a shift in the image plane. Such a rotation will also deform the image, but we will show that this deformation can be neglected.

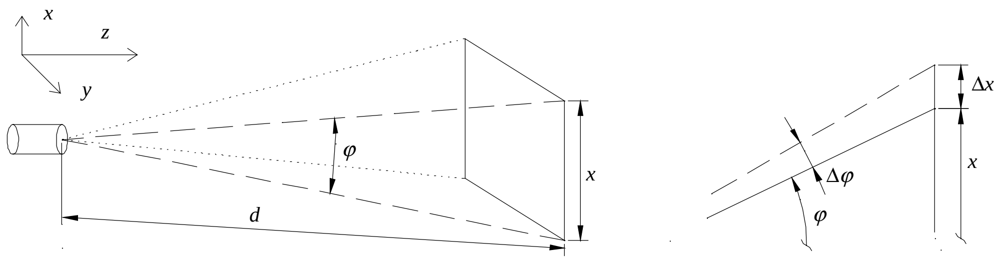


Figure 4, left: Field of view of the camera, and a definition of the axis used. Right: Change in the field of view as a result of a small camera rotation about the x or y axis.

The Δx from figure 4 (right) is the distance that the outer pixels are shifted under a camera rotation $\Delta\varphi$. This shift is minimally different from the shift of the center pixels, especially if the field of view is small. It is given by

$$\Delta x \;=\; d \cdot \tan\left(\tfrac{1}{2}\varphi + \Delta\varphi\right) - \tfrac{1}{2}x \;\approx\; d \cdot \tan\left(\Delta\varphi\right) \quad , \qquad (3)$$

taken that the opening angle φ of the optics is not too large, and $\Delta\varphi$ is very small (φ is defined as the angle, as seen by the camera, between the top and the bottom of the image; ½φ is the angular position of the edge relative to the optical axis, and defines the field of view). Figure 5 shows Δx as a function of the angular position in the image. We can see that the approximation of equation 3 holds for angles up to 15° (with an error of less than 10%), or for opening angles up to 30°. For such lenses, a small rotation about *x* and *y* can be approximated by a shift of the image.

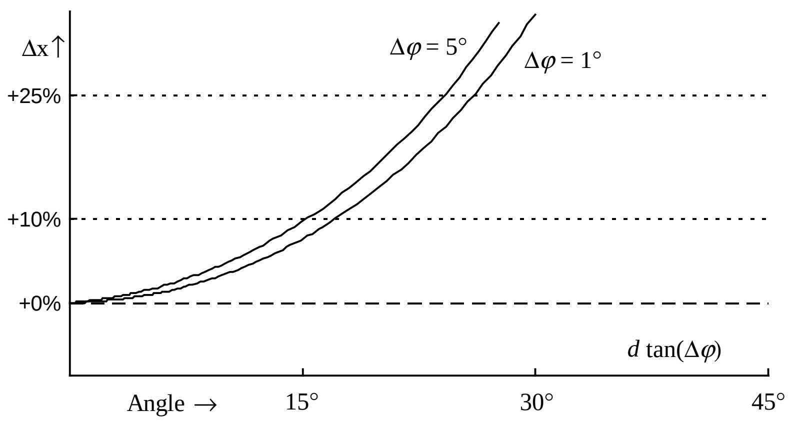


Figure 5: Relative pixel shift Δx under a camera rotation of Δφ. The angle is the distance to the center of the image. When the field of view is small, this shift is uniform across the image.

A rotation about the *z* axis causes each pixel to be moved in another direction over different distances. The maximum distance Δ*a* a pixel can be shifted by this rotation (see figure 6 for symbol definitions) is given by

$$\Delta a \;=\; \frac{x}{\sqrt{2}}\,2\sin\left(\tfrac{1}{2}\Delta\theta\right) \approx \frac{x}{\sqrt{2}}\,\Delta\theta \quad , \qquad (4)$$

if the rotation is small. To be able to ignore this rotation, the camera must rotate less than about 3′ 50″ over this axis (using $x = 128$ pixels and $\Delta a < 1/10$ of a pixel). That is, if this rotation is present, it almost certainly cannot be ignored.

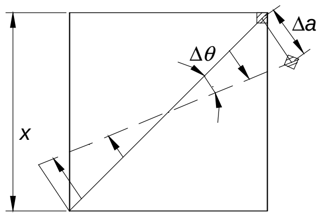


Figure 6: Maximum pixel displacement as a result of a camera rotation along the z axis.

2.6. Model for image acquisition

Besides the already mentioned finite detector array, the camera optics must also be taken into consideration to model the imaging system.

The object scene $o(x,y)$ (see figure 7) is convolved with the point spread function $psf(x,y)$ of the optics. The resulting image has a bandlimited frequency spectrum defined by this *psf*. It is projected on the detector plane with a shift (x_o,y_o). The detector elements, squares with sides *a*, cause an integration over their area expressed as $d(x,y)$. As stated before, the sensors are so small that they will be considered point sampling, making $d(x,y) = \delta(x,y)$.

The result is finally multiplied by a function *r*(*x*,*y*) that represents the limits of the detector array. Before discretization, the image *i*(*x*,*y*) can be written as

$$i(x,y) \;=\; \big(o(x,y) \,*\, psf\,(x,y) \,*\, d(x,y)\big) \cdot r(x,y) \quad . \tag{5}$$

Sampling and discretization is then accomplished with

$$i(m,n) \;=\; \mathscr{D}\big(i(m{\cdot}\Delta x \;,\, n{\cdot}\Delta y)\big) \quad . \tag{6}$$

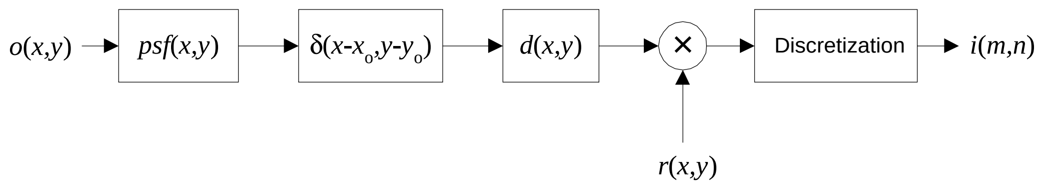


Figure 7: Model for image acquisition.

Remember that, as discussed earlier, the sampling frequency is too small in comparison with the cutoff frequency of the *psf*. This causes the image to be aliased, with high-frequency energy being folded over lower frequencies.

3. Image Registration

As mentioned in the introduction, we found two useful, general solutions to the shift estimation problem in the literature. We will discuss these, and some variations, here.

Image registration is normally used when any kind of transformations need to be detected. This includes rotation, scaling, stretching, shearing and warping, besides translation. In that case it is a *really* difficult task. At the beginning of this project we took as goal to reconstruct a high resolution image out of a sequence of *shifted* images. As other transformations make both the registration and the restoration more difficult, only translations will be considered. Rotation (rotation of the camera along the optical axis) and scaling (moving the camera closer or farther from the object) are interesting extensions to the problem, but stretching, shearing or warping should never appear in images from a rigid detector array. See section 2.5 on motion.

For a correct detection of shifts, the image must contain some features which make it possible to match two undersampled images. Very sharp edges and small details are most affected by aliasing, so they are not reliable to be used to estimate these shifts. Uniform areas are also useless, since they are translation-invariant. The best features are slopes (slow transitions between two grayvalues), or temperature ramps, as they are called in infrared imaging. These are unaffected by aliasing.

Some articles [4,12,14] describe block matching techniques to detect the (sub-pixel) shifts. We have not investigated this solution any further because of the high computational complexity and the inaccuracy. Block matching is based on comparing a block from one image (say, 10x10 pixels) with all possible blocks of the same size in the other image. The best match gives the estimate of integer-pixel shift. As best match one can use the mean absolute error or the root mean square error. A careful selection of the block is required, because it must contain image detail. Then both blocks are interpolated any number of times, and the process is repeated to obtain a fractional-pixel shift. Adding both estimates gives the sub-pixel displacement. This can be repeated a number of times for different blocks, using the average as final estimate.

3.1. Method 1: Cross-correlation

The best known method is cross-correlation, which has many variations, and can also be used to estimate rotations and scaling [18]. Here it is only discussed for estimation of shifts.

The cross-correlation function between two functions $x_1(t)$ and $x_2(t)$ is defined as [13]

$$\rho(s) = \frac{R_{12}(s)}{\left(R_{11}(0)R_{22}(0)\right)^{1/2}} \quad , \tag{7a}$$

$$R_{12}(s) \;=\; \mathrm{cov}\big(x_1(t),\, x_2(t+s)\big) \quad , \tag{7b}$$

for continuous, one-dimensional functions. The cross-correlation has a maximum at the s that makes $x_1(t)$ and $x_2(t+s)$ most similar. $R_{12}(s)$ is called the cross-covariance function, and is the same as $\rho(s)$, but not normalized. Note that (7b) can be written as

$$R_{12}(s) \;=\; \int_{-\infty}^{\infty} x_1^*(t)\; x_2(t+s)\;\mathrm{d}t \;+\; \mu_1\mu_2 \quad , \tag{7c}$$

which is just like a convolution of $x_1(t)$ with $x_2(-t)$. We can leave the constant term $\mu_1\mu_2$ out, because we are only interested in finding the maximum that will give the shift. The Fourier transform of (7c) is given by

$$\mathfrak{F}\big(R_{12}(t)\big) \;=\; X_1^*(\omega)\; X_2(\omega) \quad . \tag{8}$$

We will normalize with $|X_1(\omega)\, X_2(\omega)|$ in the Fourier domain instead of $\big(R_{11}(0)\, R_{22}(0)\big)^{1/2}$ in the space domain, even though it is not exactly the same. Later it will be apparent why. This leaves us with a different cross-correlation function $\rho'(t)$, but with equivalent properties. We will write

$$\rho'(t) \;=\; \mathfrak{F}^{-1}\left(\frac{X_1^*(\omega)\; X_2(\omega)}{\big|X_1(\omega)\; X_2(\omega)\big|}\right) \quad . \tag{9}$$

Let's define our two signals $x_1(t)$ and $x_2(t)$ equal in magnitude, but x_2 is shifted forward by an amount t_o, such that

$$x_2(t) \;=\; x_1(t-t_o) \quad . \tag{10}$$

Using the well-known shift property of the Fourier transform [11], the Fourier spectra are related by

$$X_2(\omega) \;=\; X_1(\omega)\; e^{-it_o\omega} \quad . \tag{11}$$

We can see that

$$\frac{X_1^*(\omega)\; X_2(\omega)}{\big|X_1(\omega)\; X_2(\omega)\big|} \;=\; \frac{X_1^*(\omega)\; X_1(\omega)\; e^{-it_o\omega}}{\big|X_1(\omega)\big|^2} \;=\; e^{-it_o\omega} \quad , \tag{12}$$

and, thus,

$$\rho'(t) \;=\; \mathfrak{F}^{-1}\big(e^{-it_o\omega}\big) \;=\; \delta(t-t_o) \quad . \tag{13}$$

Ideally, our modified cross-correlation is a shifted delta function (i.e. a single peak at location t_o). This, however, will rarely be the case, since the input signals will be noisy and severely aliased. It will, however, have a maximum located at t_o.

For two sampled images, the peak in $\rho'(t)$ will be only one pixel in size. Some neighboring pixels might share some of the energy of this peak, but the rest will be purely noise. The peak is easy to detect, but its position is only accurate on an integer-pixel basis. For higher accuracy, (12) needs to be transformed back to the space domain on a larger grid (say, 2 or 4 times as large in each dimension). Now the peak will be wider, and a center-of-gravity calculation will be able to detect its position with higher accuracy. (We will not discuss the two-dimensional variant, for the extension to two dimensions is trivial.)

The accuracy of this method might still not be enough, so let us look more closely at our cross-correlation in the frequency domain. (12) will have a magnitude of 1 everywhere, and a phase angle of $-ux_o-vy_o$ (for two-dimensional signals), where u and v are the spatial frequencies (in the range [-π, π]), and x_o and y_o are the shifts in pixel units. This results in an image (phase angle image) like the one in figure 8, A. It is possible to fit a plane, with the shifts as parameters, to this phase information. A least squares estimator will be the best linear unbiased estimator of x_o and y_o [2] (assuming the errors are independent). If we also assume that the noise is normally distributed, it will be the maximum likelihood estimator. In none of these assumptions turn out to be true, at least we have an unbiased linear estimator.

As the high frequencies will be dominated by noise (see figure 8, B), it is wise to use only the phase of the lower frequencies. If $|X_1(\omega)\, X_2(\omega)|$ in (12) is substituted for $|X_1(\omega)\, X_1(\omega)|$, which is expected to be the same, the magnitude will be expected to remain 1. This will not happen, however, due to noise and effects of aliasing. The magnitude can then be used as a confidence parameter for the corresponding phase pixel.

There is one more problem to overcome: the phase angle of a complex number is always in the range [-π, π]. When x_o or y_o are large enough, the angle will wrap over, even many times, for increasing u or v, as seen in figure 8 (C, D). This makes it impossible to estimate a shift. In this case, it is still possible to do an inverse Fourier transform of (12) to get a cross-correlation image, which will give us the integer-pixel shift. After one of the input images is corrected for this shift, we can start over again, with the certainty that the phase angle image is restricted to the range [-½π, ½π]

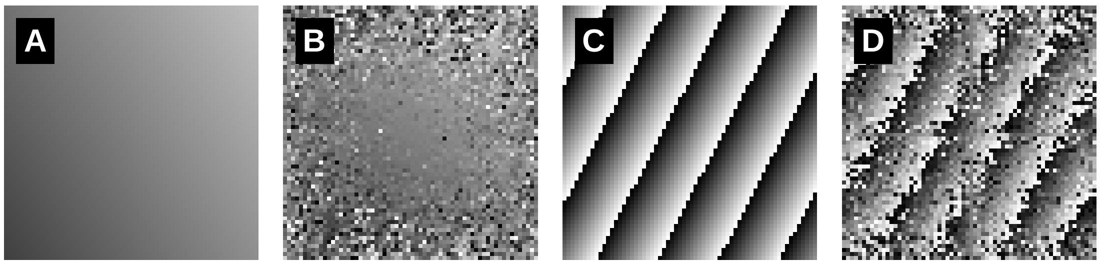


Figure 8, A: Ideal phase angle image for small shift. B: Real phase angle image for small shift; note how the higher frequencies are dominated by noise, whereas the lower ones seem useful to estimate a shift. C: Ideal phase angle image for larger shift; note the wrapping of the angle from -π (dark) to +π (light). D: Real phase angle image for larger shift.

3.2. Method 2: Gradient-based shift estimation

Another approach to shift estimation is the use of a first order Taylor series approximation of (10). It results in an equation that involves the gradient of one of the signals as well as the signals themselves [1,7].

As said, we start with the first order Taylor series expansion of (10),

$$x_2(t) = x_1(t) + t_\mathrm{o} \frac{\partial}{\partial t} x_1(t) + \varepsilon \quad , \tag{14}$$

with ε the error. Now all we have to do is minimize the sum of the squares of the errors (taking that x_1 and x_2 are sampled signals),

$$\mathrm{MSE} = \frac{1}{T} \sum_T \left(x_2(t) - x_1(t) - t_\mathrm{o} \frac{\partial}{\partial t} x_1(t) \right)^2 \quad , \tag{15}$$

which we can do by setting the partial derivative to t_o to zero. This yields

$$t_\mathrm{o} = \frac{\sum_T \left(x_2(t) - x_1(t) \right) \frac{\partial}{\partial t} x_1(t)}{\sum_T \left(\frac{\partial}{\partial t} x_1(t) \right)^2} \quad . \tag{16}$$

In two dimensions this leads to two equations which are correlated. We can solve them simultaneously by writing them as a matrix equation and inverting a 2-by-2 matrix. Equation (17) gives this equation, and (18) its solution. Here, we use h_o and v_o as shifts, and m and n are the sample indices. There are MN samples in each image.

$$\mathbf{M} \cdot \mathbf{S} = \mathbf{V} \tag{17a}$$

$$\mathbf{S} = \begin{bmatrix} h_\mathrm{o} \\ v_\mathrm{o} \end{bmatrix} \tag{17b}$$

$$\mathbf{V} = \begin{bmatrix} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left(x_2(m,n) - x_1(m,n) \right) \frac{\partial x_1(m,n)}{\partial m} \\ \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left(x_2(m,n) - x_1(m,n) \right) \frac{\partial x_1(m,n)}{\partial n} \end{bmatrix} \tag{17c}$$

$$\mathbf{M} = \begin{bmatrix} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left(\frac{\partial x_1(m,n)}{\partial m} \right)^2 & \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left(\frac{\partial x_1(m,n)}{\partial m} \frac{\partial x_1(m,n)}{\partial n} \right) \\ \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left(\frac{\partial x_1(m,n)}{\partial m} \frac{\partial x_1(m,n)}{\partial n} \right) & \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left(\frac{\partial x_1(m,n)}{\partial n} \right)^2 \end{bmatrix} \tag{17d}$$

$$\mathbf{S} = \mathbf{M}^{-1} \cdot \mathbf{V} \tag{18}$$

The question that remains is to find a suitable gradient operator. We tried many different ones, like Prewitt and Sobel [20]. It turns out that the selection of a gradient operator is very critical. The simplest one, and at the same time the one that resembles most closely the analytical derivative, is the difference of neighboring pixels. However, it introduces a phase shift (note that the difference of two neighboring pixels gives the gradient of the image in the point exactly halfway between those two pixels) that turned out fatal: the algorithm was only capable of accurately estimating positive shifts.

We chose the Gaussian gradient filter, which can be shown to give exact gradients on blurred images. It is exact in that the gradient is actually calculated analytically. The gradient of an image blurred with a Gaussian filter is written as

$$\frac{\partial}{\partial t}\left(x(t) * g(t)\right) = x(t) * \frac{\partial}{\partial t} g(t) \quad . \tag{19}$$

Here, $g(t)$ is the Gaussian used to blur the image $x(t)$; it is defined by

$$g(t) = \frac{1}{\sqrt{2\pi}\,\sigma} \cdot e^{\frac{-t}{2\sigma^2}} \quad . \tag{20}$$

The derivative of the Gaussian is known, and equal to

$$\frac{\partial}{\partial t} g(t) = \frac{t}{\sigma^2} \cdot g(t) \quad , \tag{21}$$

which is easy to sample into a discrete filter matrix. This also means that the input images $x_1(t)$ and $x_2(t)$ need to be blurred with a Gaussian so that (14) still holds. This equation then becomes

$$x_2(t) * g(t) = x_1(t) * g(t) + t_o\ x_1(t) * \left(\frac{t}{\sigma^2}\ g(t)\right) + \varepsilon \quad . \tag{22}$$

There is another positive point to the introduction of the blurring filter: as higher frequencies are more affected by the aliasing, removing them improves the estimate.

As we started off with Taylor, we need to keep in mind that the estimated shifts are only accurate if they are small enough. For larger shifts, we need to correct one of the images first. This can be done in two ways:

- Estimate the integer-pixel shift with the cross-correlation algorithm.
- Estimate repeatedly with this method, shifting one of the images, one pixel at a time, in the correct direction, until the estimated shift is small enough (smaller than one half of the pixel pitch).

Note that the equations used here are different if we swap the two images, unlike in the cross-correlation. Therefore, the estimated shift between two images can be different when using one or the other as reference frame.

3.3. Increasing accuracy

To further improve the shift estimation algorithms, we can estimate the shift between each pair of images in a sequence (giving $p(p$-1) shift estimates, where p is the number of frames), and combine these estimates in such a way as to cancel errors.

For example, in a sequence of 4 images, we would normally take the first one as reference (shift is 0) and estimate the shift of the other three in relation to that one. Instead of that, we could also estimate the relative shifts between each of the 12 possible pairs. For each frame we will then have 3 shift estimates, which we can average.

A good way of combining these estimates is to calculate the center of gravity of the estimates relative to each frame. The center of gravity of the p estimates calculated with frame 1 as reference (including the estimate for frame 1 itself, which is 0) will be the new origin for those estimates. The same applies to the other estimates. Now we can assume that each of the p^2 shifts has the same origin, and the p estimates for each frame can be averaged.

3.4. Registration experiments

One simple experiment has been done on a sequence of images, acquired by sub-sampling a correctly sampled image with random offsets. To accomplish offsets different from (0,0) or (0.25,0.75) and the like, we used a bicubic interpolator to sample the image at arbitrary locations. Bicubic interpolation is accurate enough for our purpose [10].

As the registration accuracy is highly dependent on image content, as well as aliasing, we did experiments with two images (size 256^2) and two sub-sampling factors. That gives a total of four experiments, which allows for a good comparison of the proposed algorithms. Figure 9 shows the two images we used: “trui” (left) and “lena” (right). We used sub-sampling factors of 2 and 4, producing images of 128^2 and 64^2 pixels.

Each experiment used 50 different, low resolution images. The displacements of each of these images was computed in relation to 10 different reference images, giving a total of 490 shift estimates. As we know the shift with which they were acquired, we can compare the estimates to the real shifts. This gives us a mean error and a standard deviation.

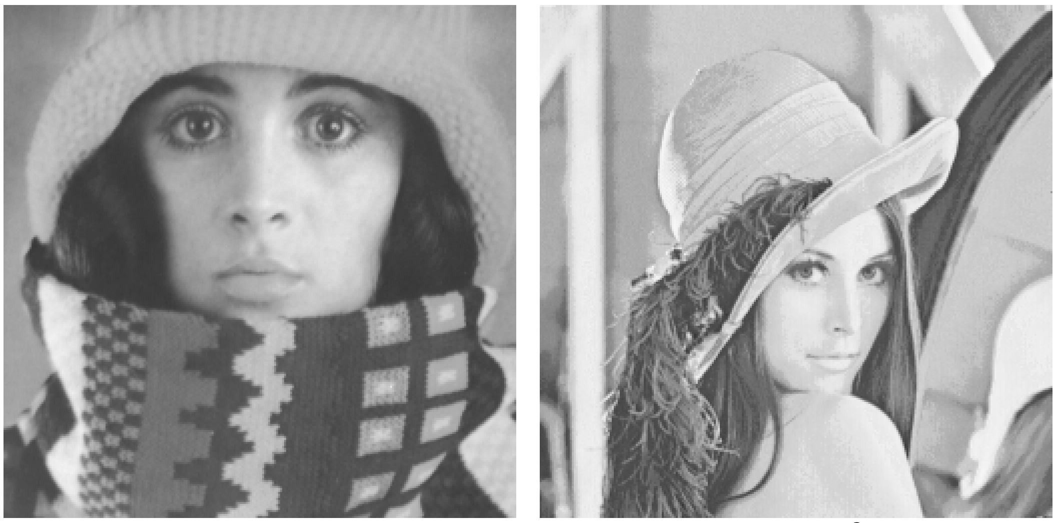


Figure 9: Original images “trui” (left) and “lena” (right). Both have 256^2 pixels.

3.4.1. Implementation

A total of 6 algorithms were implemented, including four variations of the cross-correlation method. A short description follows.

- FFTS: Uses the two-step cross-correlation method as discussed in section 3.1. The modified cross-correlation is calculated by the inverse fast Fourier transform of equation

(12), and the second image is shifted over the integer number of pixels calculated, so the new shift is expected to be between -0.5 and 0.5 in both directions. The portion of the image that falls outside of the overlapping region is thrown away, and the new part required is taken from the first image. We will assume that this has little influence because the shift will not be very large (if the shift is too large, the images are useless to reconstruct a high-resolution image anyway). The reason for not clipping the images is that the fast Fourier transform (FFT) needs square images, with 2^n as the length of the sides, for optimal performance. Then the FFT of the translated image is re-calculated and the phase angle image fitted to estimate the sub-pixel shifts.

- FFTSR: Same as FFTS, but the sub-pixel shifts are calculated twice, the second time after removing outlayers.
- FFTQ-2: The modified cross-correlation is calculated, but on a grid twice as large in both directions. This is accomplished by padding with zeros in the Fourier domain.
- FFTQ-4: Same as previous, but on a grid four times as large in both directions.
- GRS: Gradient-based shift estimation. Uses the repetitive method of handling large shifts. The Gaussian used to blur the input frames has a variance of one pixel.
- GRS-M: Calculates the shifts relative to each of the frames, using GRS, and averages these as discussed in section 3.3. This algorithm was tested in an alternate manner, to keep the results comparable.

3.4.2. Results

As expected, not all images can be registered with the same accuracy. “trui” has some slower slopes, which make the estimation more accurate. Also, each direction has different edges; for both images it was easier to estimate the horizontal than the vertical shift. See the results in table 1 and table 2.

	“trui”, sub-sampled by a factor 4				“lena”, sub-sampled by a factor 4			
	x		**y**		**x**		**y**	
	mean	**st. dev.**	**mean**	**st. dev.**	**mean**	**st. dev.**	**mean**	**st. dev.**
GRS	0.0079	0.0075	0.0123	0.0106	0.0176	0.0131	0.0273	0.0239
GRS-M	0.0068	0.0053	0.0100	0.0082	0.0110	0.0090	0.0252	0.0206
FFTS	0.0188	0.0181	0.0199	0.0163	0.0367	0.0317	0.0531	0.0452
FFTSR	0.0146	0.0134	0.0162	0.0129	0.0351	0.0307	0.0481	0.0409
FFTSQ-2	0.1191	0.0869	0.1105	0.0767	0.1620	0.1084	0.1110	0.0854
FFTSQ-4	0.1016	0.0630	0.0964	0.0565	0.1434	0.0995	0.0991	0.0753

Table 1: Errors in the estimated shift for images sub-sampled by a factor four. Means and standard deviations are over 490 realizations, except for the GRS-M algorithm.

	"trui", sub-sampled by a factor 2				"lena", sub-sampled by a factor 2			
	x		**y**		**x**		**y**	
	mean	**st. dev.**	**mean**	**st. dev.**	**mean**	**st. dev.**	**mean**	**st. dev.**
GRS	0.0037	0.0035	0.0043	0.0040	0.0061	0.0048	0.0095	0.0086
FFTS	0.0090	0.0060	0.0126	0.0075	0.0212	0.0172	0.0290	0.0219
FFTSR	0.0085	0.0056	0.0116	0.0068	0.0171	0.0143	0.0262	0.0177
FFTSQ-2	0.0976	0.0778	0.1031	0.0769	0.1152	0.0872	0.1126	0.0905
FFTSQ-4	0.0820	0.0498	0.0835	0.0509	0.0986	0.0628	0.0992	0.0714

Table 2: Errors in the estimated shift for images sub-sampled by a factor two. Means and standard deviations are over 490 realizations.

Now we can see why it is worth to estimate the shifts from the phase angle image when using the cross-correlation method. It is also shown that removing outlayers indeed improves the estimate of FFTS. Furthermore, the GRS algorithm is the most accurate one in all cases. In section 4.7 we present an experiment that indicated that one tenth of a pixel is accurate enough if we want to improve the resolution by a factor 4. In that case, both algorithms are usable.

In table 1 we also show the results for the GRS-M algorithm, which, as said earlier, was tested in an alternate manner. Instead of using a sequence of 50 images and 10 reference frames, we used 50 sequences of 10 images. This way the GRS-M algorithm will use only ten images to estimate each shift; using 50 images is not a real situation. This gives 450 estimates. Note how the estimates improve by at least 10% (and even 60% in one case), in comparison to GRS, which the algorithm relies on. However, the improvement is probably not worth the effort, since each 10 estimates by the GRS-M algorithm needed 90 runs of the GRS algorithm, instead of 9. In general, each n estimates by the GRS-M algorithm need $n(n-1)$ runs of the GRS algorithm.

3.5. Computational cost

The gradient-based shift estimation algorithm (GRS) is very inexpensive, if it were not for the convolutions we need to do to blur the images and calculate the gradient, and scales as $O(mN)$, where m is the constant size of the convolution filters (9 or 11 times the σ of the Gaussian), and N is the number of pixels each image has. The FFTS algorithm (based on the fast Fourier transform, FFT) scales as $O(N \log N)$. FFTSQ-2 and FFTSQ-4 also scale as $O(N \log N)$, but they need to do an inverse FFT on a grid 2^2 and 4^2 times as large as the original images, which takes times $O(4N \log 4N)$ and $O(16N \log 16N)$ respectively. Table 3 shows the actual number of floating-point operations reported (in millions) when using these algorithms as we implemented them (we did not really try to optimize the programs we wrote). We can see that the GRS algorithm takes more computations than FFTS, because of the convolution operator used (3 convolutions are needed, of the input images with 11-by-11 filters). However, we can also see that the difference is larger when N is small. Also notice that the extra cost involved in detecting large shifts is only about 1% for GRS, and 33% for FFTS. This is because the convolutions involve most of the computations of the GRS, and they are done only once. however, the FFTS algorithm needs to do two FFT's and one IFFT, and if the shift is large, another FFT, which is exactly 1/3 of the computation load. All the other operations are neglectable in comparison to these Fourier transforms.

	shift < ½ pixel			shift > ½ pixel		
	N = 32^2	**N = 64^2**	**N = 128^2**	**N = 32^2**	**N = 64^2**	**N = 128^2**
GRS	0.95	3.82	15.30	0.96	3.85	15.45
FFTS	0.20	0.91	4.00	0.27	1.21	5.30
FFTSR	0.21	0.91	4.02	0.27	1.22	5.33
FFTSQ-2	0.41	2.00	8.77	0.41	2.00	8.77
FFTSQ-4	1.60	6.95	30.50	1.60	6.95	30.50

Table 3: Millions of floating-point operations (Mflops) required by each registration algorithm. The first three columns show Mflops for small shifts, and the last three the Mflops when the integer-pixel shift is 1.

4. Image Interpolation

A couple of different techniques are discussed in the literature to compute a high resolution image from a sequence of aliased frames with known shifts. None proved satisfactory.

Most articles [5,6,12,14] describe iterative solutions. These do not satisfy the constraints of our problem because they are too slow. We did implement one of these, however, so as to compare it with our own solution.

Some methods [1,4] make a high resolution image using nearest neighbor interpolation. This is an interpolation method in which each high resolution pixel gets the value of the nearest input pixel. This is an easy algorithm to implement, but the results are poor. No references were found on other interpolation techniques, which might prove better. Popular interpolation techniques like B-splines and cubic convolution are difficult to implement or cannot be used at all on this non-uniformly sampled data, so we studied a few alternative methods. Another important disadvantage of these interpolation schemes is their noise sensitivity. A low SNR makes these advanced interpolation schemes useless.

One other article [7], introduced a method that estimates the high resolution frequency spectrum out of the aliased spectra of the input frames. It is a method which, under certain circumstances, produces excellent results. It is discussed later.

4.1. How many frames do we need?

If the resolution increase in both directions is equal to n, we will create an image with n^2 times the number of pixels, and thus information, than was in one original frame. It is therefore easy to see that we will at least require n^2 frames to have enough information to fill one high-resolution image. This will, however, depend strongly on the distribution of the shifts, as well as on the noise.

Nyquist’s sampling theorem states that a bandlimited signal with a cutoff frequency of ω_c should be sampled with a frequency $\omega_s > 2\omega_c$, so no information is lost [11]. It can, however, also be interpreted as needing N samples in each period NT, where T is $2\pi/\omega_s$, the uniform sampling period [9]. Notice that in the last definition, the sample positions are not taken into account.

To mathematicians this is not new. For example, a polynomial of grade n is completely defined by n+1 points, no matter how close together these points are. However, in mathematics data is never noisy. We are dealing with noisy data, so the position can indeed be of influence. It is common sense to place the available samples as far apart as possible, to minimize the influence of noise (see figure 10).

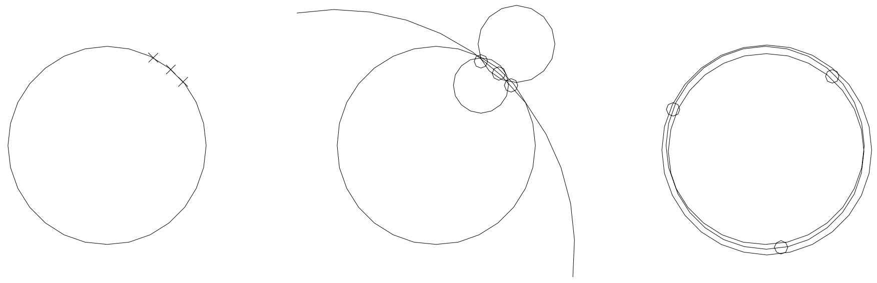


Figure 10: Any three exact samples define a circle (left). However, if the samples contain noise, and are situated close to one another, almost any circle will fit (middle). Situating these samples far apart insures a correct representation in spite of the noise.

Thus, depending on the distribution of the samples, we might need more than n^2 frames to correctly reconstruct the high-resolution image. Reconstructing an image with too few frames is possible, but we should not expect to achieve the desired resolution.

4.2. Resolution vs. noise

The sampled data is considered noisy, and suppressing this noise is always desirable. However, there is no such thing as noise reduction without data loss. In the case of pictures, noise reduction can be done by blurring, averaging each pixel with its neighbors. This blurring not only removes noise, but also suppresses lines and decreases the slope of the edges. Both are perceived as important image features.

In short, there is a trade-off between spatial resolution and signal to noise ratio in image reconstruction. Removing noise smoothes the image, and thus suppresses details. Finding the appropriate combination of resolution and SNR is difficult; all but one of the methods described in this chapter have a parameter that sets the smoothness of the output image.

4.3. Method 1: Interpolation by least square error fitting

The problem we are facing is interpolation of non-uniformly sampled data. Each input frame has samples along a uniform grid, but taking all frames together we get a collection of randomly-distributed samples, which repeats itself along the same uniform grid. The regularity can be used to spare on computation, but the samples are still non-uniformly distributed. This is the main complication to the interpolation.

For one-dimensional data, the simplest interpolation method is zero order hold (ZOH). It is a 0th order method, meaning it has only one degree of freedom. The value of the interpolated function at certain position depends only on one data point. This usually means that the function at that point gets the value of the nearest sample. That is why it is also called nearest neighbor interpolation.

Introducing a second degree of freedom we get first order hold (FOH), also known as linear interpolation. The interpolated function between two samples is a straight line joining both data points. The problem with this approach shows when the sampled data turns out to be noisy. In this case, the linearly interpolated function follows the noise, and a jagged line is the result (see figure 11a). This happens with all methods that use as many data points as degrees of freedom.

To suppress noise, one must average the available data. One method to do this is to fit a straight line through a few points (figure 11b). This can easily be done using least square error fitting. This method effectively suppresses noise, but it also blurs the image (because of the smoothing it performs). The more samples are used in the calculation of a single point, the smoother the output image becomes (see figure 12). Note that if you fit a straight line through only two points at a time, you are doing a linear interpolation.

In two dimensions, the previous discussion is still valid, but linear interpolation has 3 degrees of freedom (and one more for every new dimension), and thus needs at least 3 points. In this case, we are fitting a plane through the data points (this must not be confused with bilinear interpolation, which uses four data points on a regular grid).

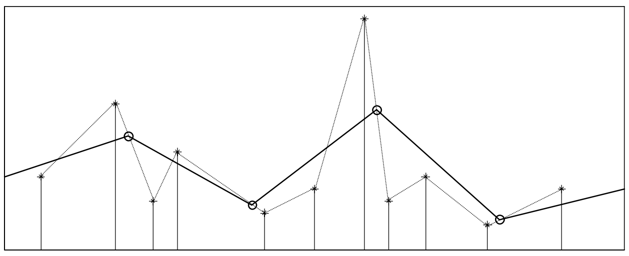


Figure 11a: Resampling data using linear interpolation.

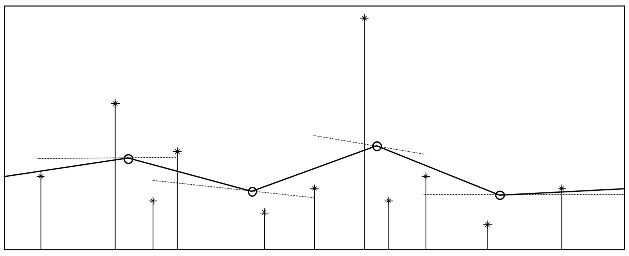


Figure 11b: Least square error fitting over four points.

When interpolating the non-uniformly sampled data, two approaches can be taken. Each output pixel is a combination of either:

- a fixed number of input samples, or
- all input samples in a predetermined area around it.

When the number of samples is constant, the noise suppression is constant over the image, but the size of the interpolation window varies. This means that the resolution changes over the image, which is noticeable by different degrees of blurring. The reverse is true when the size of the window is maintained constant: the noise suppression varies but the resolution is constant.

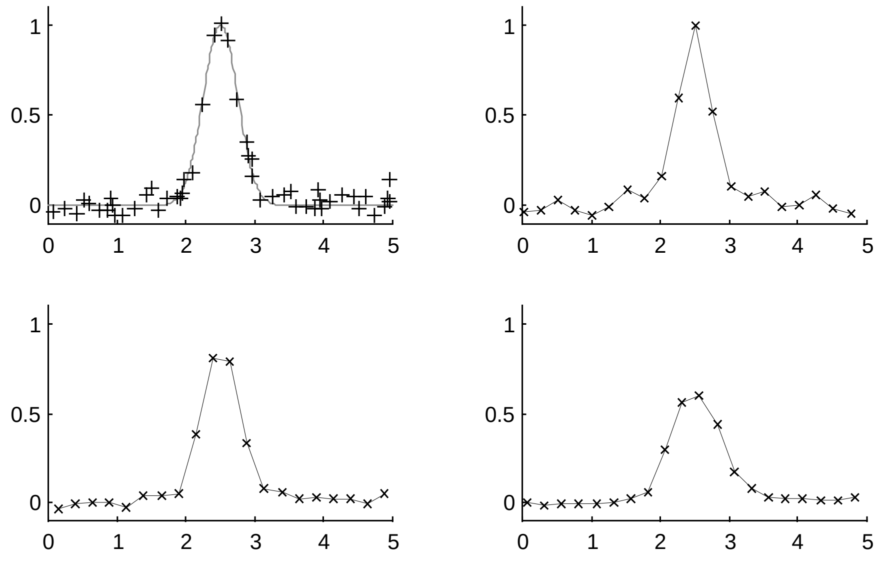


Figure 12: 8 sampled 1D 'images' (top left), interpolated with: linear interpolation (top right), fitting over 4 points (bottom left), and fitting over 8 points (bottom right). Note the jaggedness of the result of the linear interpolation, and the increasing smoothness in the other two graphs.

Least square error fitting is a very simple and inexpensive calculation. The most difficult part of the method is to determine which samples should be used in the calculation. This is done by either

- finding all samples in the specified window, or
- finding the specified number of nearest neighbors.

Once the samples are found (say, values y_i at locations x_i in the one-dimensional situation), the value y' of the pixel at position x' is calculated with

$$y' = a + bx' \quad , \tag{23a}$$

$$a = \tfrac{1}{n}\left(\Sigma\, y_i - b\,\Sigma\, x_i\right) \quad , \tag{23b}$$

$$b = \frac{\Sigma\, x_i y_i - \frac{1}{n}\,\Sigma\, x_i\,\Sigma\, y_i}{\Sigma\, x_i^2 - \frac{1}{n}\left(\Sigma\, x_i\right)^2} \quad , \tag{23c}$$

n being the number of samples. The two-dimensional version is approximately the same, except that b and x_i have two components: you need to estimate a plane through a set of points.

4.4. Method 2: Interpolation by normalized convolution

An alternative approach to interpolation is the use of kernels. When samples are placed along a uniform grid, a continuous function can be generated by a sum of specialized functions (named kernels). These are centered at the grid positions, and multiplied by their corresponding sample value (this is the same as a convolution of the samples with the kernel function). When the samples are distributed at random, this is no longer possible.

The sum of these functions will be too high were the samples are closer together, and too low where they are farther apart.

This problem can be solved by making a ‘mask’ function, a function that does not include any of the data in the samples, only their positions. This function can then compensate for the samples being too close or too far apart. This technique is known in the literature as normalized convolution [8]. The continuous (interpolated) function is then described by

$$i(x) = \frac{f(x) * g(x)}{m(x) * g(x)} \quad , \tag{24}$$

where $f(x)$ denotes the pulse train of the samples,

$$f(x) = \Sigma\, y_i\; \delta(x - x_i) \quad , \tag{25}$$

$m(x)$ the mask function,

$$m(x) = \Sigma\, \delta(x - x_i) \quad , \tag{26}$$

and $g(x)$ the kernel. See figure 13.

We have chosen the Gaussian function as kernel because it is defined and different from 0 in the whole range (-∞,∞), so $i(x)$ in (24) is defined for all x, no matter how sparse the samples are. Other kernels, however, might be easier to compute.

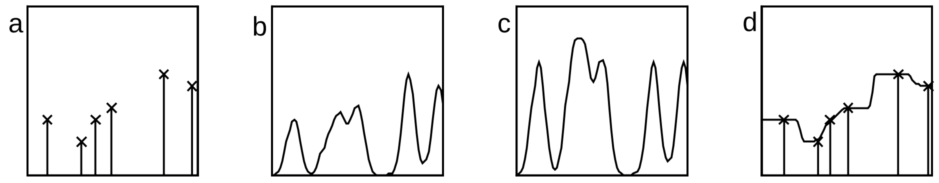


Figure 13: Interpolation by normalized convolution with a narrow Gaussian curve. a) data points. b) f(x)∗g(x). c) m(x)∗g(x). d) data points and interpolated curve.

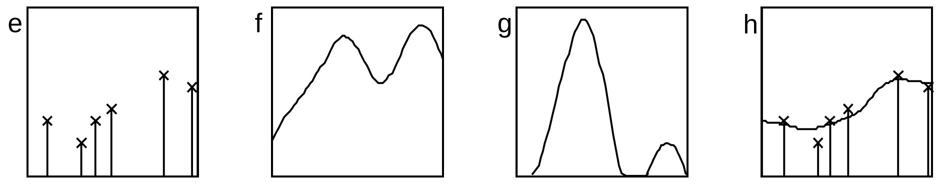


Figure 13 (continued): Interpolation by normalized convolution with a wider Gaussian curve. e) data points. f) f(x)∗g(x). g) m(x)∗g(x). h) data points and interpolated curve. Note the smoothness of the interpolated function.

For resampling at regular intervals, $f(x)$ only needs to be computed at specified locations, which is an easy task. At a location x', the value y' of the pixel is

$$y' = \frac{\sum y_i\; g(x' - x_i)}{\sum g(x' - x_i)} \quad . \tag{27}$$

This also makes it possible to use a different size kernel for each data point. Where samples are far apart, a wider kernel can be used, so that the noise reduction can be kept constant (in analogy to the least square error method with a fixed number of points). Using the same kernel on all data points, a fixed window is achieved, and the method yields a constant resolution over the entire image.

Extension of (27) to the two-dimensional case is as straightforward as it seems.

The reader might note that, because of the fixed patterns in both the input and the output, it is possible to make a set of convolution filters. Each one would be able to calculate the value of a set of output pixels. Interlacing these pixels would then result in the final, high resolution image.

Example: Consider p input images of N^2 pixels, sampled at T in both directions. The output image should be up-sampled n times in both directions, resulting in a pixel pitch of T/n. The output image then has $(nN)^2$ pixels. We would need n^2 filters, each one operating on all p input images, and each one generating $1/n^2$ of the total output pixels. Limiting the action of the filter to an area of m x m input pixels, each output pixel is a linear combination of m^2p points. The weights are computed in advance using the estimated shift of each image. Therefore, such a collection of filters would just require m^2n^2p weights to be calculated, and $p(mnN)^2$ multiplications and additions for the convolution.

4.5. Method 3: Estimation of the frequency spectrum

A completely different approach to the estimation problem was presented by E. A. Kaltenbacher [7]. It is based on frequency spectrum analysis and the fact that the aliasing was done after the image was shifted (see section 2.2). The basic idea is to estimate the original, unaliased spectrum using multiple, aliased spectra.

Each aliased spectrum was formed by adding repeated versions of the original spectrum (as in figure 14). However, each aliased spectrum was formed based on the spectrum of shifted versions of the original scene. This implies that each aliased spectrum is different, and not just by the factor $e^{-j\delta\omega}$ caused by the shift (again, see section 2.2). Using the registration parameters, a set of equations can be solved to calculate each point in the original, unaliased frequency spectrum.

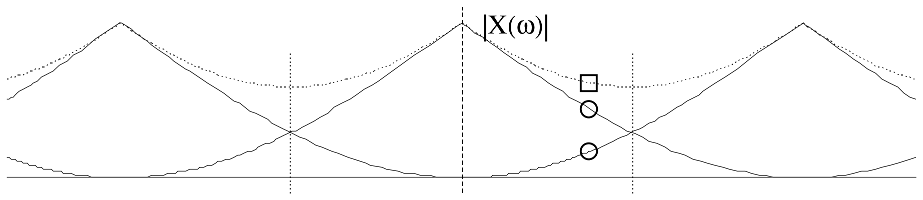


Figure 14: An aliased spectrum (dotted line) is the summation of many realizations of the same, unaliased spectrum. The value of the aliased spectrum at any point ω is the summation of the values of the unaliased spectrum at points ω+mW, with W equal to half the sampling frequency, and m all integer values between minus infinity and plus infinity.

We will discuss the one-dimensional case here. The two-dimensional case is a simple extension, only complicated by the conversion of two-dimensional matrices to vectors (for computational reasons).

We start with the shift property of the Fourier transform,

$$F_k(\omega) = e^{j\delta_k\omega}F(\omega) \quad , \tag{28}$$

where $F_k(\omega)$ is the continuous Fourier transform one of the p input frames, and $F(\omega)$ is the transform of the unaliased image. The camera samples this signal, which converts this equation into

$$\tilde{F}_k(n) = \frac{1}{T}\sum_{m=-\infty}^{\infty} F_k\left(\frac{2\pi n}{NT} - m\omega_s\right) \quad . \tag{29}$$

Here $\tilde{F}_k(n)$ is the discrete Fourier transform of one of the input frames, T is the sampling period and ω_s the sampling frequency. N is the amount of samples, and n is the sample number, in the range [0, N-1]. If we use T = 1 pixel, $\omega_s = 2\pi$ radian per pixel.

All images are band-limited, which means that they have a cutoff frequency ω_c such that

$$F(\omega) = 0 \qquad , \qquad |\omega| \geq \omega_c \quad . \tag{30}$$

We can now define a parameter L such that $L\omega_s \geq \omega_c$. $2L$ is the increase in resolution. We will also need at least $2L$ input frames, because we will have as many overlapping spectra. After setting this parameter, we can change the limits in the summation in (29) to $-L$ and L-1.

For each n we can write a set of equations in matrix form,

$$\mathbf{G}_n = \phi_n\mathbf{F}_n \quad , \tag{31}$$

where $\mathbf{G}_n$ is a column vector with the n^{th} sample in each frame,

$$\mathbf{G}_n(k) = \tilde{F}_k(n) \quad , \tag{32}$$

ϕ_n is the p x $2L$ matrix defined by

$$\phi_n(k, m) = \frac{1}{T}e^{j\delta_k\left(\frac{2\pi n}{NT}+(m-1-L)\,\omega_s\right)} \quad , \tag{33}$$

and $\mathbf{F}_n$ is the column vector with the $2L$ output samples dependent on n,

$$\mathbf{F}_n(m) = F\left(\frac{2\pi n}{NT} + (m-1-L)\,\omega_s\right) \quad , \tag{34}$$

for m = 1, 2, … $2L$. The solution is given by

$$\mathbf{F}_n = \phi_n^{-1}\mathbf{G}_n \quad . \tag{35}$$

When more than $2L$ input frames are used, a least squares solution for the inverse of ϕ_n can be obtained [17]. The solution can then be written as

$$\mathbf{F}_n = \left(\phi_n^{\mathrm{T}}\phi_n\right)^{-1}\phi_n^{\mathrm{T}}\mathbf{G}_n \quad , \tag{36}$$

and will suppress noise.

This method does not work if there are less than $2L$ aliased images available, because it is impossible to invert such a matrix. The largest drawback to this method is that it needs a uniform distribution of the shifts. If the shifts are poorly distributed, there will be information missing, and the matrix inversions will be unstable. Notice that if, for example, two frames have the same shift, there will be two identical equations, and the problem will not have a solution. More than $2L$ images are required to overcome this problem if the shifts are random.

4.6. Method 4: Iterative solution

The best solution to the reconstruction problem can be achieved by minimizing the cost function [5]

$$C(\mathbf{z}) = \frac{1}{2}\sum_{m=1}^{pM}\left(y_m - \sum_{r=1}^{N} w_{m,r} z_r\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{N}\left(\sum_{j=1}^{N}\alpha_{i,j} z_j\right)^2 \quad . \qquad (37)$$

This can only be done by iterating until a small enough error has been achieved. Here, images are written in vector form. $C(\mathbf{z})$ is minimized when the predicted high resolution image $\mathbf{z}$ (N pixels), projected through the observation model $\mathbf{w}$, matches the p measured, low resolution images $\mathbf{y}$ (M pixels). This is the first term on the right hand side of (37).

This problem has, however, not a unique minimum; therefore a smoothness term has been added to the error function. This extra term penalizes sharp edges and noise. α is selected so that this regularization term minimizes when $\mathbf{z}$ is smooth. In [5], Hardie et al. chose

$$\alpha_{i,j} = \begin{cases} 1 \;, & i = j \\ -1/4 \;, & z_j \text{ is a cardinal neighbor of } z_i \end{cases} \quad . \qquad (38)$$

This way, when the pixel z_j is the same as its four neighbors, no contribution is made to $C(\mathbf{z})$. Large differences among neighbors make large contributions to the cost function.

λ is a parameter that controls the influence of the last term, and thus controls the smoothness of the output. This parameter is comparable to the ‘window size’ parameter in the other two methods described above.

We consider this the best solution because the output is matched with the input, achieving an optimum. This is, of course, only optimal under the assumptions taken (the observation model, the square difference error measure, etc.). We consider this method better than the previous one (estimation of frequency spectrum) because we are dealing with (possibly too few) randomly shifted images.

An iterative solution is not useful in real-time applications, so this method is only added to this report for comparison. Both gradient descent and conjugate gradient minimization algorithms were implemented, as described by Hardie et al., and are used to compare the other methods to. For more information on conjugate gradient and other minimization algorithms, see [16].

4.7. One-dimensional interpolation experiments

To get acquainted with the interpolation techniques presented in this chapter (and especially with their parameters), we started by doing some experiments on one-dimensional data. We have studied not only the accuracy of the result, but also the noise suppression.

The experiments we did were on a one-dimensional, normalized Gaussian curve,

$$g(x) = e^{\frac{-(x-\mu)^2}{2\sigma^2}} \quad . \tag{39}$$

It was sampled at a very low rate at fixed intervals, but with a random offset. The sampling period is 4σ, which means that the signal is undersampled by a factor 4 (see Appendix A). This serves as one low-resolution, one-dimensional image. A series of these images, each one with a different offset, was used as input for the interpolation algorithms. See the example in figure 15. The random offset of each image is assumed to be known for the purpose of testing the interpolation methods.

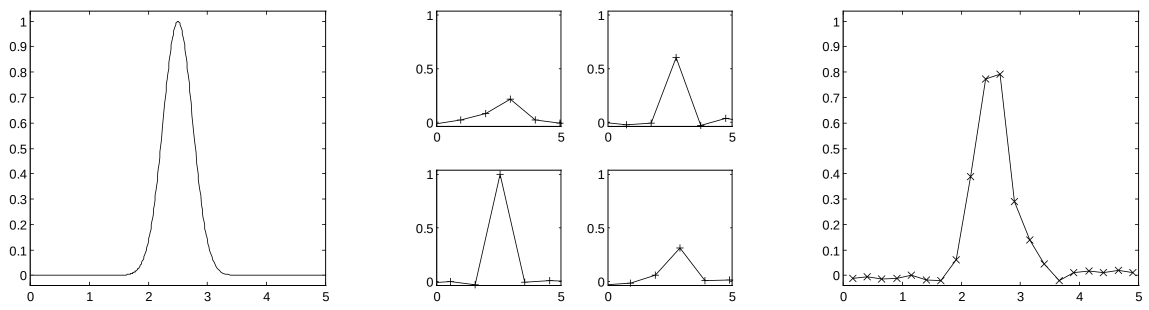


Figure 15: Original image (left), four undersampled input images (center), and the high resolution output image (right), acquired with linear interpolation on the four input images.

The output 1D image has 4 times the sampling density of the original images (that is, 20 points). It is, thus, sampled densely enough to satisfy the Nyquist criterion. This does not mean that there is no aliasing, which was introduced in the input images.

The noise added to the samples is $N(0, \sigma_n^2)$, making the signal to noise ratio of the input images (in dB)

$$SNR = 20 \log_{10} \frac{1}{\sigma_n} \quad . \tag{40}$$

To test the performance of the interpolation, an objective measure for the error of the output is needed. The best way to see how well the algorithm performs is to look at the output image, and examine the SNR, detail, contrast, etc. There is, however, no way to tell a computer how to do this, so this is not an objective measurement. The next best error seems to be the L^p-norm,

$$L^p = \left(\frac{1}{2L} \int_{-L}^{+L} \left| f(x) - g(x) \right|^p dx \right)^{1/p} \quad . \tag{41}$$

Here, *f*(*x*) is the measured function and *g*(*x*) is the expected result. This is, however, very difficult to do analytically, and should be done numerically. The sampled version of this method reads

$$L_N^p = \left(\frac{1}{2N} \sum_{n=-N}^{+N} \left| f[n] - g[n] \right|^p \right)^{1/p} \quad , \tag{42}$$

and can be proven not to be influenced by aliasing if both signals are sampled according to Nyquist and $p = 2$ (see Appendix B).

This is the error measure we have used, with $p = 2$, and without the constant term *2N*. The problem with this error measure is that it does not differentiate between uncorrelated noise and "systematic" errors due to smoothing, which suppresses small details. In the example of the peak in figure 15, the error might be the same for a very noisy output image as for an image with a flattened peak, but which otherwise looks very good.

4.7.1. Implementation

Two algorithms each were implemented for methods 1 and 2, as both methods know two variations. Each algorithm has a "window size" parameter, defined as follows:

- The algorithm for least square error fitting with variable window size uses *n* points: $n_l = \lfloor n/2 \rfloor$ points to the left and $n_r = n - n_l$ points to the right of the location for the high-resolution pixel.
- The algorithm for least square error fitting with fixed window size uses all points in the window. But if all points are to the left (or to the right) of the location for the HR pixel, or if there are no samples at all in the window, the value of the nearest neighbor is copied. This is necessary because there must be *interpolation*, not *extrapolation*. Any value extrapolated can be much too large, creating a peak in the output that was never present in the input.
- The algorithm for normalized convolution with variable window size uses a Gaussian kernel with σ equal to half the distance to the kth nearest neighbor.
- The algorithm for normalized convolution with a fixed window size uses a Gaussian kernel with σ equal to a preset fraction (¼) of the window size.

4.7.2. Results

The 1D interpolation algorithms have been tested. The errors are studied as a function of window size and the number of input frames. The results are presented in figure 16 through figure 19. Each of the points in these figures is the average of 20 realizations of the same experiment. Each method is evaluated for 2 SNRs: 40dB (high SNR) and 20dB (middle SNR). The noise added to the input images was Gaussian distributed.

It immediately shows that the more frames are used, the better the results are. Also, each method has a different “best choice” window size depending on the number of images that are combined and on the SNR in the input.

The interpolation with least square error fitting is consistently better than the normalized convolution algorithm, although the choice of the parameter is more critical.

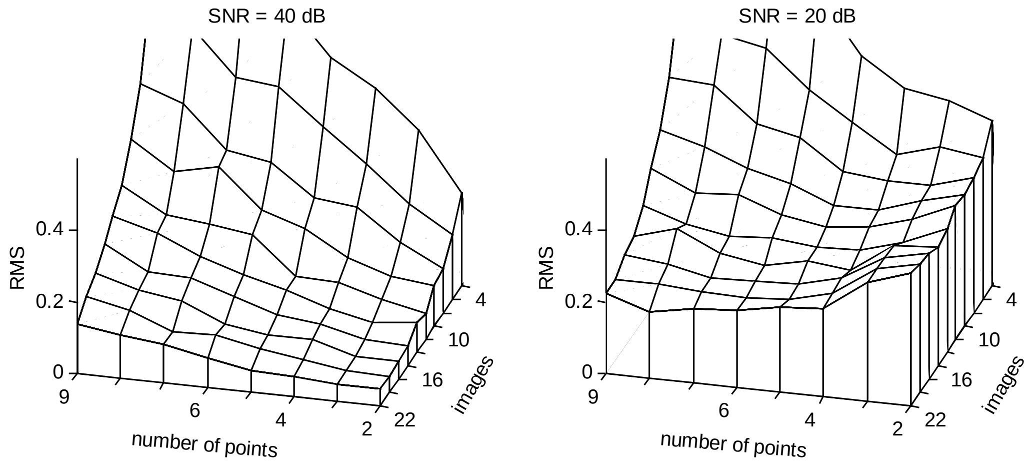


Figure 16: L^2-norm as a function of the number of points and the number of input frames, for the least square error fitting algorithm with variable window size.

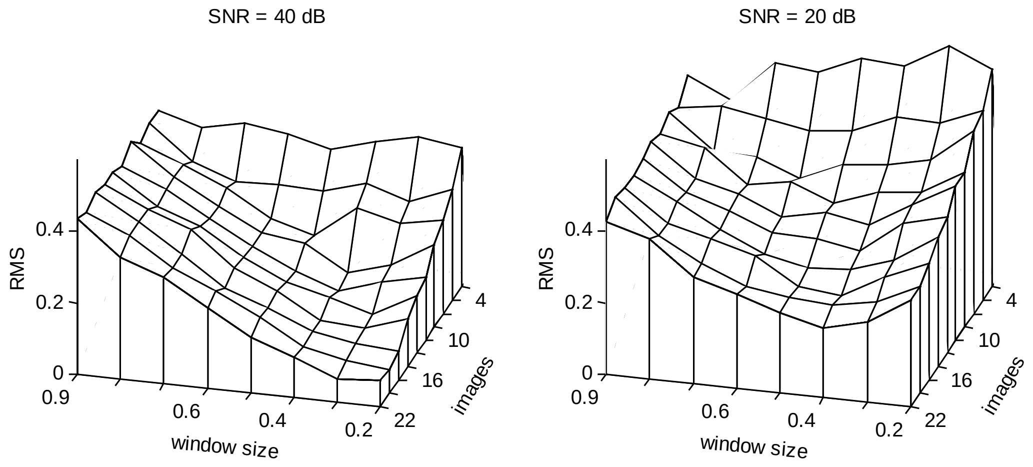


Figure 17: L^2-norm as a function of the window size and the number of input frames, for the least square error fitting algorithm with fixed window size.

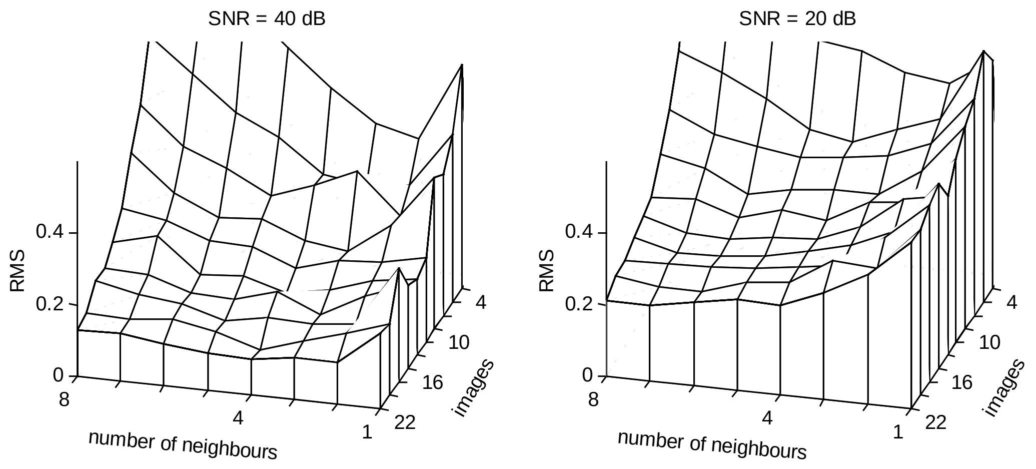


Figure 18: L^2-norm as a function of the number of neighbors and the number of input frames, for the normalized convolution algorithm with variable window size.

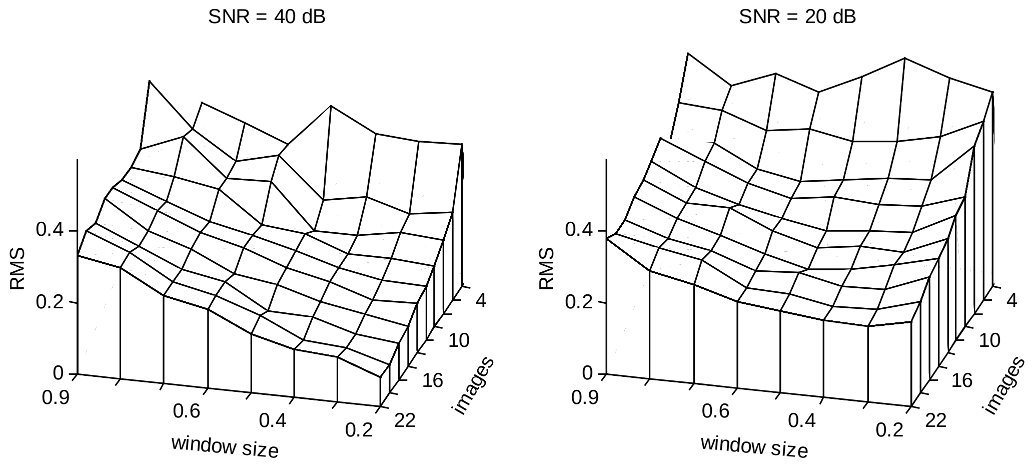


Figure 19: L^2-norm as a function of the window size and the number of input frames, for the normalized convolution algorithm with fixed window size.

The influence of errors in the shift estimation was also tested. Figure 20 shows the results for normalized convolution with fixed window size (window size was 0.3, the kernel was a Gaussian with $\sigma = 0.075$), using 10 input images and an upsampling factor 4. The other methods were similarly influenced. This figure shows the influence of noise added to both the shift and the sample values.

The SNR of the shift indicates the variance of the noise added to the shifts before the interpolation was done. The signal level is set to the pixel pitch, so 20dB indicates that the error in the shift is normally distributed with a standard deviation of 10% of the original pixel pitch.

As seen in figure 20, the influence of the noise in the shifts is about equivalent to the noise in the samples. Also, it appears that small amounts of noise in the shifts have no effect at all on the interpolation error. This was also seen when the shift data was rounded off: only the first decimal was relevant in these experiments.

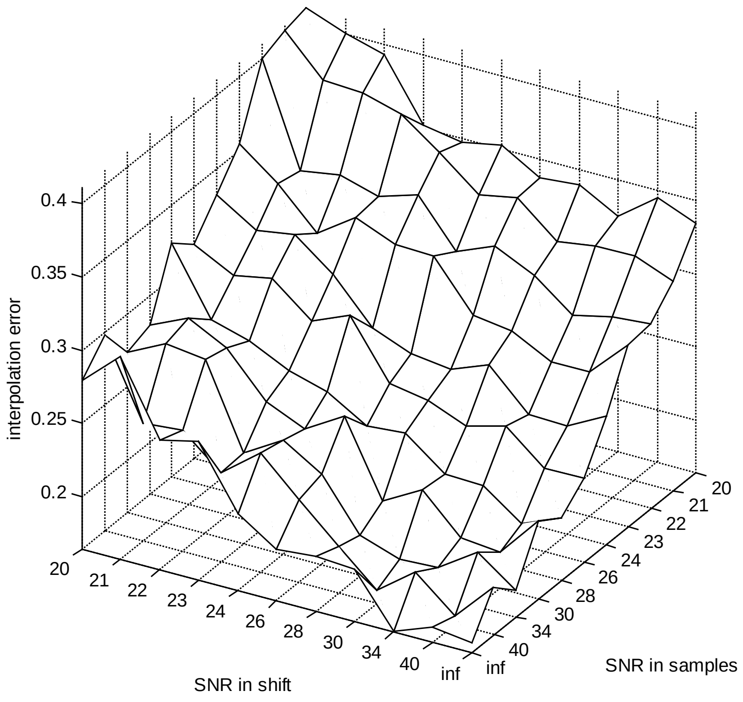


Figure 20: Influence of noise in the shifts on the interpolation error. The variance in the errors is very high; nevertheless, each point is the mean of 50 realizations.

4.8. Two-dimensional interpolation experiments

The experiments from the previous section were repeated with two-dimensional images. The results were roughly the same. Different images were used to test the interpolation accuracy of both methods (see figure 21):

- a Gaussian with σ set to one fourth of the sampling period (factor 4 undersampled),
- a step function, and
- a sine wave with the same period as the sampling function (factor 2 undersampled).

The error measure is, again, the mean square error (L^2-norm). The (normally distributed, additive) noise in the input images is 40dB.

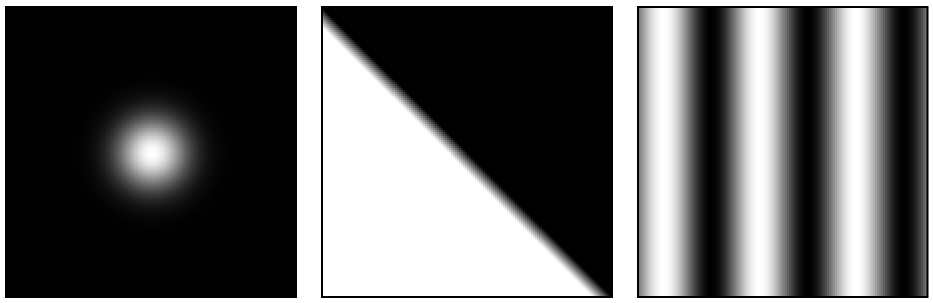


Figure 21: Test images used to measure the interpolation accuracy of the algorithms. Left, a Gaussian; middle, a step function; right, a sine wave.

4.8.1. Implementation

Two interpolation algorithms were implemented: least square error fitting over a fixed number of points (LSE), and normalized convolution with a fixed kernel size (NC). The last one is exactly the same as the one-dimensional equivalent. The former one, however, is a bit more complicated. First of all, at least three data points are needed to fit a plane. To avoid extrapolation, the near neighborhood of a high resolution sample point is searched for the n nearest data points that surround it. This is done in the following two steps:

1. Using Delaunay triangulation, the data points are searched for the smallest triangle that contains the high resolution sample point.
2. To these 3 points, other ones are added to from a list of nearest data points.

Using the regularity in the sampled data, this process only needs to be done n^2 times (with n the upsampling factor). In analogy to the discussion at the end of section 4.4 on normalized convolution, we can limit the search area to 3x3 input pixels. The Delaunay triangulation then needs to be done once, on $9p$ data points (p is the amount of input images). This triangulation data has to be searched n^2 times, and each search gives a set of data points to be used in the calculation of $1/n^2$ of the output pixels.

Once the data set is known, performing the least squares fit is straightforward.

When using only 3 points, the LSE algorithm performs as if it were generating a surface through all data points (just like a rendering program does for 3D structures) and then sampling this surface at the high resolution grid points.

4.8.2. Results

As stated earlier, the results for these two-dimensional experiments are practically the same as for the one-dimensional ones. Figure 22 shows graphs with the root mean square error made by the least square error interpolation algorithm applied to the images in figure 21. Note the similarity to the left graph in figure 16. Also note the difference in scale for the vertical axis (the error). This is due to the different undersampling factors in the three input images.

The results using normalized convolution (not shown) are also scaled duplicates of the one-dimensional experiments.

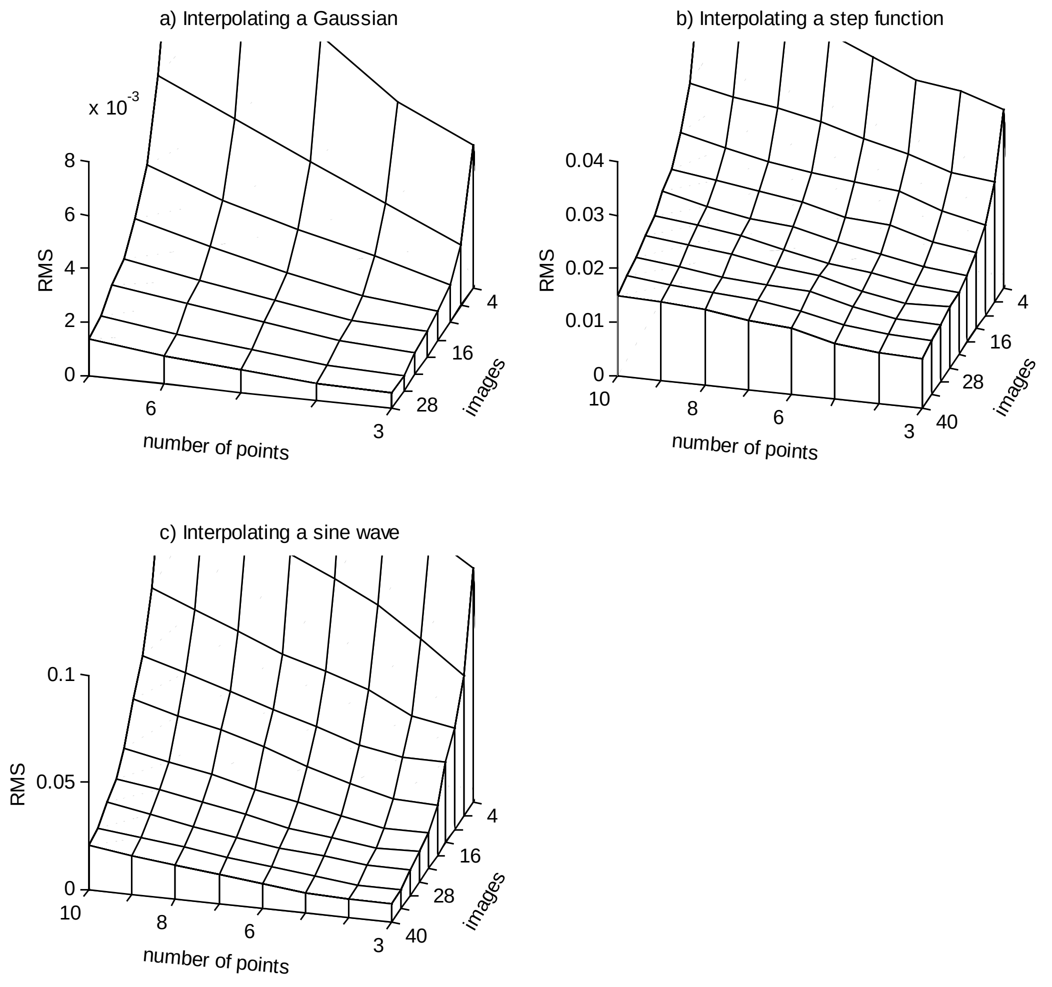


Figure 22: L^2-norm as a function of the number of points and the number of input frames, for the least square error fitting algorithm with variable window size. Input images are a) a Gaussian curve (σ = ¼), b) a step function, and c) a sine wave with period 1.

4.9. Image interpolation experiments

We have evaluated the four methods presented in this chapter by applying them to a series of images acquired by sub-sampling the image “trui” (figure 9, left). It was sub-sampled by a factor 4 in both directions, in the same manner as was done for the registration experiments (see section 3.4). These images were used, along with the known shifts, to estimate a high resolution image (upsampling factor 4), which could be compared to the original image.

The four methods presented in this chapter were compared by using the same sequence of frames. The output image is compared to the original in the root mean square difference sense (L^2-norm). This gives a numerical quantization of the performance of each algorithm. See section 4.7 for a justification for the use of the root mean square error.

The four algorithms used here are:

- Interpolation by least square error fitting over a fixed number of points (LSE), as in the previous section. The window parameter *w* is the number of points used.
- Normalized convolution with a fixed kernel size (NC), as in the previous section. The window parameter *w* is chosen so that the Gaussian kernel used has $\sigma = w/4$.
- The frequency spectrum estimation algorithm (FE).
- A conjugate gradient minimization algorithm (CG) that minimizes (37).

The CG algorithm is expected to give the best results, because it is allowed to run until the output best matches the input in accordance to the model used to build the cost function. If CG fails to perform, it is because the model was not correct.

A gradient descent minimization algorithm has also been implemented and tested. As it minimizes the same cost function CG minimizes, its results are the same, but with a slower convergence rate. Therefore, these results are not presented here.

4.9.1. Results

To see that each of these algorithms indeed improves the resolution of the input frames, compare the results in figure 23 through figure 25 to the top two images in each figure. The top left image is one single frame, and the top right one is that same frame, enlarged by bicubic interpolation to the same number of pixels as the other images. Notice, for example, the reflection in the eyes, and the structure of the wool on both the hat and the sweater.

The sequence of frames used as input have random displacements, as shown in figure 28. The frames marked as ‘o’ are the first 10 in the sequence, and were always used. Where 16 frames were required, those marked with a ‘*’ were used as well. The frames marked with ‘+’ are the last 9, and were used only when 25 frames were required. The first 16 frames have a poor distribution, only all 25 frames taken together sufficiently span the whole area.

Table 4 shows the root mean square error between the reconstructed high resolution images and the original “trui”, for all four algorithms. LSE was used with window parameters *w* of 3 and 6, and NC with *w* = 0.3 and 0.6. GD was allowed to run for 10 steps, starting with a bilinear interpolate of the first frame, and λ equal to 10^{-4}.

As expected, this table shows us that GD provides the best solution by far. However, a single step of GD is about four times as complicated (computationally) as LSE. Note that neither NC nor LSE perform too badly in comparison to this iterative method. Not shown in the graph, but also important, is the fact that the first step of GD produced a result better to that of LSE or NC (a RMS of $1.14 \cdot 10^{-2}$ with 10 images, $7.47 \cdot 10^{-3}$ with 25).

This table also shows us how (because of the high SNR) LSE performs better with a small window parameter. NC, however, performs better with a small *w* only if the amount of input frames is large enough. If there are only a few frames available, a larger *w* will provide a better interpolation. This is the same conclusion we got after studying the one-dimensional versions in section 4.7. Remember that NC will mimic a nearest neighbor interpolation in those parts of the image where the samples are far apart. LSE will always be comparable to linear interpolation.

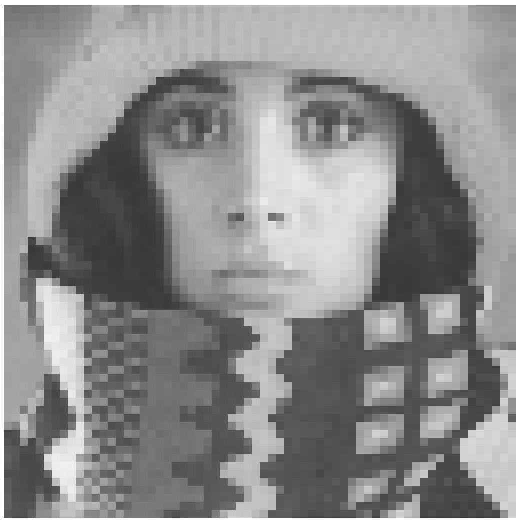
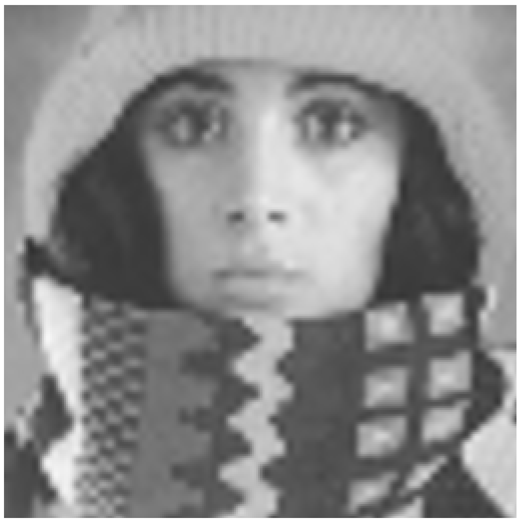
	LSE, *w*=3	**LSE, *w*=6**	**NC, *w*=0.3**	**NC, *w*=0.6**	**FE**	**GD**
10 images	$1.84 \cdot 10^{-2}$	$2.16 \cdot 10^{-2}$	$2.26 \cdot 10^{-2}$	$2.15 \cdot 10^{-2}$	*no solution*	$9.05 \cdot 10^{-3}$
16 images	$1.52 \cdot 10^{-2}$	$1.63 \cdot 10^{-2}$	$1.62 \cdot 10^{-2}$	$1.68 \cdot 10^{-2}$	$7.96 \cdot 10^{-2}$	$5.59 \cdot 10^{-3}$
25 images	$8.85 \cdot 10^{-3}$	$1.11 \cdot 10^{-2}$	$1.08 \cdot 10^{-2}$	$1.27 \cdot 10^{-2}$	$1.79 \cdot 10^{-2}$	$3.03 \cdot 10^{-3}$

Table 4: Root mean square difference between the reconstructed images and the original one. As a comparison measure, the RMS of the bicubic interpolated image is $5.10 \cdot 10^{-2}$.

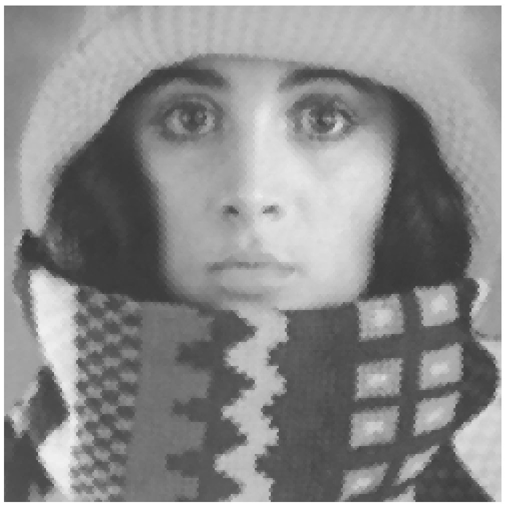
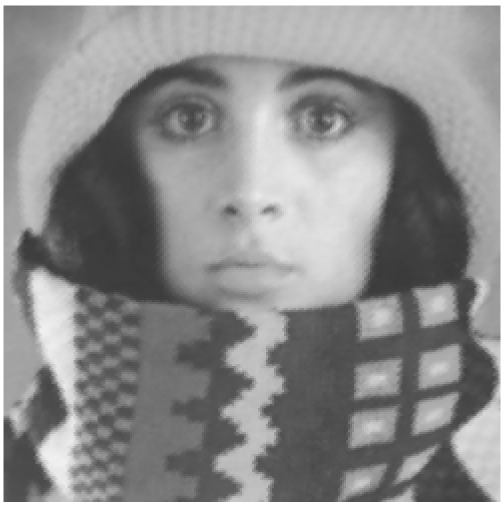
As stated before, figure 23 through figure 25 show the results acquired with the interpolation algorithms, as well as a single low-resolution input frame for comparison. Each of the figures show the results for a different number of input frames. Figure 23 is for the case where there are too few frames available to acquire the desired resolution increment (4^2). The results are acceptable.

In figure 27 the results of GD with 10 frames and LSE with 25 are compared. Note that these two indeed closely resemble each other, as expected by their similar, low RMS. The gradient descent minimization algorithm did not converge after 10 iterations, but it was quite close to convergence.

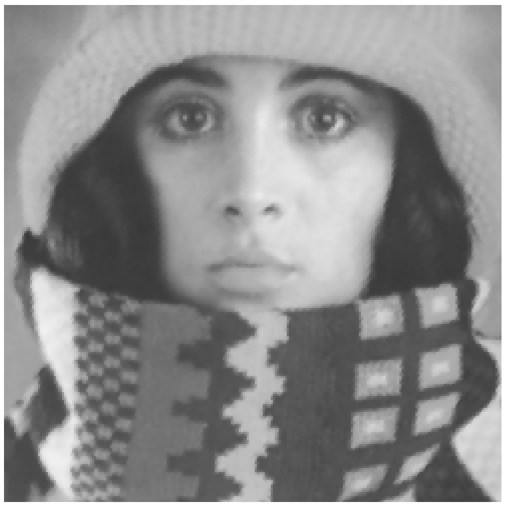
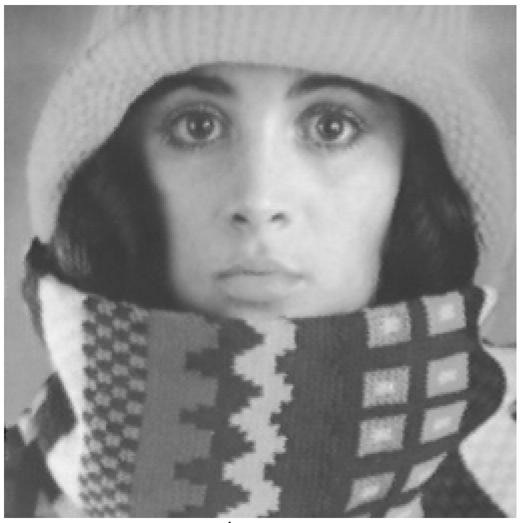
The image produced by FE is worth discussion. After looking at the image (figure 26, left) it is apparent that the resolution has indeed increased by a factor 4 in both directions. All the details are there. But there is a disturbing “grid” superimposed over the image. It is a structure with a frequency of 4 pixels in both directions, equal to the resolution increment, and is caused by lines in the frequency spectrum (figure 26, right). These lines mark the borders between the 16 blocks that were estimated, based on spectra of the input frames. These lines are very small in magnitude, and the contrast of the image was stretched to show them. The cause is mainly the lack of evenly distributed shifts in the input sequence. Other input sequences gave much better results. Adding more frames (see figure 25, bottom left image), or using frames with better distribution of shifts, these lines disappear. When the input sequence has been obtained by downsampling the original image 4 times in each direction (generating 16 frames with displacements (0,0), (0, 0.25), (0, 0.5), etc.), the result of the FE algorithm is exact (a root mean square difference with the original of about $3 \cdot 10^{-16}$, caused primarily by round-off errors). Note that in the result of the FE algorithm with 25 frames, some of the grid pattern is still visible along the edges of the image. This causes the RMS to be lower than one would suggest.

a) One low-resolution frame *b) Bicubic interpolation*

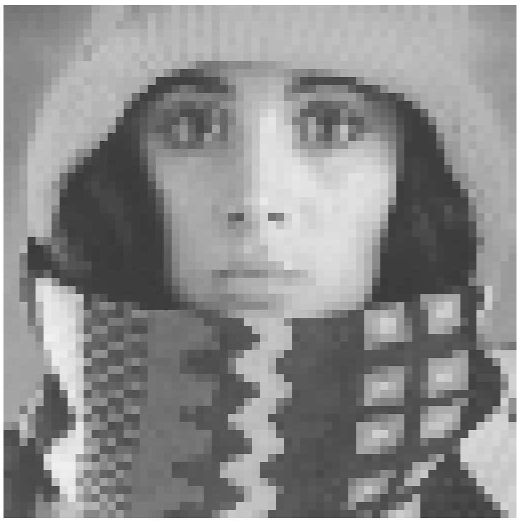



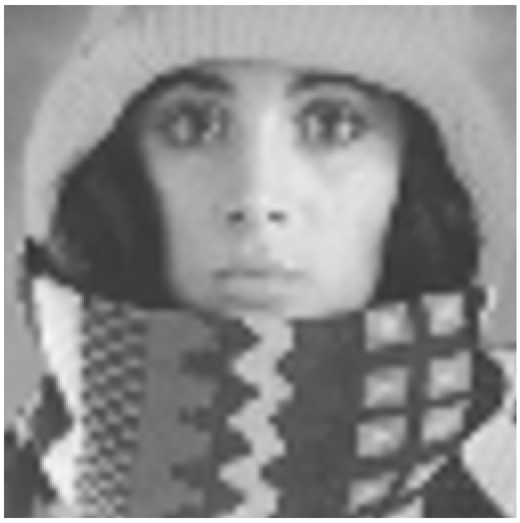
c) NC with w = 0.3 *d) LSE with w = 3*

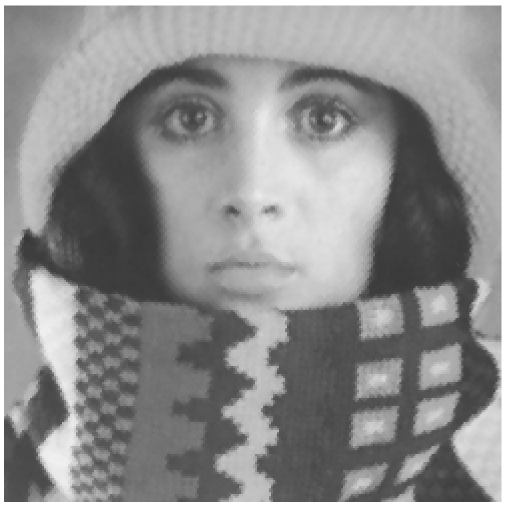



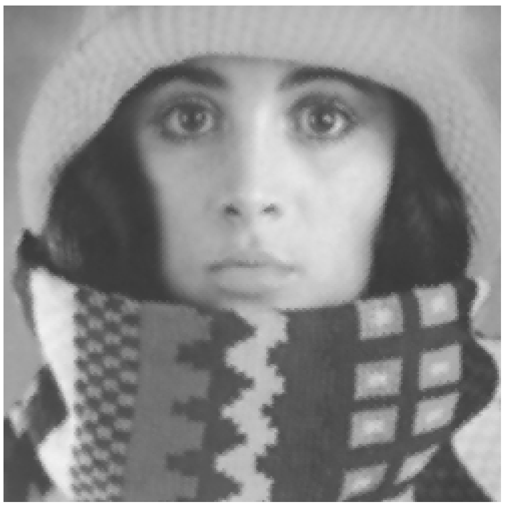
e) NC with w = 0.6 *f) CG with $\lambda = 10^{-4}$, after 10 steps*

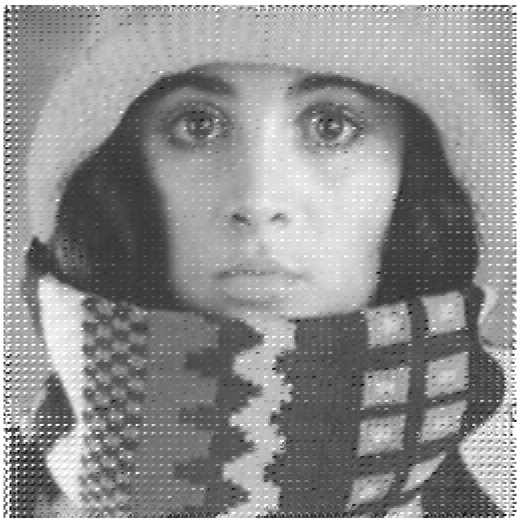
Figure 23: Results after interpolation using 10 input images.


a) One low-resolution frame


b) Bicubic interpolation


c) NC with w = 0.3


d) LSE with w = 3


e) FE

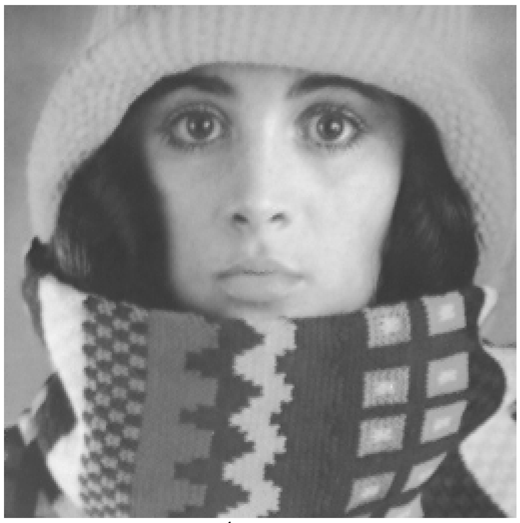
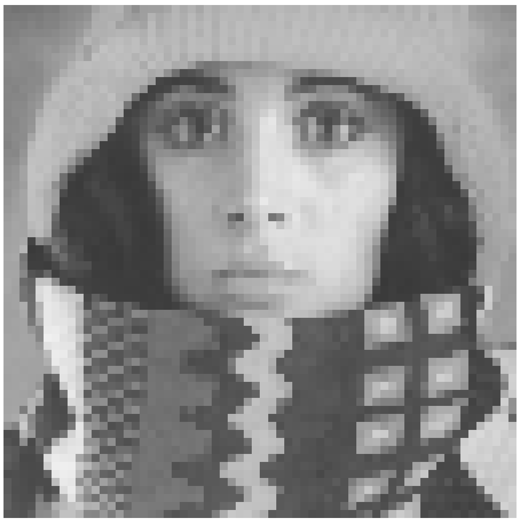
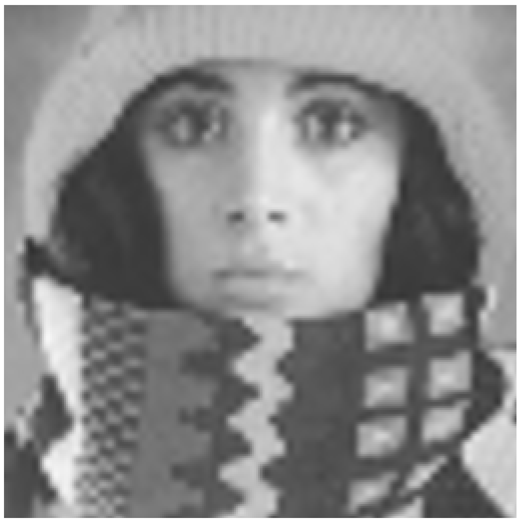
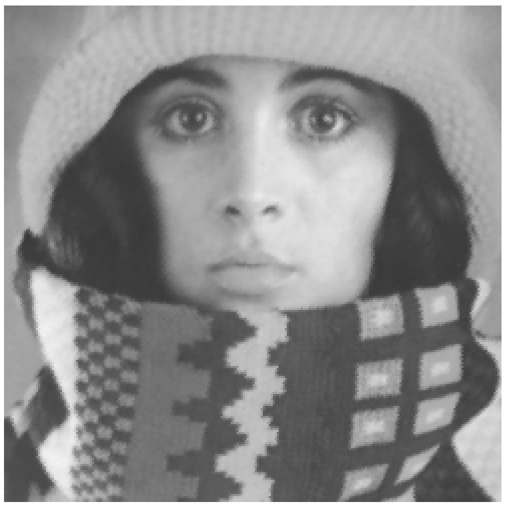
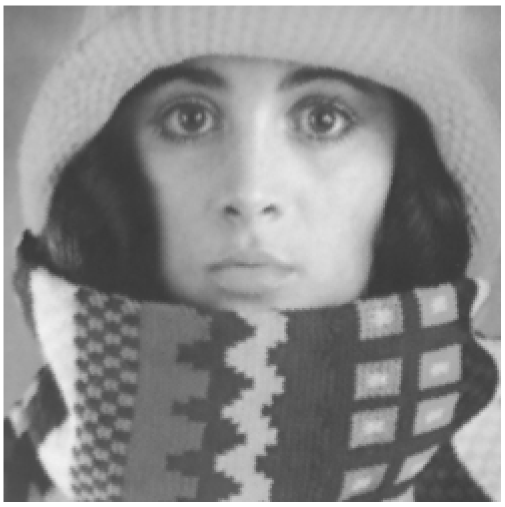

f) CG with $\lambda = 10^{-4}$, after 10 steps

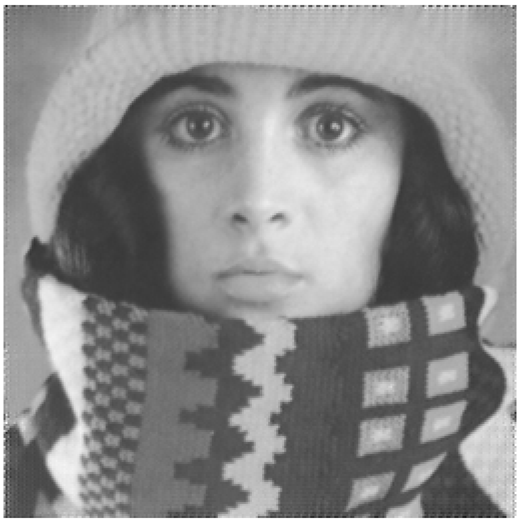
Figure 24: Results after interpolation using 16 input images.


a) One low-resolution frame


b) Bicubic interpolation


c) NC with w = 0.3


d) LSE with w = 3


e) FE

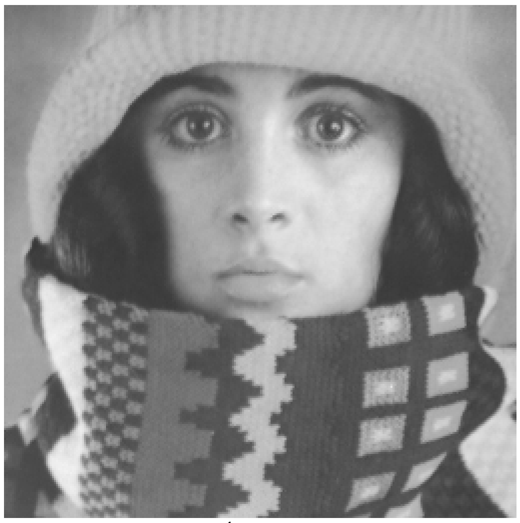

f) CG with $\lambda = 10^{-4}$, after 10 steps

Figure 25: Results after interpolation using 25 input images.

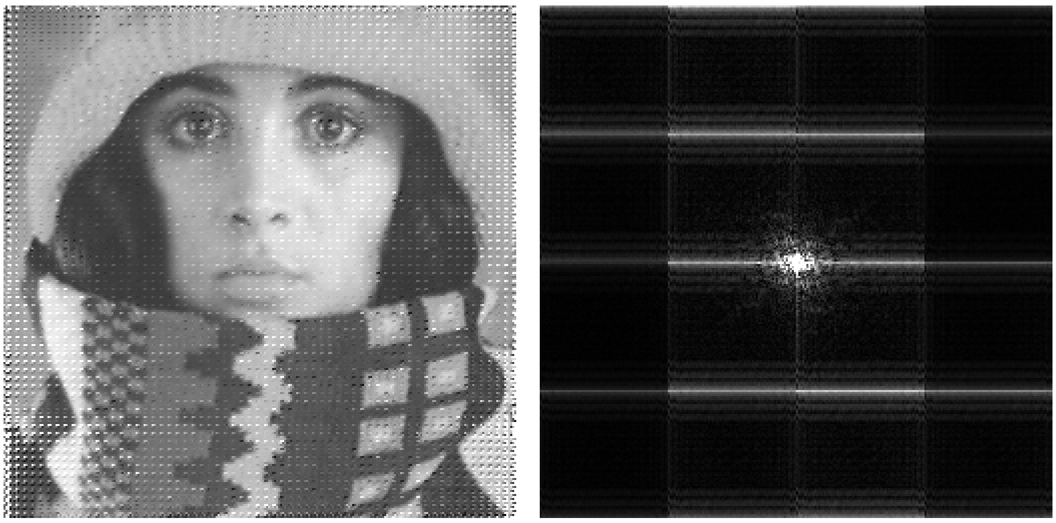


Figure 26, left: Result of FE algorithm using 16 input images, with disturbing "grid" superimposed over the image. See text for details. Right: Estimated frequency spectrum, contrast stretched to show the block structure that causes the artifacts.

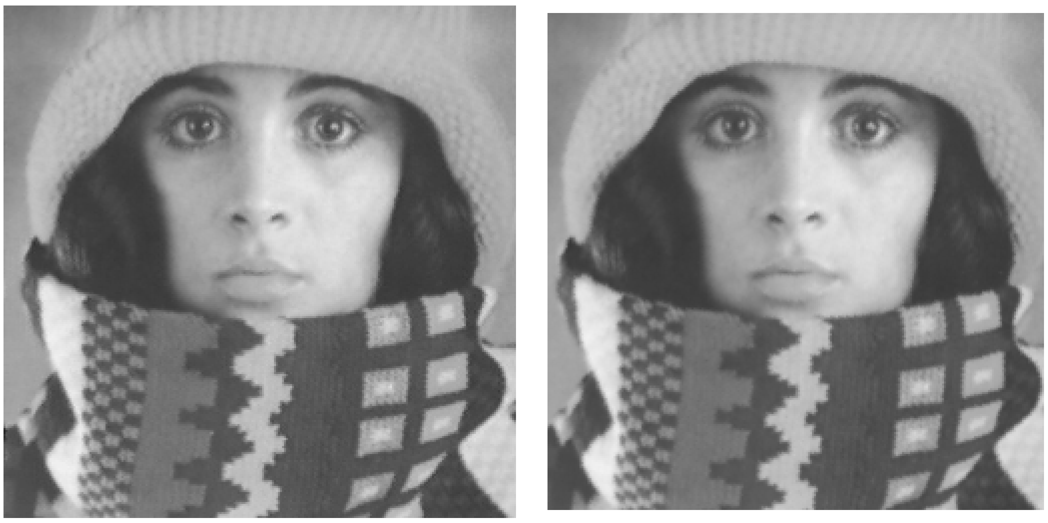


Figure 27, left: Result of CG with 10 frames, after only 10 iterations. Right: Result of LSE (w=3) with 25 frames. Note that the quality is indeed similar in both images.

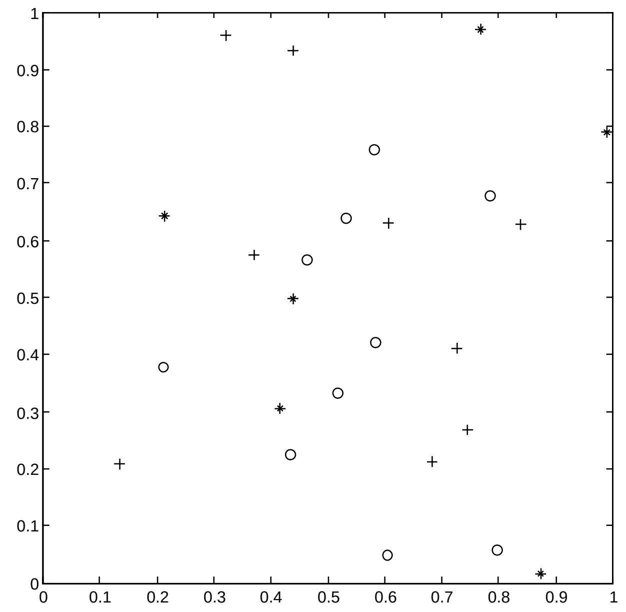


Figure 28: Relative displacements used to generate the low resolution sequence of "trui". When ten frames were required, the ones marked with a 'o' were used. The sequence of 16 frames used those marked with the 'o' and the ''. The sequence of 25 frames used them all.*

4.10. Computational cost

We have measured the computational cost of the algorithms tested in the previous section. We will use N as the number of pixels in each low-resolution frame, n is the increase in pixels for the output image (which then has Nn pixels), and p is the number of input frames used. These three quantities determine the number of operations each algorithm needs to perform.

NC is the easiest method to implement, and also the easiest to analyze. It scales $O(pnN)$. The LSE algorithm, however, is less dependent on p, because only the Delaunay triangulation uses an amount of data proportional to p. The bulk of the computations are $O(wnN)$, where w is the number of samples used to calculate each high-resolution pixel.

The FE algorithm uses the FFT, which scales $O(N \log N)$. It also inverts N matrices size pxn, which is an operation $O(n^3)$ if $p = n$, or $O(pn^2 + n^3)$ if $p > n$. The whole algorithm should scale $O((p+n)N \log N + Npn^2)$.

CG is more difficult to analyze. The initialization requires $O(N)$. Then each step does three mappings: twice from the a high-resolution image to the low-resolution images ($O(pN)$), and once the inverse ($O(nN)$). Also some convolutions of the high-resolution image are required ($O(nN)$). It probably scales $O(nN + pN)$.

Table 5 shows the actual number of floating point operations (in millions) reported when the algorithms were tested. No optimizing was attempted. We see that LSE indeed scales independently from p. For large p, NC is an order of magnitude more expensive than LSE. This is because LSE was used with $w = 3$, which means that each high-resolution pixel was calculated using only 3 input samples. NC always uses 9*p* samples, no matter how small the window parameter is taken. That means that a lot of useless computations are being done. This can be changed if a more complex program is written. We believe that the normalized convolution algorithm is the one that could profit most from optimizations.

Finally, the conjugate gradient (CG) algorithm indeed requires a number of operations directly proportional to both *N* and *p*; however, *n* has less influence than *N* on the complexity.

	N=64^2, n=2^2	N=32^2, n=4^2	N=64^2, n=4^2	N=128^2, n=4^2
LSE (*w*=3), *p* = 10	0.86	0.88	3.42	13.82
LSE (*w*=3), *p* = 16	0.87	0.95	3.49	13.89
LSE (*w*=3), *p* = 25	0.90	1.05	3.60	13.99
NC, *p* = 10	2.98	2.80	11.89	49.06
NC, *p* = 16	4.64	4.36	18.54	76.50
NC, *p* = 25	7.13	6.71	28.52	117.67
FE, *p* = 16	27.45	29.51	118.70	476.72
FE, *p* = 25	42.07	66.72	268.13	1,070.00
CG initialization	0.33	0.33	1.32	5.26
CG (1 step), *p* = 10	8.47	2.36	13.00	55.50
CG (1 step), *p* = 16	13.45	3.68	18.06	75.40
CG (1 step), *p* = 25	20.92	5.67	25.64	105.30

Table 5: Millions of floating point operations (Mflops) needed by each image estimation algorithm, for different number of input images (p), size of input images (N), and size increment (n). The output image has Nn pixels, so for the first two columns equal size images were created.

5. Comparison of the Methods

In this chapter we will summarize the results found in the two previous chapters, and then we will make a choice based on performance, cost, and other considerations. Finally, we will give some of the results we obtained using real images.

5.1. The methods we chose

When reviewing the performance of the registration algorithms, we found that gradient-based shift estimation (GRS) and estimation of shifts from the fast Fourier transform of the cross-correlation (FFTS or FFTSR) are the only choices to be considered. GRS performs better, but is also a bit slower due to the convolutions. However, FFTS scales as O(N log N) and needs 33% more operations if the shift is larger than half a pixel (since one image has to be corrected for the integer-pixel shift first). GRS has the inconvenience of needing an iterative portion (or using the first half of the FFTS algorithm, as described at the end of section 3.2). To reconstruct the images at the end of this chapter we used GRS, simply because it is the most accurate of the two.

Of the two interpolation methods we used to fuse the available data, LSE and NC, LSE was much faster, but probably only because of a lack of optimization in the NC algorithm. Some changes to NC might make its performance even better to that of LSE. The advantage of NC over LSE is that it does not need Delaunay triangulation of the data, and is very simple to implement. It is also the most versatile of the two methods. We also reviewed two other image estimation algorithms. One is iterative, and thus useless under the constraints of the problem. The other has two major drawbacks: its sensitivity to poor distribution of shifts, and its enormous computation load. We finally chose NC to reconstruct the sample images. However, LSE must not be dismissed.

5.2. The results

We recorded a couple of sequences at TNO-FEL, using a camera as described in chapter 2. To get the required motion, we shook the table the camera was on. The camera corrects for pixel offset and gain variance, but no correction was done for hot and cold pixels. Figure 29 through figure 34 show some of these images and the results we got with them.

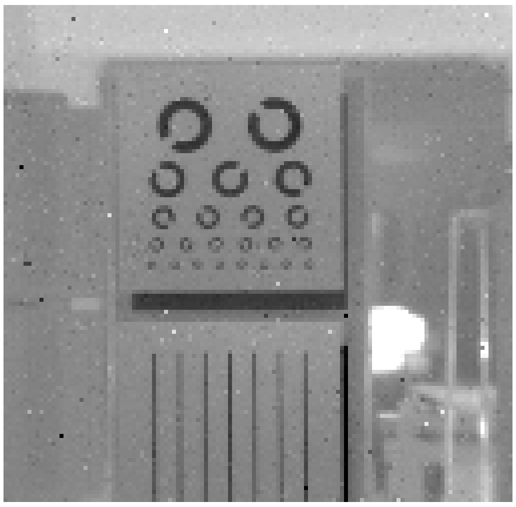
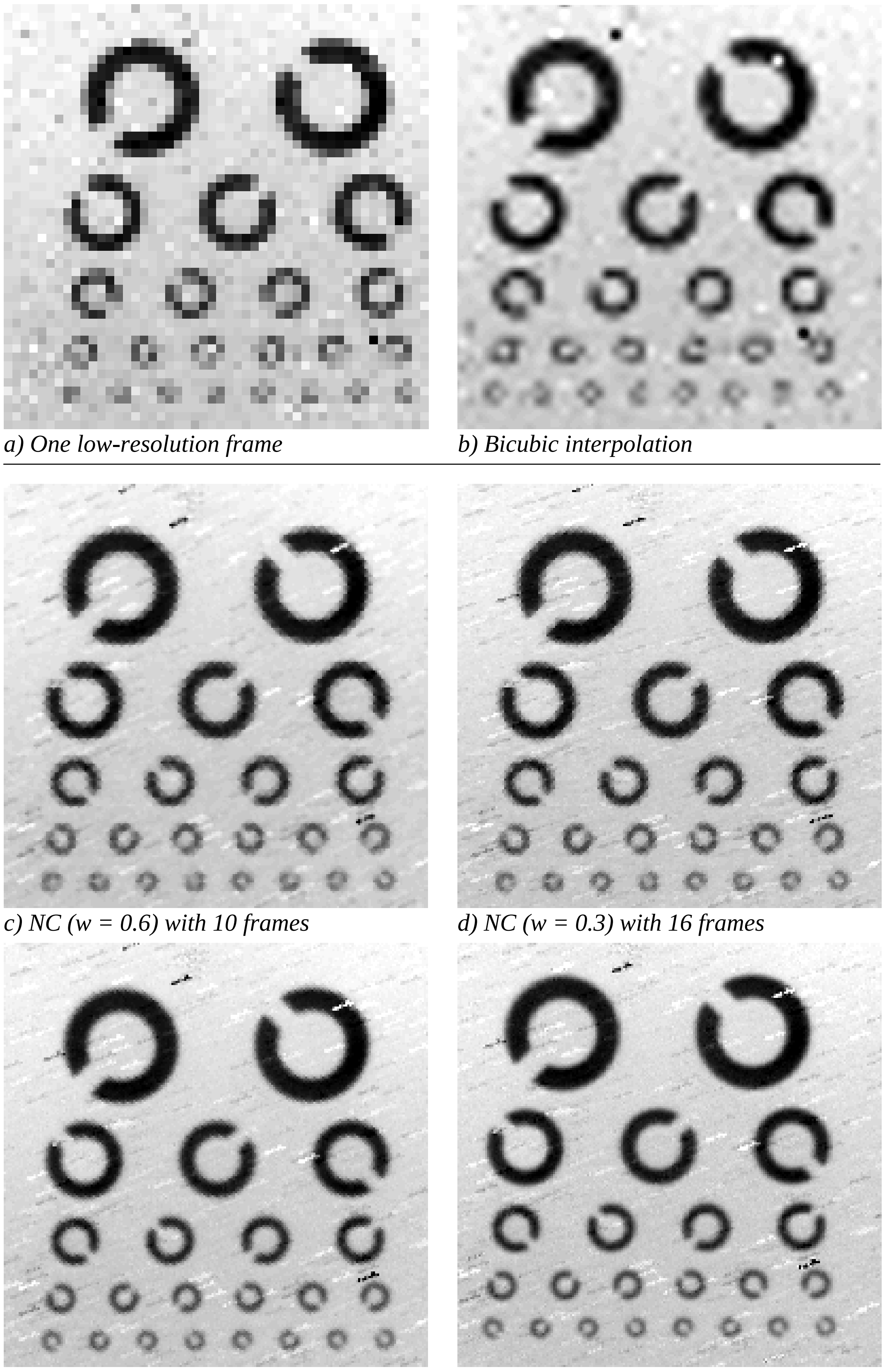


Figure 29: Original, low-resolution, aliased IR frame.

Figure 29 is one of the undersampled input frames we recorded. It contains two perforated targets, in front of a cold plate. The temperature difference is only about 5° C, which means that the SNR is about 42dB (as calculated in section 2.3). At one side we can see a lamp and some other objects. Figure 30 contains the results obtained with that sequence (contrast-stretched to make the differences better visible). The reconstruction was done on the whole image, but we will only show the top target, which contains rings of different sizes and demonstrates the resolution increment obtained. The top two images are of a single frame (one bicubic interpolated), the bottom right image is what we consider the best solution to the problem, calculated using the conjugate gradient method on 25 images. These three images are included to compare the proposed algorithm to. The other three images were reconstructed using gradient-based shift estimation (GRS) and normalized convolution (NC), applied to 10 (with a window parameter *w* of 0.6), 16 and 25 (*w* = 0.3) low-resolution frames. Note that the result acquired by interpolating 16 frames can be improved upon by using more frames. Also note the small difference between the best solution and the result of the proposed algorithm using 25 frames. We must mention here that the shifts in this sequence were poorly distributed (see figure 31); only all 25 frames together correctly fill the area.

Figure 32 shows results with a different target in the same setting. The sequence has a better distribution of shifts. Finally, figure 34 shows results of another sequence, which contains some objects with both higher and lower contrast.

Note (this is especially visible in figure 30) how high-resolution image restoration is also capable of removing some noise. Hot and cold pixels, which were not removed prior to the restoration, become lines in the high-resolution image. This is due to the motion of the camera. As the camera moves, the acquired scene moves too, but the position of these pixels on the sampling grid does not. We use the scene as a reference, so these pixels will move over it, and the interpolation will convert them to lines.



a) One low-resolution frame

b) Bicubic interpolation

c) NC (w = 0.6) with 10 frames

d) NC (w = 0.3) with 16 frames

e) NC (w = 0.3) with 25 frames

f) Conjugate gradient with 25 frames

Figure 30: Portion of original frame, containing the target with the rings. All six images have been contrast-stretched to enhance the details.

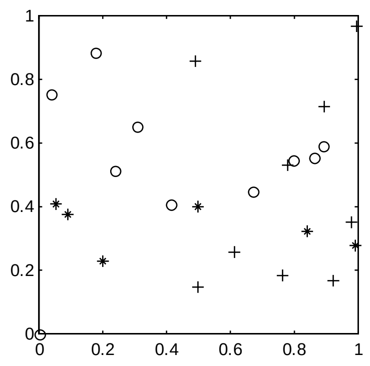
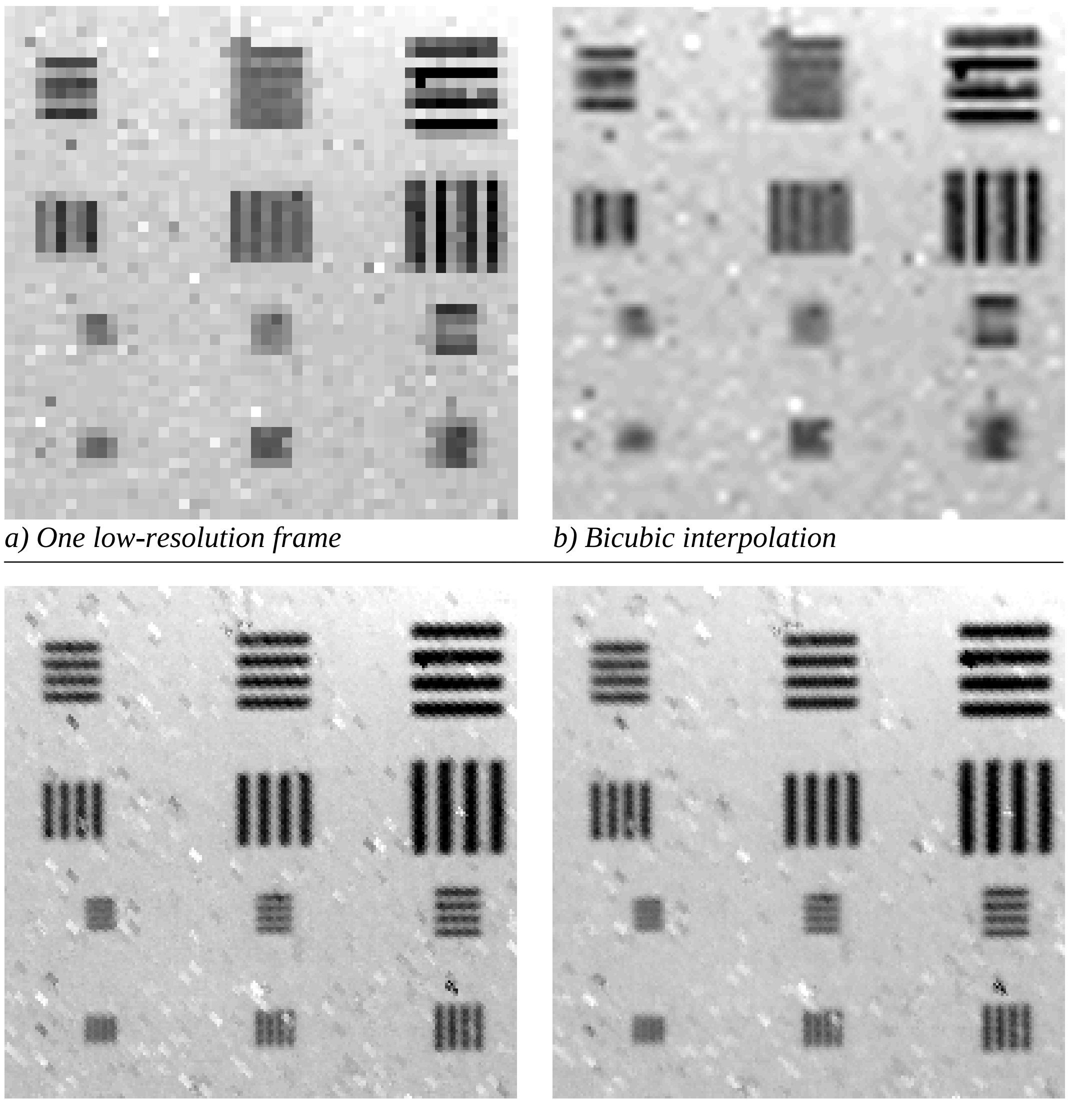


Figure 31: Shifts for the sequence used in figure 30. The circles indicate the shifts for the first 10 frames, the stars add 6 frames, and the pluses are the shifts for the 9 added to make 25 frames. Note the poor distribution of the circles, and how only the plusses make the distribution equilibrated.



a) One low-resolution frame *b) Bicubic interpolation*

c) NC (w = 0.3) with 16 frames *d) LSE (w = 3) with 16 frames*

Figure 32: Portion of the original frame, containing the target with the bar patterns. All four images have been contrast-stretched to enhance the details.

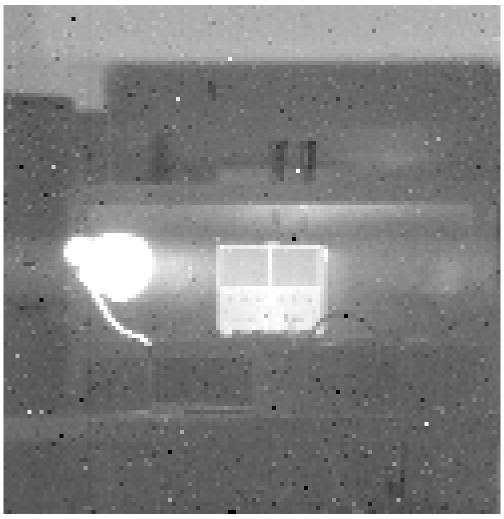


Figure 33: One original frame of a scene, which contains a lamp, which is hot and was clipped, and a transformer with two gauge plates and some buttons and a LED. The image was contrast-stretched to enhance the details.

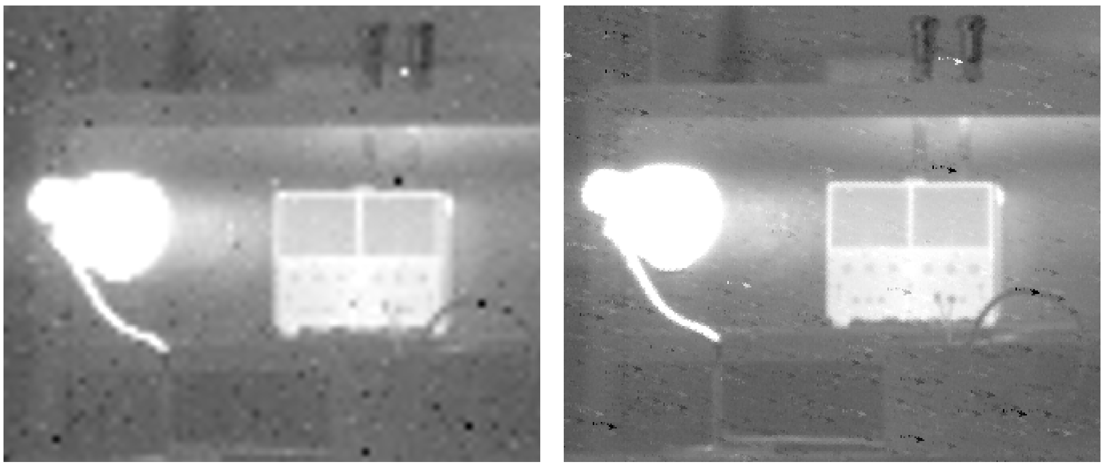


Figure 34, left: Portion of original frame from figure 33, bicubic interpolated to 4 times its size. Right: Same portion of high-resolution image acquired using 16 input frames, GRS and NC with w = 0.3. Edges are sharper and features better distinguishable. Also some new details are visible. Note that (except for the lamp, which was clipped), the objects in the scene do not have a very large temperature difference. The histogram of the image was manipulated (before reconstruction) to enhance the details. This also causes the noise to be enhanced.

6. Conclusion

We have discussed some algorithms that will solve our problem, namely the reconstruction of a high-resolution image out of a sequence of shifted, undersampled (and thus aliased) images, in a way that could be used in real-time. We have seen that two tasks must be completed sequentially; first the shift between the images must be estimated, then the data can be combined into a single high-resolution image.

The first task is accomplished by either an estimation method based on the fast Fourier transform, or one based on the gradient of one of the images. The former method is a few times faster, the latter a few times more accurate. However, they both suffice.

For the second task we proposed two new methods. They are not new themselves, but the application is. The methods we found in the literature either performed worse than our own, or are iterative (iterative methods do not fit our constraints). The two methods we proposed use different interpolation schemes to resample non-uniformly sampled data, with the possibility to reduce noise. One is based on fitting a plane through a number of input samples, the other on normalized convolution. We showed that these interpolation methods are indeed capable of increasing resolution and reducing noise. Both produce comparable results in comparable time.

We discussed that the minimum number of input frames is equal to the increase in resolution, and how important it is to have a good distribution of shifts. We showed that using less frames still gives good results, and how using more frames we can improve the output image.

We saw that our methods perform worse than the iterative method that in the literature is considered the best possible solution (which cannot be used in real-time). However, the results are acceptable, and the algorithms at least one order of magnitude faster. Finally, we applied our methods to sequences of real-world IR images to show that our goals are met.

Some Ideas for Further Research

Although it did not start as such, this project turned out to be a pilot study. Results were promising, and there are many more possibilities to explore. Some things were outside the initial goals of this project, and many more were thought of during this study. Here is a list of ideas we compiled, suitable for a follow-up project.

- We found and compared algorithms that are suitable to be implemented in hardware. How this implementation should actually be done was never a part of this project. However, that is also an interesting challenge. Most algorithms should adapt very well to multiprocessor systems.
- We assumed that the camera movements were slow, that is, we ignored any motion blur that might be present. Removing motion blur before reconstruction could, in some cases, enhance the final high-resolution image.
- We also assumed that the scene did not change from one frame to the next. Adapting the registration algorithms to recognize moving objects and difference in speed between foreground and background objects is also interesting.
- Other transformations we should be able to recognize are rotation and scaling, both locally and globally.
- It would also be useful to have the superresolution algorithm provide a confidence measure of the result. It should, for example, be able to recognize situations where the overlapping portion in the frame sequence is too small, or situations where the motion is not adequate (i.e. the shifts are poorly distributed).
- As stated before, superresolution is also applicable in many other fields, like echography. It could also be possible to get more information out of the images from security cameras, or recover degraded films.
- It is interesting to study other applications, like using these methods to remove noise instead of increasing resolution, increase resolution beyond the capabilities of the imager used, or recover lost samples.
- It is also necessary to study pre-processing of the data. Inter-scene and intra-scene offset and gain inhomogeneity correction, dead pixel elimination, and lighting change correction are some of the possible additions.

References

[1] Alam, M.S., et al., *“High Resolution Infrared Image Reconstruction Using Multiple, Randomly Shifted, Low Resolution, Aliased Frames”*, SPIE Proceedings vol. 3063-9, April 1997.

[2] Bos, A. van den, *“Parameter Estimation”*, Handbook of Measurement Science, vol. 1, chapter 8, John Wiley & Sons Ltd., 1982.

[3] Braat, J.J.M., *“Technishe Optica”* (Dutch), c32a course notes, Faculty of Applied Physics, TU Delft, 1993.

[4] Gillette, C., T.M. Stadtmiller and R. C. Hardie, *“Aliasing Reduction in Staring Infrared Imagers Utilizing Subpixel Techniques”*, Optical Engineering, vol. 34, no. 11, pp. 3130-3137, November 1995.

[5] Hardie, R.C., et al., *“High Resolution Image Reconstruction From a Sequence of Rotated and Translated Infrared Images*”, SPIE Proceedings vol. 3063-10, April 1997.

[6] Hardie, R.C., K.J. Barnard and E.E. Armstrong, *“Joint MAP Registration and High Resolution Image Estimation Using a Sequence of Undersampled Images”*, IEEE Transactions on Image Processing, vol. 6, no. 12, pp. 1621-1633, December 1997.

[7] Kaltenbacher, E.A., and R.C. Hardie, *“High Resolution Infrared Image Reconstruction Using Multiple, Low Resolution, Aliased Frames”*, SPIE Proceedings, vol. 2751, pp. 142-152, 1996.

[8] Knutsson, H., and C.F. Westin, *“Normalized Convolution: A Technique for Filtering Incomplete and Uncertain Data”*, SCIA’93, Proceedings of the 8th Scandinavian Conference on Image Analysis, vol. 2, pp. 997-1006, 1993.

[9] Marvasti, F., *“Nonuniform Sampling Theorems for Bandpass Signals at or Below the Nyquist Density”*, IEEE Transactions on Signal Processing, vol. 44, no. 3, pp. 572-576, March 1996.

[10] Netten, H. *“Exact Reconstruction of Sampled Images”*, Graduation Report, Pattern Recognition Group, Faculty of Applied Sciences, TU Delft, August 1990.

[11] Oppenheim, A.V., A.S. Willsky and I.T. Young, *“Signals and Systems”*, Prentice-Hall International, Inc., 1983.

[12] Patti, A.J., M. Ibrahim Sezan and A. Murat Tekalp, *“Superresolution Video Reconstruction with Arbitrary Sampling Lattices and Nonzero Aperture Time”*, IEEE Transactions on Image Processing, vol. 6, no. 8, pp. 1064-1076, August 1997.

[13] Priestley, M.B., *“Spectral Analysis and Time Series”*, Academic Press Ltd., 1981.

[14] Schultz, R.R., and R.L. Stevenson, *“Extraction of High Resolution Frames from Video Sequences”*, IEEE Transactions on Image Processing, vol. 5, no. 6, pp. 996-1011, June 1996.

[15] Schutte, K., TNO-FEL, private communications, 1998.

[16] Shewchuk, J.R., *“An Introduction to the Conjugate Gradient Method Without the Agonizing Pain”*, ftp://warp.cs.cmu.edu/quake-papers/painless-conjugate-gradient.ps, August 1994.

[17] Simonis, J., *“Lineaire Algebra”* (Dutch), a31-Et course notes, Faculty of Applied Mathematics and Computer Science, TU Delft, 1993.

[18] Srinivasa Reddy, B., and B.N. Chatterji, *“An FFT-Based Technique for Translation, Rotation, and Scale-Invariant Image Registration”*, IEEE Transactions on Image Processing, vol. 5, no. 8, pp. 1266-1271, August 1996.

[19] Vliet, L.J. van, F.R. Boddeke, D. Sudar and I.T. Young, *“Image Detectors for Digital Image Microscopy”*, Digital Image Analysis of Microbes, chapter 3, Wiley and Sons, Ltd.

[20] Young, I.T., J.J. Gerbrands and L.J. van Vliet, *“Fundamentals of Image Processing”*, Faculty of Applied Sciences, TU Delft, 1997.

Appendix A

Determination of the Nyquist frequency of a Gaussian function

The Gaussian function,

$$g_\sigma(t) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{\frac{-t^2}{2\sigma^2}} \quad , \tag{43}$$

with Fourier transform

$$G_\sigma(\omega) = e^{\frac{-\omega^2\sigma^2}{2}} \quad , \tag{44}$$

does not have a cutoff frequency ω_c. We will define ω'_c, an approximation to the cutoff frequency. Let us neglect aliasing of frequencies where the frequency spectrum is less than 1% of its maximum height (as in figure 35). The cutoff frequency ω'_c is then defined by

$$\left|G_\sigma(\omega)\right| < \frac{1}{100}\max\left\{\left|G_\sigma(\omega)\right|\right\} \quad , \quad \forall|\omega| > \omega'_c \quad . \tag{45}$$

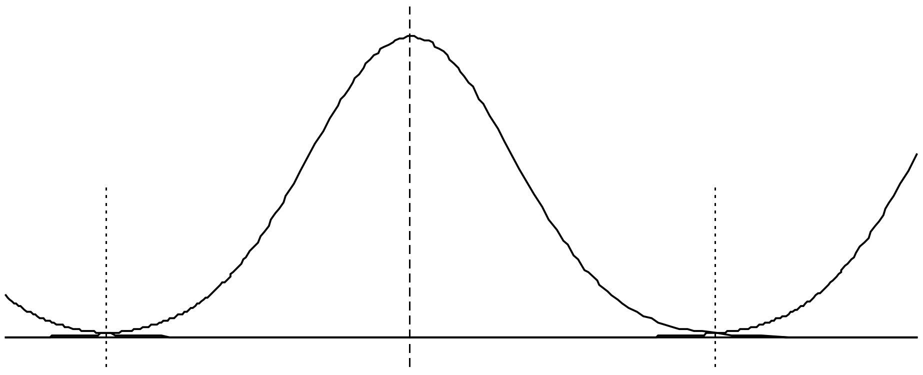


Figure 35: Frequency spectrum of a Gaussian function sampled at σ.

This leads to a cutoff frequency ω'_c equal to $3\sigma^{-1}$. The sampling frequency ω_s should be larger than twice the cutoff frequency,

$$\omega_s > 2\omega'_c \quad , \tag{46}$$

and the distance between samples T is

$$\frac{2\pi}{T} > 2\,\frac{3}{\sigma} \quad \Rightarrow \quad T < \frac{2\pi}{6}\,\sigma \approx \sigma \quad . \tag{47}$$

Appendix B

Aliasing in the discrete L^p-norm

The L^p-norm for the measure of the difference between two functions,

$$\mathrm{L}^{\mathrm{p}} = \left(\frac{1}{2L} \int_{-L}^{+L} \left| f(x) - g(x) \right|^p dx \right)^{1/p} \quad , \tag{48}$$

can be made discrete by sampling the functions being compared. The result reads

$$\mathrm{L}^{\mathrm{p}}_{\mathrm{N}} = \left(\frac{1}{2N} \sum_{n=-N}^{+N} \left| f[n] - g[n] \right|^p \right)^{1/p} \quad , \tag{49}$$

supposing that both $f[n]$ and $g[n]$ are sampled at the Nyquist rate. Let us consider $p = 2$.

Let $y(x) = f(x) - g(x)$, and set the integration limits to infinity. Knowing that

$$\int_{-\infty}^{+\infty} y(x) dx = \int_{-\infty}^{+\infty} y(x)\, e^{-j\omega x}\, dx \Bigg|_{\omega=0} = \mathfrak{F}\big(y(x)\big)\Big|_{\omega=0} \quad , \tag{50}$$

we can see that

$$\int_{-\infty}^{+\infty} y^2(x) dx = \mathfrak{F}\big(y^2(x)\big)\Big|_{\omega=0} = \mathfrak{F}\big(y(x)\big) * \mathfrak{F}\big(y(x)\big)\Big|_{\omega=0} \quad . \tag{51}$$

If the frequency spectrum of $y(x)$ is bandlimited, with $Y(\omega) = 0$ for $|\omega| \geq \mathrm{W}$, $y^2(x)$ will be bandlimited to 2W (see figure 36).

After sampling at the Nyquist rate for $y(x)$ (sampling period $\mathrm{T} = \pi/\mathrm{W}$), $y^2(x)$ will be completely aliased, except at its ground frequency, $\omega = 0$. The L^2-norm is comparable to $(\mathrm{Y}*\mathrm{Y})_{(\omega = 0)}$, and thus also undisturbed by the aliasing. Hence, L^2 is sampled according to Nyquist if $y(x)$ is.

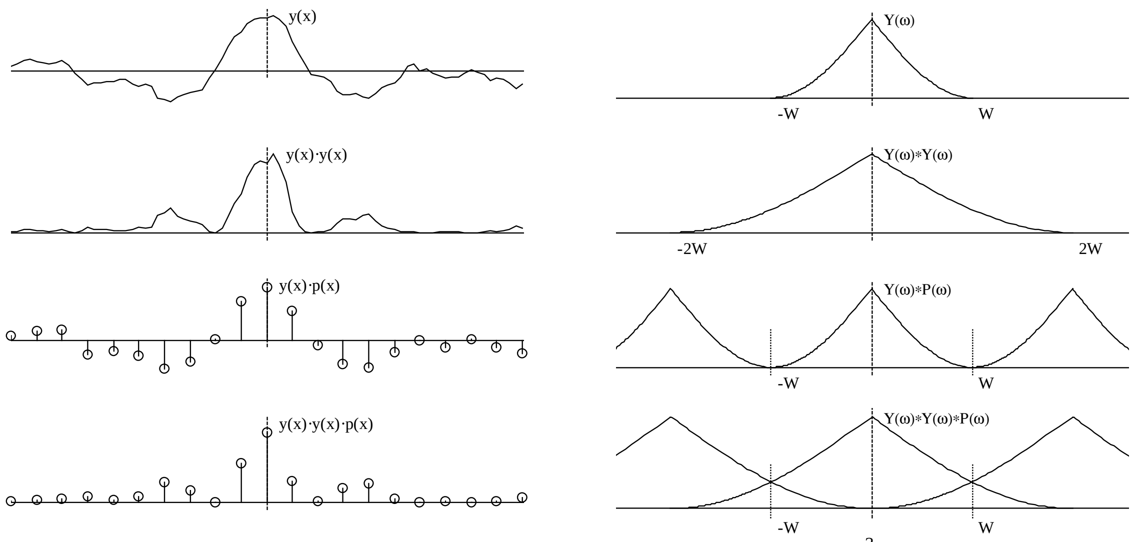


Figure 36: Example for a frequency spectrum of a function y(x) and $y^2(x)$, and the discrete versions, sampled at the Nyquist rate.