ON RE-SAMPLING OF SOLAR IMAGES

C. E. DeFOREST
Southwest Research Institute, Boulder, CO, U.S.A.
(e-mail: deforest@boulder.swri.edu)

(Received 14 August; accepted 18 September 2003)

Abstract. Digital image data are now commonly used throughout the field of solar physics. Many steps of image data analysis, including image co-alignment, perspective reprojection of the solar surface, and compensation for solar rotation, require re-sampling original telescope image data under a distorting coordinate transformation. The most common image re-sampling methods introduce significant, unnecessary flaws into the data. More correct techniques have been known in the computer graphics community for some time but remain little known within the solar community and hence deserve further presentation. Furthermore, image distortion under specialized coordinate transformations is a powerful analysis technique with applications well beyond image resizing and perspective compensation. Here I give a brief overview of the mathematics of data re-sampling under arbitrary distortions, present a simple algorithm for optimized re-sampling, give some examples of distortion as an analysis tool, and introduce scientific image distortion software that is freely available over the Internet.

"First get your facts straight. Then you can distort them as you please." – Mark Twain

1. Introduction

Coordinate transformations and data re-sampling are central to modern solar data analysis. Linear coordinate transformations are used to co-align simultaneous digital images taken with different telescopes; the family of perspective map projection transformations are used to generate maps of the solar surface layers, as in running synoptic maps (Hoeksema *et al.*, 2000); to remove solar rotation or other perspective effects from images of the solar surface; or to highlight particular aspects of the data, such as super-radial expansion of a coronal hole (DeForest *et al.*, 1997).

While the theory and techniques of data re-sampling are well-known parts of the computer science curriculum (Heckbert, 1989; Foley *et al.*, 1996), there is not a strong awareness in the solar data analysis community of either the advantages or pitfalls of image distortion and data re-sampling. As a result, image data are often analyzed less well than they could be in an appropriate coordinate system; and, when existing re-sampling tools are used (as for solar rotation compensation), unneeded spatial artifacts and photometric errors are introduced into the data. These types of systematic errors, while avoidable, are common enough in the current literature that it is not worthwhile to single out particular examples.

Solar Physics **219:** 3–23, 2004.

In the computer graphics field, the subject of coordinate transformation re-sampling is called 'texture mapping' (Heckbert, 1989), and the emphasis of current work is on improving and speeding implementation of the specific class of perspective transformations used to render a 3-D scene. Very clear overviews of the mathematics and theory of re-sampling, texture mapping, and spatial filtering may be found in Heckbert (1989), Turkowski (1990), and Turkowski (1993).

Software to perform specific coordinate transformations on solar data is widely used. The SolarSoft distribution tree (Freeland and Handy, 1998) contains several popular re-sampling tools for specific coordinate transformations. Those modules include many piecemeal subroutines, the `ZTOOLS` (DeForest, 1998) and `PLOT_MAP` (Zarro, 1998) utility packages, and high-level graphical codes such as `IMAGE_TOOL` (Wang, 1994). Other specialized tools exist in the SOHO/MDI (Scherrer *et al.*, 1995) and GONG (Harvey *et al.*, 1996) data analysis pipelines, for example to apply particular map projections to solar oscillation data (Bogart *et al.*, 1995).

General-purpose image re-sampling under nonlinear coordinate transformations is less commonly used but is supported by several available software packages. For example, NRAO's AIPS (Greisen, 2003) is intended for analysis of radio astronomy data, but includes image re-sampling code; ISIS is a remote-sensing cartographic transformation package distributed and maintained by the United States Geological Survey (Torson and Becker, 1997); and Rutherford Laboratory's STARLINK software library (Berry, 2001) includes generic coordinate transformation code written in C (under the module name AST).

Most existing re-sampling software uses direct interpolation or sampling of an input data array to produce distorted output data. But the mere fact of popularity does not indicate that interpolation is the best, or even an especially good, way to re-sample images. Quite the contrary, Section 2 of this article shows why direct interpolation is not a good approach. Further, Section 3 describes a sampling algorithm that overcomes the limitations of local interpolation without excessively inefficient use of computing resources. Section 4 introduces `PDL:Transform`, a software toolkit that implements optimized re-sampling and provides a natural interface for manipulating generic coordinate transformations; and Section 5 demonstrates two applications of specialized image distortion to visualize particular aspects of solar image data.

2. The Mathematics of Image Re-sampling

A solar image or other dataset is a collection of pixel values that are defined on a regular grid. It is a representation of a more general construct: a mapping D from N independent variables (such as focal plane spatial coordinates, wavelength, or time) to M separate components (such as brightness in independent wavelength passbands, different physical parameters, or spatial components of a vector parameter):

$$D_{C_1} : \mathbb{R}^N_{C_1} \rightarrow \mathbb{R}^M, \tag{1}$$

where C_1 is the original coordinate system of the data and $\mathbb{R}^N$ is the space of N-vectors with real-valued components.

One has a coordinate transformation C that converts vectors in the input coordinate system C_1 to an output coordinate system C_2:

$$C : \mathbb{R}^N_{C_1} \rightarrow \mathbb{R}^N_{C_2}. \tag{2}$$

The idea is to create another dataset, D_{C_2}, such that

$$D_{C_2}\left(C\left(x_i\right)\right) = D_{C_1}(x_i), \tag{3}$$

where i is a dimensional index and x_i is an input-plane coordinate. Letting X_i be the output-plane coordinate $C(x_i)$ one may write:

$$D_{C_2}(X_i) = D_{C_1}\left(C^{-1}\left(X_i\right)\right). \tag{4}$$

The mappings D_{C_1} and D_{C_2} are the images that are represented digitally with arrays of pixel values. Equation (4) is useful because it describes the output data values in closed form, given the input data values and coordinate transform.

In the remainder of this article I will implicitly set $N = 2$ and refer to 'images' and image re-sampling; but the techniques are applicable in the general higher-dimensional case.

2.1. Sampling, interpolation, and aliasing

In practice, digital images aren't infinitely detailed mappings from $\mathbb{R}^N \rightarrow \mathbb{R}^M$. The data exist only at locations with integer pixel indices:

$$D_{C_1,\mathbb{Z}} : \mathbb{Z}^N_{C_1} \rightarrow \mathbb{R}^M, \tag{5}$$

where $D_{C_1,\mathbb{Z}}$ is a digital data array such as the data portion of a FITS file, and $\mathbb{Z}$ represents the integers. Producing the re-sampled image $D_{C_2,\mathbb{Z}}$ involves looping over all the pixel locations $X_i \in \mathbb{Z}^N$ in the output image, calculating the associated $x_i = C^{-1}\left(X_i\right)$, and determining the appropriate pixel value from $D_{C_1,\mathbb{Z}}$. It is this last step of finding the output pixel value that needs improvement in common usage.

Figure 1 shows an example of a coordinate transform in image space: the conversion of a SOHO/EIT image (Delaboudinière *et al.*, 1995) to a plate caree ('longitude/latitude') map of the Sun. In this case, the coordinate transformation C^{-1} is the true-perspective projection mapping from longitude and latitude to focal plane coordinates. The mapping is fully determined by *a priori* knowledge of the SOHO orbit and the assumption that the Sun is spherical[1]. The output plane is a plate caree

[1]For very accurate positioning of surface features, additional information is also needed about the optics in the telescope: treating a compound telescope as a pinhole camera yields inaccuracies due to the additional tan (θ) distortions imposed by magnification in the optics. Other types of optical distortion enter at higher order.

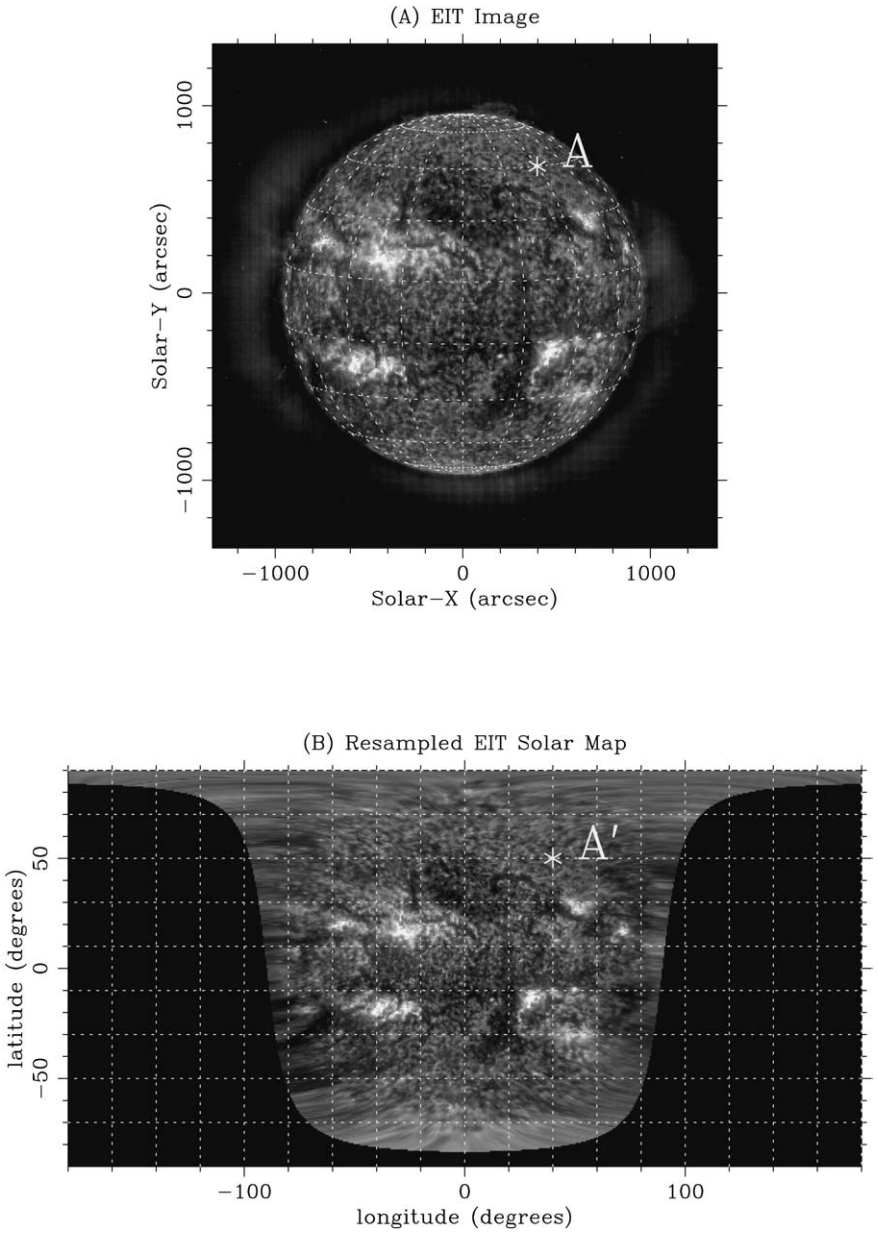

Figure 1. Mapping between an EIT solar image and the derived solar map. A perspective transformation maps locations on the EIT image to associated locations on the solar map (A'). Following common practice, each output pixel value of this image was interpolated directly from the corresponding input plane location. While the result looks convincing, it contains systematic errors due to the sampling.

(longitude/latitude) map of the solar surface; the input plane is the focal plane of a solar telescope. The input plane coordinates are marked in 'arc sec', but are more properly described as 'tangent arc sec', as flat focal plane coordinates correspond to the tangent of incident angle. The values of the image at the right-hand side were calculated by transforming each pixel coordinate (such as A') via C^{-1} back into input image plane locations, and interpolating from the nearby input pixel values in the left-hand image (such as at A).

Interpolation is only half of the story. Figure 2 shows visually the relationship between geometry and re-sampling for a coordinate transformation C_P that is described by a high-order bivariate polynomial in x_i. Figures 2(A) and 2(B) show the relationship between the input pixels (grid) and the output pixels (marked with '$\times$' at each pixel center), in both the input plane and the output plane. The transformation acts on a 21×21 pixel image that is shown in the input (x_i) plane in Figure 2(C). Figure 2(D) contains a transformed image, produced by bi-linearly interpolating the values from integer x_i locations at the '$\times$' loci in Figure 2(A).

At the left side of each panel of Figure 2, the image is being magnified, and each input plane pixel is sampled by many output plane pixels. This is visible in Figure 2(A) as a dense cloud of sample loci; in Figure 2(B) as large pixels in the distorted input grid; and in Figure 2(D) as the large, blurred image of the vertical bar at the left of Figure 2(C).

At the right side of each panel of Figure 2, the image is being reduced[2], and only a few input pixels are sampled. This is visible in Figure 2(A) as a scarcity of sample loci; in Figure 2(B) as a region where the distorted grid squares are smaller than output pixels; and in Figure 2(D) as a noisy, aliased region at the right side of the re-sampled image.

The reduced area of the image in Figure 2(D) is noisy and incoherent because of the large spaces between the pixel samples shown in Figure 2(A). The spaces between individual samples are larger than the input pixel spacing, so that the image is only sporadically sampled. The white pixel in the center of the upper right-hand quadrant is not sampled at all, much of the lower and middle horizontal bars are missing, and the vertical bar at the right side of the image is reduced to 2 bright pixels (out of its ideal length of 9 pixels in the re-sampled image).

The lowest portion of the image shows that the aliasing problem holds wherever the image is reduced in any direction, not only where the reduction is uniform. Here, the input pixels are magnified in one diagonal direction and reduced in the perpendicular direction, so that only four output pixels are significantly influenced by the bottom edge of the input image. (They are at X coordinates (11,0), (12,1), (13,2), and (16,6); and approximate x coordinates (5.3,0.5), (6.2,-0.3), (7.1,-0.6), and (8.9,0.8), respectively.)

This type of aliasing exists whenever local sampling (including interpolation) is used for image reduction: the input image is *under-sampled*, aliasing high spa-

[2]In graphics literature one often finds 'decimated' rather than 'reduced'.

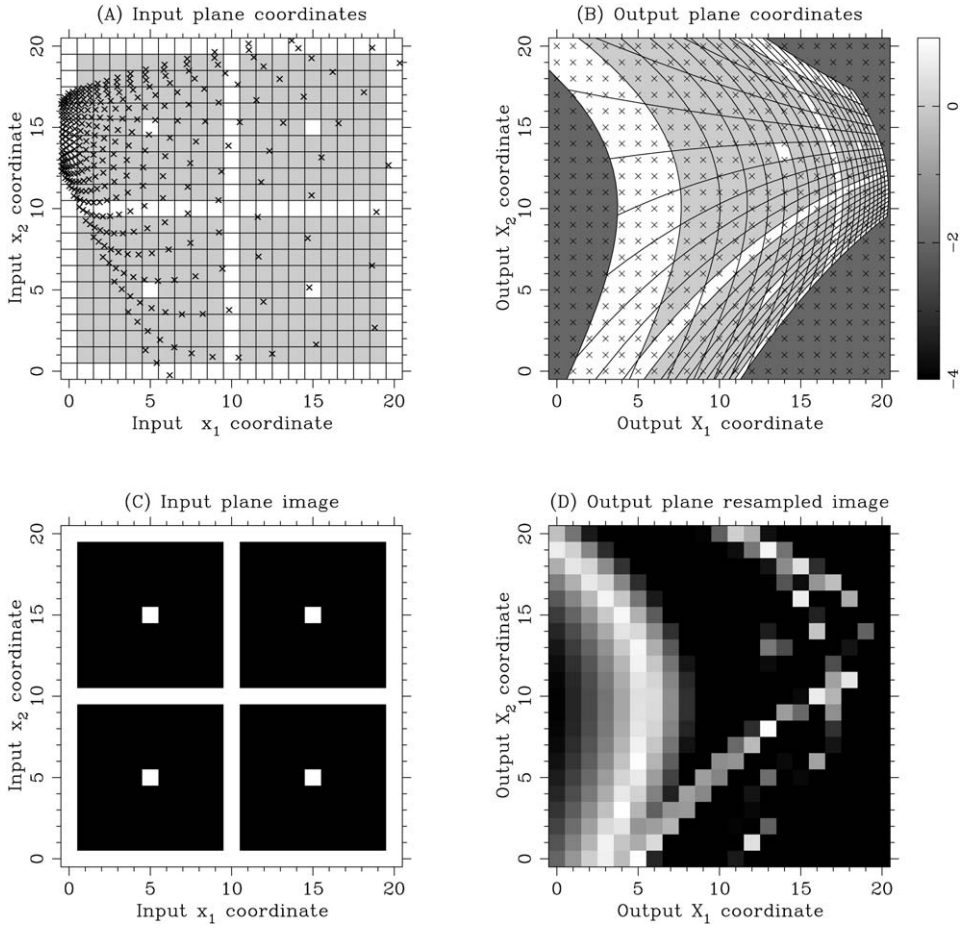

Figure 2. Map of a sample polynomial coordinate transform, showing the effects of local image magnification, stretching, and reduction with simple sampling. (A) The input plane, showing the irregular grid of locations ('×') from which output values are interpolated in the x_i plane. (B) The output plane, showing the regular spacing of output pixel centers ('×') and distorted input grid in the X_i plane. (C) A sample image in the original input plane. (D) Bi-linearly interpolated re-sampled data, showing extreme aliasing and loss of photometry near the it bottom and right-side of the figure.

tial frequencies from above the new Nyquist frequency down into the represented spatial frequency range for the new image. In typical cases, photometric noise and errors in the shape and size of small features are introduced into the output image. In extreme cases, entire features may be missed or exaggerated more than tenfold in strength.

2.2. THE FOOTPRINT OF A PIXEL, AND LOCAL LINEARIZATION

There are several strategies for overcoming the under-sampling problem. Unfortunately the two most obvious and widely-used strategies to mitigate aliasing are neither particularly accurate nor particularly efficient.

The most obvious technique is to oversample the output plane by a factor m, and then apply $m \times m$ pixel summing to get it down to the desired dimension. The problem is that oversampling by a linear factor of L in N dimensions increases the computing load by $O(L^N)$, and is easily overwhelmed by transformations that are even mildly pathological: aliasing will occur unless the output oversampling factor exceeds the highest reduction ratio anywhere in the image plane. The technique is marginally feasible for image data ($N = 2$), but horrifically bad in higher dimensions. Removing the aliasing from all points of Figure 2 would require 5×5 oversampling, at a factor-of-25 computational cost. In three dimensions, similar oversampling would cost more than a factor of 100 in computing load.

Another common approach is to reduce the high spatial frequencies in the input plane by uniform smoothing, which reduces aliasing at low CPU cost but shifts the heavy $O(L^N)$ burden to unnecessary loss of resolution in the data. Smoothing the input plane of Figure 2 with a boxcar 5×5 kernel would remove the spatial aliasing but would spread the $x_1 = 0$ white vertical bar to the entire left half of the X_i image.

What is needed is a spatially variable filter that is keyed to the characteristics of the transformation, providing spatial averaging where necessary and direct sampling where appropriate (Turkowski, 1990). The function C^{-1} itself contains the required information to build such a filter. In particular, each output pixel has a 'footprint' in the input plane, and the shape of that footprint should determine the mix of averaging and interpolation that is used to calculate the value of the output pixel.

In general, finding the transformed shape of a pixel under a nonlinear transformation is computationally difficult and analytically intractable. Fortunately, linear transformations are quite tractable and essentially all useful transformations can be locally linearized. Taylor expanding around the point x_0 gives:

$$x_i \simeq C^{-1}_{L,X_0}(X_i) = x_{0,i} + J^{-1}_{0,ij}(X_j - X_{0,i}) + \cdots , \tag{6}$$

where j is an implicit summation index and the matrix $J^{-1}_{0,ij}$ is the Jacobian derivative matrix of the inverse transformation at X_0:

$$J^{-1}_{0,ij} = \left[\frac{\mathrm{d}x_i}{\mathrm{d}X_j} \right]_{X=X_0} . \tag{7}$$

Linearizing C about each pixel center brings *singular value decomposition* to bear on the problem. Every square matrix J_{ij} can be resolved into a rotation matrix A_{kj} that rotates column vectors into J_{ij}'s singular value basis, a diagonal scaling matrix S_{lk} of singular values; and a second rotation matrix B_{il} that rotates the singular value basis into the output space. One writes:

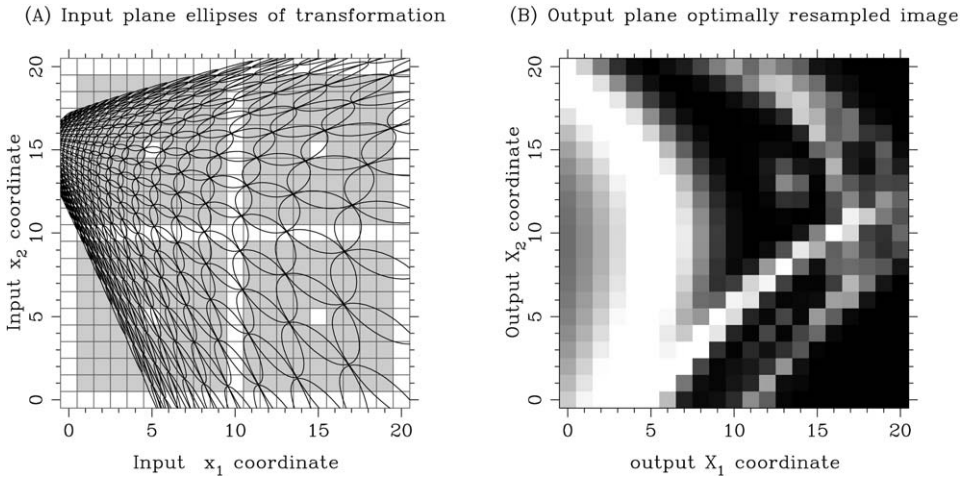

Figure 3. (A) Pixel footprints for the transformation shown in Figure 2. The ellipses are inverse-transformed circles centered on each output-plane pixel. (B) The output plane, with the image re-sampled by averaging over the input-plane footprint of each pixel. Photometry and spatial structure are much better preserved than in Figure 2(D).

$$J_{ij}^{-1} = B_{il} S_{lk} A_{kj}. \tag{8}$$

where repeated indices are again meant to be summed over.

Applying $J_{0,ij}^{-1}$ to each vector on a unit circle centered on the origin yields an ellipse[3]. The major and minor axes of the ellipse are the singular values of J^{-1}, expressed as the diagonal elements of S_{lk}. The second rotation matrix B_{il} determines the orientation of the ellipse. This *ellipse of transformation* of J_0^{-1} is useful because, when translated by the additive terms in Equation (6), it approximates the locus in input space that is mapped to the output pixel X_0 by C; and it can be used as the basis of more exact approximations (such as skewed versions of original square pixels).

Figure 3 shows a variation on Figures 2(A) and 2(D) showing the ellipse of transformation of each output pixel's local linearization of C^{-1}. The ellipses were produced from circles with radius 0.7, close to the diagonal radius of each pixel. Re-sampling the image by averaging over each footprint preserves the average surface-brightness photometry and morphology even on the right-hand side of the panel, where simple interpolation fails. Figure 3(B) is directly comparable to Figure 2(D), showing vast improvement compared to sparse interpolation.)

The simplest way to produce output pixel values using the linearized footprints is to average together all of the input data points whose pixel centers lie within the elliptical footprint of each output pixel. However, this naive scheme fails where J^{-1} has small singular values, because the footprint may be small enough not to intersect any pixel centers at all. This happens, for example, at the left side of

[3]In higher dimensions, an ellipsoid.

Figure 3(A). Hence, the footprints used for averaging pixel center values must be constructed not from J^{-1}, but from a padded version J_P^{-1} whose A and B matrices are the same as those of J^{-1} but whose S matrix values are padded to a minimum value of at least unity.

Averaging over each footprint applies a spatial low-pass filter to each neighborhood in the image, so that the overall image is subjected to a spatially variable low-pass filter. The theory of spatially variable filters for re-sampling is covered by Turkowski (1990). The ideal re-sampling filter is a local convolution of anti-aliasing filters in the input and output planes; padding the ellipses of transformation approximates the convolution operation.

In practice, one uses an analytic filter function to weight the pixel values in each footprint. An isotropic Gaussian filter whose tails extend beyond the ellipse boundaries is ideal for critically sampled data with information up to the spatial Nyquist frequency of the image and an isotropic frequency spectrum. Broader filter functions blur the output image; excessively narrow ones allow aliasing. The optimum balance occurs with a full-width at half maximum of about $\sqrt{2}$, the maximum diameter of a square pixel.

In cases where the image frequency spectrum follows the sampling frequency (for an n-dimensional square grid, the sampling frequency varies by factor of $\sqrt{n}$ between grid-aligned and grid-diagonal directions), an anisotropic filter better reproduces the original data. One uses a filter that is the product of individual one-dimensional filter functions along each dimension – a Hanning window ($\sin^2$ roll off) in each dimension is a robust choice that preserves pixel locality while not impacting the image frequency spectrum too much.

3. Singular-Value Padding: a Better Re-sampling Algorithm

Here is an algorithmic recipe for carrying out optimized re-sampling of a single pixel under an arbitrary coordinate transform in n dimensions, using the padded ellipse of transformation to approximate the input sampling area for each output pixel. Looping, vectorizing, or parallelizing are language-dependent and not presented here.

The algorithm requires an input array `$A[`$N_{A,0}, \ldots, N_{A,n-1}$`]`, an input X-space vector `$ox`, and a subroutine `Ci()` that accepts a vector in X space and returns the associated inverse-mapped vector in x space. In the typical case, $n = 2$ and the data are images; but the technique works for higher dimensions as well. `Ci` is assumed to work in index (pixel) coordinates in the input and output arrays. In practice, `Ci` is typically the composition of a main transform in physical space and two interface transforms that convert between physical coordinates and array index coordinates.

3.1. FIND THE JACOBIAN OF C^{-1}

Allocate the inverse Jacobian matrix: an $n \times n$ array `$Ji[n,n]`. Looping over `$i`, set each column of `$Ji` to the symmetric discrete derivative with respect to X_i of $C^{-1}(X)$:

```
for ($i=0; $i < $n; $i++) {
   $delta = zeroes( $n );
   $delta( $i ) += 0.5;
   $Ji( $i, : ) .= Ci( $ox - $delta ) - Ci( $ox + $delta );
}
```

3.2. MANIPULATE THE JACOBIAN'S SINGULAR VALUES

Singular-value decompose `$Ji` into its components `$Ra`, `$S`, and `$Rb`. (see, e.g., Press *et al.*, 1986 for singular value decomposition algorithms).

Pad the singular values of J^{-1} to at least unity. Using the rotation matrices and the padded singular values, re-constitute the modified Jacobian back into `$Ji`, and also assemble its inverse `$J`:

```
($Rb, $S, $Ra) = svd($Ji);
$S->diagonal .= maximum( $S->diagonal, 1 );
$Ji = $Rb x $S x $Ra; ## 'x' is matrix multiplication here.
$Si = $S;
$Si->diagonal .= 1.0 / $Si->diagonal;
$J = transpose($Rb x $Si x $Ra);
```

3.3. CALCULATE THE FILTER-WEIGHTED SUM

Find the maximum singular value of `$Ji`: `$Smax = maximum($S)`. Find the input location `$ix = Ci( $ox )`.

Loop over input pixels within a square 4∗`$Smax` on a side, approximately centered on `$ix`. At each integer pixel location `$px`, calculate the offset vector `$of = $px - $ix`. The weighting value for that pixel is then `filt( $J x $of )`. Accumulate and sum weighted pixel values over the whole square.

The filter function `filt()` accepts a vector and returns a weighting value. Two obvious choices of filter function are: (A) a circularly symmetric Gaussian, which is useful for images with isotropic frequency spectra despite the anisotropy of the square grid:

$$\mathrm{filt}_g(\Delta X) = \exp\left(-\sum_i 2b\Delta X_i^2\right), \tag{9}$$

where b is a blurring parameter whose normal value would be 1; and (B) an independent Hanning function in each dimension, which approximately preserves the shape of the original pixels:

$$\mathrm{filt}_h(\Delta X) = \prod_i \left[\frac{(\cos(\min(|\Delta X_i|, 1)\pi) + 1)}{2} \right]. \tag{10}$$

The filter function should be normalized to a total integral of unity over all of X_i space.

It is likely (as in the transformation in Figures 2 and 3) that some coordinates will be mapped outside the domain of the input data array, so some care is required in handling out-of-bounds samples. For example, some transforms that involve spherical coordinates require periodic boundary conditions on the longitude axis and truncation on the latitude axis.

If you want to conserve flux, finish by multiplying the final sum by the determinant of the *unpadded* inverse Jacobian, present in step 2.

3.4. Optimizations

Several optimizations to the basic algorithm are immediately apparent. Here are a few:

Filter shapes: the spatial filtering is accomplished by grabbing a large, square chunk of the input array and multiplying by a weighting function. When the ellipse of transformation is particularly eccentric, many pixels (in the region where the filter function has a negligible value) are referenced unnecessarily. More careful manipulation of the sample shape significantly speeds processing when the transformation has high levels of geometric distortion.

Pre-scaling: in cases where a large part of the image is reduced, uniformly reducing the original image by an integer factor can yield significant speed gain, because $N \times N$ pixel binning is much faster than the weighted-average scheme described here.

Vectorization: in interpreted, vectorizing languages such as PDL or IDL, it is best to implement each step for all pixels in parallel, to minimize use of the interpreter; but this is not the ideal way to use a computer with high-speed CPU cache and slower main RAM (which is the norm for current desktop workstations). Low-level compiled implementations that perform all of the steps in order for each pixel are likely to run significantly faster than comparable implementations in stepwise (vectorized) order, because the pixel-wise order preserves the fast CPU cache whereas the stepwise order generally requires access to slower RAM.

3.5. Photometric comparison

Here is a comparison of optimized and interpolated re-sampling for a typical solar case, differential rotation of a solar image. After linear transformation to co-align

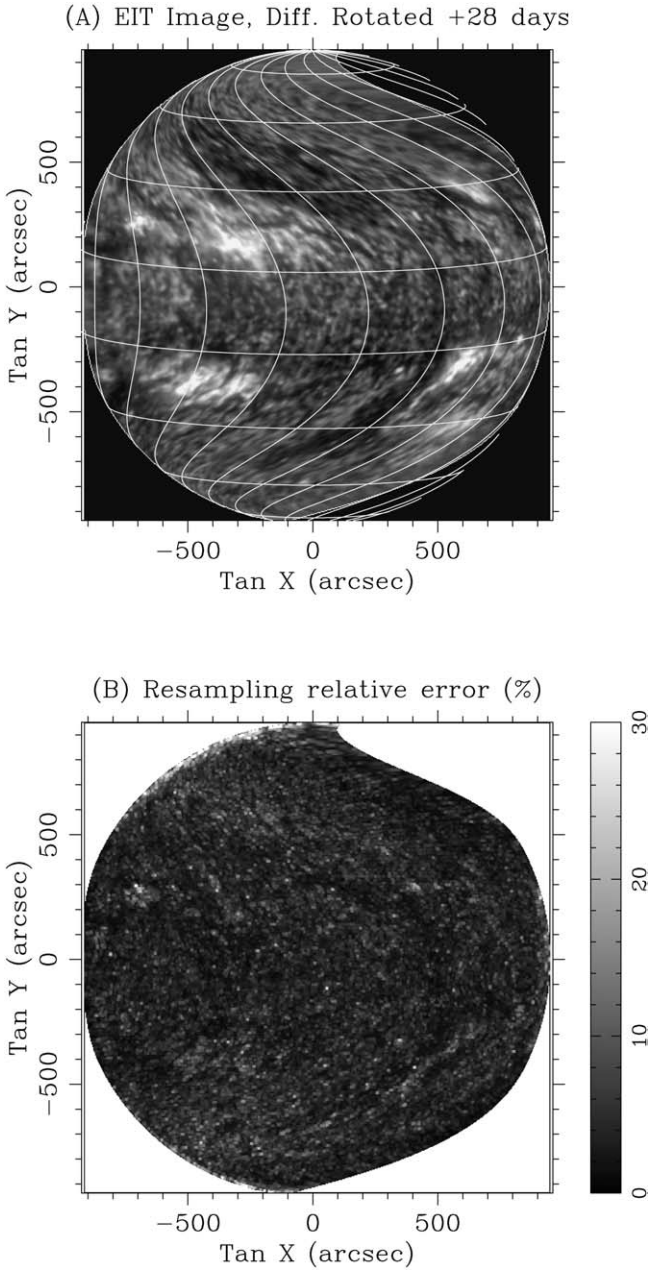

Figure 4. The EIT image of Figure 1, differentially rotated forward 28 days, to illustrate the difference between interpolated and optimized re-sampling. (A) The rotated EIT image, with optimized re-sampling. The long/lat. grid has been warped as well, to illustrate the amount of distortion. (B) Proportional error between optimized and linear re-sampling. About 5% of pixels show errors of 5% or more. Errors are most pronounced near small structures and regions that are sheared or reduced.

images, differential rotation compensation is perhaps the most common application of solar data re-sampling. De-rotation includes shearing and compression aspects, and so is particularly susceptible to aliasing. Figure 4 shows the benefit of optimized re-sampling.

At the top of Figure 4, The EIT image in Figure 1 has been advanced 28 days using the solar differential rotation curve from Cox (2000). The warped longitude/latitude grid indicates the degree of distortion in just one solar rotation. At bottom, the absolute value of the relative difference between the two methods (after subtraction of the EIT digitizer offset) shows that sampling errors are concentrated near small solar structures and where the image shear or reduction is greatest. Hence the eastern limb near the poles has the highest concentration of error, while most of the total error is spread along the western portion of the image. The cleanest portion of the image is near and just east of disk center, where the mapping doesn't reduce the local feature size in any direction.

This EIT image has a balance of large and small features, but is not particularly 'spiky': most features are at least several pixels across. Nevertheless, relative errors of at least 5% occur in about 5% of the linearly interpolated pixels for this transformation, and errors of over 30% occur in about 0.2% of the pixels. Transformations with more shear or acting on 'spikier' data will yield more and stronger errors.

Sampling errors are enhanced by high spatial frequencies, so optimized re-sampling should be used wherever the objects of interest are small and well-separated, such as tracking of magnetic ephemeral regions, or identification of bright points in EUV images. The errors introduced by local interpolation in Figure 4 are systematic with position on the transformed plane and could easily poison the results of statistical surveys or evolution studies.

Sampling errors of the type shown here could certainly affect helioseismology results: transformations with singular values below unity will lengthen the parallel component of any observed wave's spatial k vector. k vectors that extend beyond the spatial Nyquist frequency of the re-sampled image will be aliased to lower spatial frequencies. The Gaussian-weighted average over each pixel's footprint has the effect of applying a spatially variable filter, locally tuned to prevent aliasing. For more information, see Heckbert (1989).

4. An Implementation of Coordinate Transformations

I have implemented the algorithm described above in the PDL language (Glazebrook *et al.*, 2003), which is available as the package 'PDL' in the SolarSoft distribution (Freeland and Handy, 1998). The algorithm is present in the `PDL::Transform` and `PDL::Transform::Cartography` modules, which include support for many standardized (and parameterized) coordinate transforms, including most commonly used map projections (e.g., Snyder, 1987). The software has a simple, streamlined inter-

```
#!/usr/bin/perl   # UNIX scripting magic
map{eval "use $_"} (PDL, PDL::Transform::Cartography, PDL::AutoLoader);

    # Read in the EIT data. Any FITS file with WCS pointing info
    # will work. (PDL stores the header as part of $a).

$a = float rfits('eit-sample.fits');

    # Prepare data for Figure 1 of this article.
    # Define mapping from lon/lat to image coordinates. Set B angle and
    # observer distance. im_unit fills in for CUNIT<n> in the hdr.

$t = t_perspective( B=>6.5, r0=>217, im_unit=>"arcsec" );
$amap = $a->map( !$t,  method=>Jacobian  ); # '!' is func. inverse
wfits($amap, "eit-map.fits");

    # Prepare data for Figure 4 of this article.
    # Demonstrate differential rotation after 28 days: 'x' is function
    # composition; t_diff_rot shears the data in lon/lat space.

$t2 = $t  x  t_diff_rot( 28.0 )  x  !$t
$awarp1 = $a->map( $t2,  method=>Jacobian  );
$awarp2 = $a->map( $t2,  method=>linear  );
wfits( $awarp1, "eit-warped.fits" );
wfits( $awarp1 - $awarp2, "eit-warp-diff.fits" );
```

Figure 5. PDL script demonstrating re-sampling for Figures 1 and 4. An EIT image is read in, then converted to solar long./lat. coordinates using spherical perspective projection. The map's FITS header is updated automatically. At bottom, the image is rotated forward by 28 days to generate the error plot in Figure 4. The top two lines ensure that the script can run directly from the UNIX command line, but are unnecessary for use within PDL's interactive environment.

face. The underlying language environment is built on perl, the popular scripting language (Wall, Christiansen, and Orwant, 2000).

The `PDL::Transform` module defines a new data type that represents a coordinate transformation. Transform objects can be inverted, composed, and otherwise manipulated with a simple expression syntax. Several generators are included for common transformations – e.g. `t_linear()` creates an arbitrary linear transformation based on a variety of parameters, `t_fits()` accepts a FITS/WCS header and returns a transformation object mapping pixel values to scientific coordinates, `t_radial()` creates a radial coordinate transformation, and `t_diff_rot` represents solar differential rotation using one of the familiar rotation laws from Cox (2000) or Howard, Harvey, and Forgach (1990). More than twenty specialized map projections and other transforms are available.

Transforms operate on vector data, transforming collections of vectors from one coordinate system to another via the `apply` method. For example, the expression `$a->apply (t_radial())` converts the set of vectors `$a` from Cartesian to radial coordinates. Transforms also operate on image data via the `map` method,

implementing several re-sampling schemes including the Jacobian singular-value padding method presented in Section 3. Typical image distortions require only 1–2 lines of code. Figure 5 lists the script that was used to distort the solar data in Figures 1 and 4. To my knowledge, the `PDL::Transform` package in perl/PDL is the only scientific software package that combines a generic coordinate transformation facility with optimized re-sampling.

5. Specialized Distortions and Applications

After linear transformation to co-align images, the most common application of image re-sampling is removal of differential rotation from time-series data. However, in many cases specialized coordinate transformations can greatly aid data analysis. Here I present two examples of coordinate transformations that reveal subtle but important aspects of particular solar images, and discuss further applications for image correction and regularization.

5.1. AZIMUTHAL STRUCTURE

Radial coordinate transformation is useful wherever azimuthal symmetry is important to an observation. For example, DeForest *et al.* (1997) and DeForest, Plunkett, and Andrews (2001) used azimuthal projection to study the structure of coronal holes during solar minimum. Figure 6 shows two images of the active, declining phase corona recorded by SOHO/LASCO (Brueckner *et al.*, 1995) in August of 2003, and the corresponding conformal radial projection[4]. Radially projecting shows the super-radial and sub-radial expansion rates at a glance. The conformal radial projection has an additional advantage: because the distance scale is logarithmic, it is possible to meaningfully overlay data from instruments that measure the solar surface and the outer corona (DeForest *et al.*, 1997), despite vast differences in scale.

Radial projection maps concentric circles to horizontal lines; notionally, a radially projected time series of images is a generalization of the sinogram data used by DeForest, Lamy, and Llebaria (2001) and Li *et al.* (2000) to study timeseries evolution of the solar corona: individual coronal 'sinograms' or 'TIDs' are slices through such a data cube.

Because radial projection has a strongly varying spatial scale, it is particularly important to use optimized re-sampling for this type of analysis, to avoid missing important structures in the very sparse sampling grid far from the origin. DeForest, Plunkett, and Andrews (2001) were able to use the sparseness to average large

[4]Conformal mappings preserve the shape, but not size or orientation, of small features. In mathematical terms, a mapping is conformal if its Jacobian is everywhere a scalar times a rotation matrix.

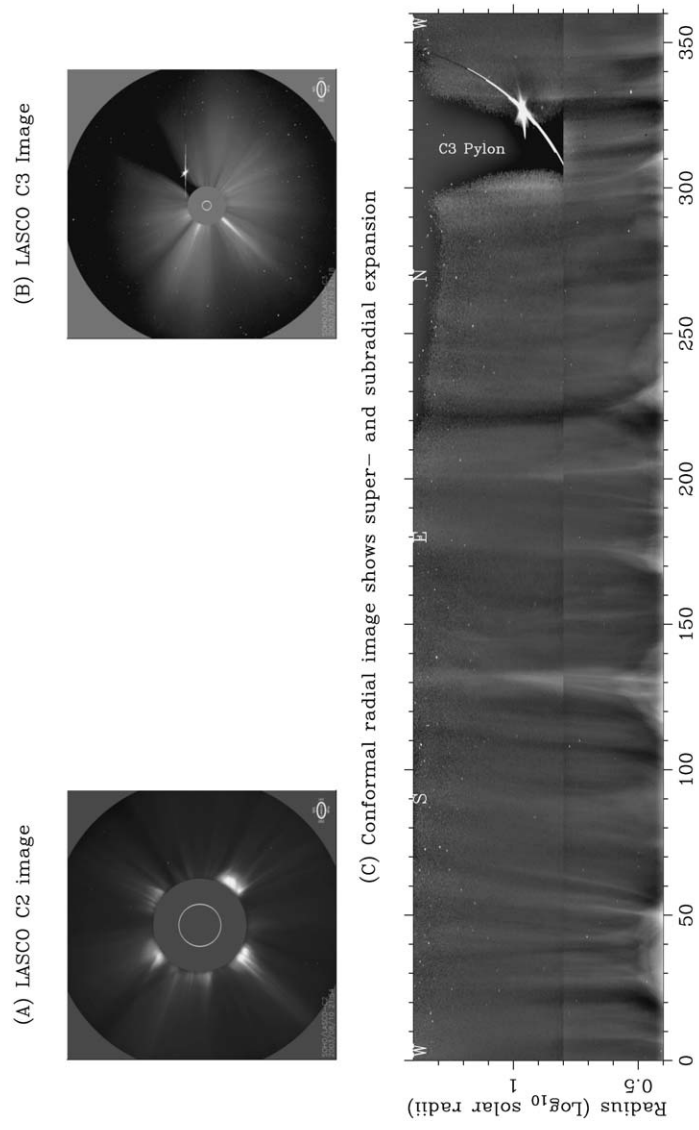

Figure 6. SOHO/LASCO images of the active, declining solar corona from 10 August 2003. (A) LASCO C-2 image showing the corona from 2–6 solar radii. (B) LASCO C-3 image showing the corona out to 30 solar radii. Venus is present near the dark pylon at the northwest part of the image. (C) A radially mapped image combining data from the two instruments. The radial projection is not quite conformal: vertical scale has been exaggerated by 50% for clarity, resulting in a slightly tall aspect to the imaged features.

numbers of pixels and beat down photon noise in the dim regions far from the Sun, enabling visual detection of polar plumes at altitudes over 25 R_s.

5.2. PULSE WAVE PROPAGATION

Several traditional map projections of the solar surface are useful for helioseismology and other studies. In particular, gnomonic projection is useful for studying propagation of directional disturbances on the solar surface, because of its property of preserving geodesics (great circles) as straight lines; and Postel's azimuthal, equidistant projection (Snyder, 1987) is useful for identifying wave propagation from a point source, because the projection preserves the k-vector of waves passing through the center of projection (Bogart *et al.*, 1995). Both of these projections introduce significant spatial distortions, and benefit from optimized re-sampling because the Jacobian is significantly sheared over large portions of the sphere.

Figure 7 shows an application of Postel's projection for characterization of pulse waves in the solar corona. Figure 7(A) and 7(B) are unmodified difference images taken with SOHO/EIT's 195 Å pass band on 24 September 1997. In the minutes before Figure 7(A), a flare occurred in the active region at lower left, yielding a Moreton wave with (rare) direct signatures in the 195 Å band. Some minutes later (Figure 7(B)), a coronal pulse wave of the type commonly referred to as an 'EIT wave' is visible about 0.5 solar radii from the origin. Biesecker *et al.* (2002) have plausibly identified these two features as the same wave front viewed at different times.

De-projecting the perspective view from EIT and reprojecting the solar surface into an oblique Postel projection centered on the active region (Figures 7(C) and 7(D)) removes the perspective distortion of the telescope view and demonstrates that the pulse wave in fact propagates nearly isotropically. In this projection, k-vectors through the origin (in this case the active region) are preserved, so that the wave front should remain exactly circular if the wave speed doesn't vary with direction. Because the wave propagates nearly isotropically, it fits very well between the concentric dotted circles (Figure 7(D)).

Distorting the image one more time by radial expansion about the origin (Figure 7(E) and 7(F)) reveals more about the two observed fronts: they are not concentric, suggesting that they may in fact be different phenomena. In this coordinate system, circles concentric to the origin of the projection appear as horizontal lines. The pulse wave front in Figure 7(F) fits well the boundaries drawn here, but the Moreton wave front is clearly not concentric with the pulse wave front. Moving the coordinate system origin so that the Moreton wave is rendered as a horizontal line renders the 'EIT wave' as a sine curve in this coordinate system. The two wave centers are separated by about 2 arc min.

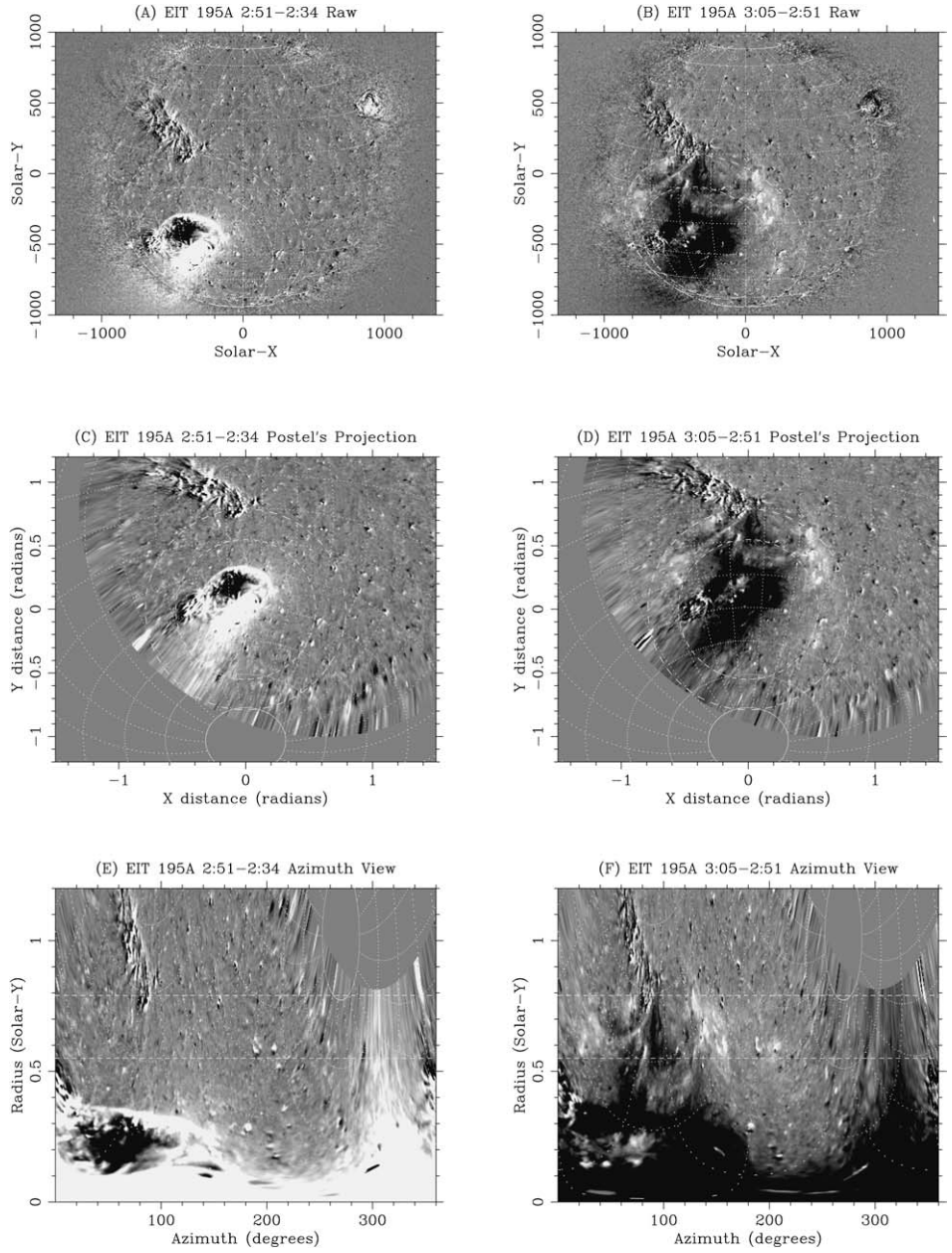

Figure 7. SOHO/EIT running-difference images in the Fe XII (195 Å) band showing coronal pulse wave phenomena associated with the flare of 24 September 1997. The running-difference images approximate the time derivative of brightness. (A) and (B) show two wave fronts in the original focal-plane coordinate system, visible 15 min apart; (C) and (D) are an oblique Postel projection that removes foreshortening, maintaining the wave fronts as nearly circular; and (E) and (F) are an azimuthal transformation of the Postel plane, showing that the two waves are non-concentric: the Moreton wave in (E) is not horizontal, because it is not centered at the origin of the projection.

5.3. DISCUSSION: OTHER APPLICATIONS

Parameterized image distortions have uses beyond perspective correction and appropriate coordinate-system selection. In particular, one might re-sample to remove geometric distortion that is inherent in the collecting optics. One example is correcting images from a telescope to simulate a pinhole camera: telescopes with optical magnification introduce significant $\tan(\theta)$ distortion into the edges of the field of view. Another is correction for off-axis, obliquely oriented focal planes. It is not uncommon for the plane of best average focus in a spectrometer or off-axis telescope to have known but variable magnification across the plane. In either case, knowing the distortion function allows one to remove the optical distortion from the data and work with images in a fictional pinhole-camera focal plane.

Telescopes that operate within an atmosphere have the problem of image distortion from an unknown optical system – the layered atmosphere overhead. Solar observatories typically identify the eccentricity of the ellipse described by the Sun's limb and stretch images along the minor axis, to account for the refractive effect of the atmosphere. A larger problem is atmospheric seeing, which distorts the image in an unknown way. Shine (1997) and others have developed image correlation and spatiotemporal filtering techniques that can remove from an image sequence, *a posteriori*, most of the geometric effects of seeing. In image correlation, small displacements are used to maximize the correlation coefficient between small patches of images collected at similar times. The image displacements determined by the maximization describe a distortion function which is then used to re-sample and stabilize the images. Optimized re-sampling is important for this application, to avoid re-sampling errors that are correlated with either seeing or surface motion.

6. Conclusions

Image re-sampling technique significantly influences the result of mathematical image distortion. Given the increasing popularity of image distortion as a tool for removing perspective and solar rotation effects, authors should be aware of the limitations of simple sampling and interpolation. Image re-sampling algorithms exist that can preserve image data much better under arbitrary transformations than does interpolation. I have presented one such algorithm, singular-value padding, that is simple to implement in most computational environments. An implementation exists in the Perl/PDL environment distributed with SolarSoft; authors are encouraged either to use it or to implement similar re-sampling code in other popular environments.

It is demonstrated that nonlinear coordinate transforms are useful in more contexts than ordinary perspective and rotation compensation. Two examples show cases where image morphology can be much easier tracked in coordinate systems that are unrelated to the simple projective geometry of a solar telescope. Azimuthal

projection is useful for systems where angle is important, such as the morphology of the mid corona. Specialized map projections exist that are useful for many applications; one such projection, Postel's projection, is demonstrated in the context of global tracking of coronal pulse waves.

Interactive manipulation of coordinate systems is a powerful analysis technique and should be part of the software 'toolkit' that data analysts bring to bear on solar data. Several useful software packages exist to fill this need. The Perl/PDL re-sampling code mentioned above exists in the context of one such general purpose nonlinear coordinate transformation suite. Other coordinate transformation packages exist, but appear to not (yet) use optimized re-sampling.

Acknowledgements

This work was funded under NASA grants NAG5-9795 and NAG5-11595. Special thanks to Sam Freeland for supporting PDL distribution through SolarSoft, to Meredith Wills-Davey for informative discussions about the EIT-wave literature, and to the SOHO LASCO/EIT team for the kind distribution of their data. SOHO is a project of international collaboration between NASA and ESA.

References

Berry, D. S.: 2001, 'Providing Improved WCS Facilities Through the Starlink AST and NDF Libraries', *ASP Conf. Ser. 238: Astronomical Data Analysis Software and Systems* X, p. 129.

Biesecker, D. A., Myers, D. C., Thompson, B. Jr., Hammer, D. M., and Vourlidas, A.: 2002, *Astrophys. J.* **569**, 1009.

Bogart, R. S., Sa, L., Duvall, T., Haber, D., Toomre, J., and Hill, F.: 1995, 'Plane-Wave Analysis of Solar Acoustic-Gravity Waves: a (slightly) New Approach', in *Helioseismology, ESA SP, Proc. SOHO 4*, p. 147. Available online: *http://rick.stanford.edu/pubs/Asilomar_1.html.*

Brueckner, G. E., Howard, R. A., Koomen, M. J., Korendyke, C. M., Michels, D. J., Moses, J. D., Socker, D. G., Dere, K. P., Lamy, P. L., Llebaria, A., Bout, M. V., Schwenn, R., Simnett, G. M., Bedford, D. D., and Eyles, C. J.: 1995, *Solar Phys.* **162**, 357.

Cox, A. N.: 2000, in A. N. Cox (ed.), *Allen's Astrophysical Quantities*, 4th ed. AIP Press, New York.

DeForest, C. E.: 1998, 'ZTOOLS'. Software package distributed via SolarSoft.

DeForest, C. E., Lamy, P. L., and Llebaria, A.: 2001, *Astrophys. J.* **560**, 490.

DeForest, C. E., Plunkett, S. P., and Andrews, M. D.: 2001, *Astrophys. J.* **546**, 569.

DeForest, C. E., Hoeksema, J. T., Gurman, J. B., Thompson, B. J., Plunkett, S. P., Howard, R., Harrison, R. C., and Hassler, D. M.: 1997, *Solar Phys.* **175**, 393.

Delaboudinière, J. -P., Artzner, G. E., Brunaud, J., Gabriel, A. H., Hochedez, J. F., Millier, F., Song, X. Y., Au, B., Dere, K. P., Howard, R. A., Kreplin, R., Michels, D. J., Moses, J. D., Defise, J. M., Jamar, C., Rochus, P., Chauvineau, J. P., Marioge, J. P., Catura, R. C., Lemen, J. R., Shing, L., Stern, R. A., Gurman, J. B., Neupert, W. M., Maucherat, A., Clette, F., Cugnon, P., and van Dessel, E. L.: 1995, *Solar Phys.* **162**, 291.

Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F.: 1996, *Computer Graphics: Principles and Practice*, 2nd ed., Addison Wesley, New York.

Freeland, S. L. and Handy, B. N.: 1998, *Solar Phys.* **182**, 497.

Glazebrook, K., Brinchmann, J., Cerney, J., DeForest, C. E., Hunt, D., Jenness, T., Lukka, T. J., Schwebel, R., and Soeller, C.: 2003, *The Perl Data Language, v.2.4.0.* Available online: *http://pdl.perl.org*.

Greisen, E. W.: 2003, 'AIPS the VLA, and VLBA', in *Information Handling in Astronomy – Historical Vistas*, p. 109.

Harvey, J. W., Hill, F., Hubbard, R., Kennedy, J. R., Leibacher, J. W., Pintar, J. A., Gilman, P. A., Noyes, R. W., Title, A. M., Toomre, J., Ulrich, R. K., Bhatnagar, A., Kennewell, J. A., Marquette, W., Patrón, J., Saá, O., and Yasukawa, E.: 1996, *Science* **272**, 1284.

Heckbert, P. S.: 1989, 'Fundamentals of Texture Mapping and Image Warping'. Master thesis, University of California, Berkeley, EE/CS Dept. Available online: *http://www-2.cs.cmu.edu/~ph/texfund/texfund.pdf*.

Hoeksema, J. T., Bush, R. I., Chu, K. -C., Liu, Y., Scherrer, P. H., Sommers, J., Zhao, X. P., and SOHO/MDI Team: 2000, 'Synoptic Magnetic Field Measurements', *AAS/Solar Physics Division Meeting* **32**.

Howard, R. F., Harvey, J. W., and Forgach, S.: 1990, *Solar Phys.* **130**, 295.

Li, J., Kuhn, J., LaBonte, B., Raymond, J. C., and Acton, W. T.: 2000, *Astrophys. J.* **538**, 415.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T.: 1986, *Numerical Recipes: the Art of Scientific Computing*. Cambridge University Press, Cambridge.

Scherrer, P. H., Bogart, R. S., Bush, R. I., Hoeksema, J. T., Kosovichev, A. G., Schou, J., Rosenberg, W., Springer, L., Tarbell, T. D., Title, A., Wolfson, C. J., Zayer, I, and MDI Engineering Team: 1995, *Solar Phys.* **162**, 129.

Shine, R. A.: 1997, *Bull. Am. Astron. Soc.* **29**, 916.

Snyder, J. P.: 1987, *Map Projections – A Working Manual*, No. 1395, USGS Professional Paper, United States Government Printing Office.

Torson, J. M. and Becker, K. J.: 1997, 'ISIS - A Software Architecture for Processing Planetary Images', in *Lunar and Planetary Institute Conference Abstracts*. p. 1443, *http://isis.astrogeology.usgs.gov/main.html*.

Turkowski, K.: 1990, 'Filters for Common Resampling Tasks', in *Graphics Gems I.* Academic Press, New York. Available online: *http://www.worldserver.com/turk/computergraphics/papers.html*.

Turkowski, K.: 1993, 'The Differential Geometry of Texture-Maping and Shading'. Technical report, Apple Computer. Available online: *http://www.worldserver.com/turk/computergraphics/papers.html*.

Wall, L., Christiansen, T., and Orwant, J.: 2000, *Programming Perl*, 3rd ed., O'Reilly, New York.

Wang, L.: 1994, 'IMAGE_TOOL'. Software distributed via SolarSoft.

Zarro, D. M.: 1998, 'PLOT_MAP'. Software package distributed via SolarSoft. Available online: *http://orpheus.nascom.nasa.gov/~zaroo/idl/maps.html*.