The Integrating Resampler and Efficient Image Warping[*]

Ming-Chao Chiang
Columbia University
Department of Computer Science
New York City, NY 10027
chiang@cs.columbia.edu

Terrance E. Boult
Lehigh University
Department of EECS
Bethlehem, PA 18015
tboult@eecs.lehigh.edu

ABSTRACT

This paper introduces *integrating resamplers* as an efficient method for image warping using a class of imaging-consistent reconstruction/restoration algorithms proposed in [Boult and Wolberg-1993]. Examples are given to show the use of the integrating resamplers described herein for two low-level computer vision applications. The first is geometric correction for images degraded by radial lens distortion, demonstrated as a preprocessing step in correlation-based stereo matching. The second is warping-based data fusion for polarization computations in a small-baseline multi-camera systems. These two applications are chosen because they exemplify the kinds of applications for which the integrating resamplers and the imaging-consistent reconstruction/restoration algorithms are designed–problems that use warped intensity values (as opposed to edge structure).

1 Introduction

Image warping has been around almost as long as image processing, allowing users to reshape the image geometry. Image warping requires the underlying image to be "resampled" at non-integer locations, hence it requires image reconstruction. When the goal of warping is to produce output for human viewing, only moderately accurate image intensities are needed. In these cases techniques using bi-linear interpolation have been found sufficient. However, as a preprocessing step for vision, the precision of the warped intensity values is often important. For these problems, bi-linear image reconstruction may not be sufficient.

This paper shows how to efficiently warp images using an image reconstruction techniques that includes a simple sensor model. By coupling the degradation model of the imaging system directly into the reconstruction process, we can derive better reconstruction techniques which more accurately mimic a "digital lens." This realization leads to the idea of imaging-consistent reconstruction/restoration algorithms proposed in [Boult and Wolberg-1993].

We present the integrating resampler, an efficient method for warping using imaging-consistent reconstruction/restoration algorithms. This algorithm is well suited for today's pipelined micro-processors. In addition, the integrating resampler can allow for modifications of the intensity values to better approximate the warping characteristics of real lenses.

The paper ends with a demonstration on two problems where we compare the integrating resampler with bi-linear resampling.

2 Image Formation and Sensor Model

Due to the limit of space, we only briefly review the image formation process as well as the sensor model proposed in [Boult and Wolberg-1993]. Image formation is generally described as a sequence of filtering operations, as depicted in Figure 1 (disregarding the feedback loops for the moment).

Let $f(x, y)$ be the intensity distribution of a scene at the front aperture of a lens. That distribution is acted upon by $h_1(x, y)$, the blurring component of the lens, yielding $f_1(x, y)$. A geometric distortion function $h_2(x, y)$ is then applied to produce image $f_2(u, v)$. Although the blurring and geometric distortions induced by a real lens are not necessarily decoupled in this manner, this approximate model lends itself to conceptual and computational simplifications. If we presumes the blur is small and not dominated by depth-of-field effects, this allows us to replace a spatially-varying point spread function with

[*] This work is supported in part by ARPA contract DACA-76-92-C-007 and DOD MURI program ONR N00014-95-1-0601. Several other agencies and companies have also supported parts of this research.

This paper appeared in the 1996 DARPA Image Understanding Proceeding and is copyrighted.

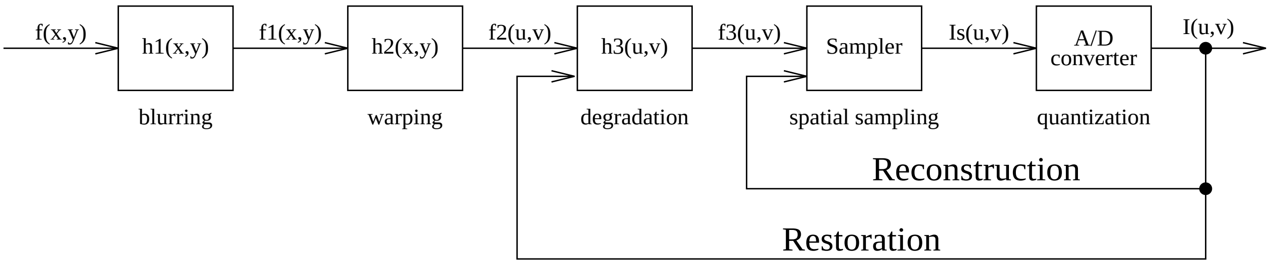


Figure 1: The image formation process and the relationship between restoration and reconstruction.

a cascade of two simpler components: a spatially-invariant blur and a geometric warp.

At this point, f_2 strikes the image sensor where it undergoes additional blurring by a point spread function $h_3(u,v)$ to generate image $f_3(u,v)$. This blurring reflects the limitation of the sensor to accurately resolve each point without the influence of neighboring points. We choose to use a simple model wherein this blurring takes place within one pixel because for CCD and CID cameras, the physical boundaries between photo-sites generally allow only insignificant charge transfer between pixels.

Image f_3 undergoes spatial sampling as it hits the discrete photo-sites in a CCD or CID camera. The combination of h_3 with sampling is known as area sampling. It reflects the finite size of the sampling area. If h_3 is taken to be an impulse, then we have point sampling. This ideal, often assumed for theoretical considerations, is generally not true in practice. In either case, intensities in the sampled image I_s are now defined only for integer values of u and v. The digital image $I(u,v)$ is obtained via an analog-to-digital converter that quantizes the samples of I_s. This completes the image formation process, leaving $I(u,v)$ as the starting point for subsequent processing, including image reconstruction and restoration.

3 Image Restoration

Reconstruction and restoration start with I (and models for one or more h_i), and seek to solve for one or more of the f_js. Recovering f is known as image restoration and is of considerable interest in image processing. Note that recovering a functional form of f_2 is a sub-pixel restoration. We shall consider only recovery of f_2, as proper restoration of f generally requires global methods.

There is an interesting relationship between reconstruction and restoration, as depicted by the feedback loops in Figure 1. While both processes start from the image samples in I, reconstruction limits itself to the problem of deriving the continuous function f_3. Restoration attempts to estimate the original input function f_2 (or f_1). Obviously, the two problems, reconstruction and restoration, are related, and the latter is more difficult. With reasonable assumptions, exact reconstruction of f_3 is at least theoretically possible. On the other hand, exact reconstruction of f_2 requires very tenuous assumptions. Even given the necessary assumptions, the problem of approximate restoration is generally more difficult.

If, however, we could recover a functional form for f_2, we could unwarp the image (applying the inverse of h_2) and then reapply the h_3 and sample the result to get a corrected image. We thus seek to derive an efficient restoration algorithm that inverts the point spread function of imaging sensors (but not that of the lens). The method can be local as long as the pixel's PSFs has at most one-pixel extent, a reasonable assumption for common imaging systems .

4 Imaging-Consistent Algorithms

Briefly stated, an algorithm is imaging-consistent if it is the exact solution for some input (from some space of allowed functions) which, according to the imaging model, would have generated the measured input. For image reconstruction, we achieve this by first restoring the image to yield f_2, then blurring it again by the pixel PSF. Let us review the derivation for one such imaging-consistent reconstruction/restoration algorithm. More complex image models and more complex point-spread-functions are discussed in [Boult and Wolberg-1993]. We present only one dimensional image models; higher dimensions are treated separably.

To simplify our discussions, let us denote the image values by p_i and pixel boundaries as k_i with regular spacing m. For our algorithms, the intensity value p_i is centered (located) at $k_i+m/2$. It is also convenient to let $x = (t - k_i)/m$, as t varies over a pixel. We do not consider end conditions; since the algorithms are local, any choice for the end conditions will have limited extent.

Once the idea of area samples as information is accepted, probably the simplest method to consider is based on a piecewise quadratic model for the image. If we assume a Rect PSF filter (1 in the pixel, 0 outside), the imaging consistent algorithm is easy to derive. To ensure the function is continuous and the method is local we define the value of the reconstruction at the pixel boundaries k_i and k_{i+1}, to be equal to be E_i and E_{i+1}. Any method of reconstruction could be used to compute E_i, though our examples will only include linear interpolation and cubic convolution.

Given the values E_i at the pixel edges, an additional constraint, that the integral across the pixel equal p_i, results in exactly three constraints:

$$Q(0) = E_i; \quad Q(1) = E_{i+1}; \quad \int_0^1 Q(x)\,dx = p_i.$$

From this, one can determine the following quadratic polynomial:

$$\begin{aligned} Q(x) &= 3(E_i - 2p_i + E_{i+1})x^2 \\ &\quad +2(-2E_i + 3p_i - E_{i+1})x \\ &\quad +E_i \end{aligned}$$

where $x = (t - k_i)/m$. Using cubic convolution to derive E_i and E_{i+1} yields:

$$\begin{aligned} E_i &= \tfrac{1}{8}(A(p_{i-2} + p_{i+1}) + (4 - A)(p_{i-1} + p_i)) \\ E_{i+1} &= \tfrac{1}{8}(A(p_{i-1} + p_{i+2}) + (4 - A)(p_i + p_{i+1})) \end{aligned}$$

The parameter A is generally in the range [-3, 0] in order to make the cubic convolution kernel resemble the sinc function, with the values $-.5$, $-.75$, and -1 having special significance, see [Simon-1975, Keys-1981, Park and Schowengerdt-1983]. Note that with $A = 0$, we have $E_i = (p_{i-1} + p_i)/2$ and $E_{i+1} = (p_i + p_{i+1})/2$.

Remember, the above gives f_2, a sub-pixel restoration. If one wants a reconstruction algorithm, we simply blur the resulting restoration by a pixel model (Rect PSF) for the output scaling. Assuming a Rect PSF, then the integral One can integrate the above quadratic polynomial to derive a functional form for reconstruction (at the same scale). The result is a cubic polynomial that spans from the center of one input pixel to the next.

$$\begin{aligned} &\int_w^1 Q(x)\,dx + \int_0^w Q(x)\,dx = \\ &\qquad [E_{i+2} - E_i - 2(p_{i+1} - p_i)]w^3 \\ &\qquad +[-E_{i+2} - E_{i+1} + 2E_i + 3(p_{i+1} - p_i)]w^2 \\ &\qquad +(E_{i+1} - E_i)w + p_i \end{aligned} \tag{1}$$

It is interesting to note that if we use linear interpolation to determine E_i, the resulting imaging consistent reconstruction algorithm is tantamount to cubic convolution with the "optimal" value of $A = -.5$.

5 Integrating Resampler

The imaging-consistent algorithms described above and in [Boult and Wolberg-1993] are linear filters. For linear and affine spatial transformations, the traditional implementation would be in terms of convolution with the impulse response. However, we presented reconstruction/restoration in functional form because we designed them for use in what we call the *integrating resampling* approach. This section describes integrating resamplers and their use in image warping.

As described before, our model of image formation requires the image to be spatially sampled with a finite area sampler. This is tantamount to a weighed integral being computed on the input function. Because we have a functional form for the reconstruction/restoration, we can simply integrate this function with the PSF for the output area sampler. In this section, we assume the output sampler has a Rect PSF, though there is no limitation on the degradation models that one can use.

When we are resampling the image and warping its geometry in a nonlinear manner, this new approach allows us to efficiently do both pre-filtering and post-filtering. Because we have already determined a functional form for the input, no spatially-varying filtering is needed, as would be the case if direct inverse mapping were done.

Computing the exact value of the restored function weighted by the PSF could be done in functional form if the mapping function has a functional inverse and the PSF is simple (as in this example). In general, however, it cannot be done in closed form and numerical integration is required. To reduce the computational complexity, we propose a scheme where for each input pixel we use a linear approximation to the spatial warping within that pixel, but use the full nonlinear warp to determine the location of pixel boundaries. This integrating resampler, presented in Figure 2, also handles antialiasing of partial pixels in a straightforward manner.

Assume n input pixels are being mapped into k output pixels according to the mapping function $m(t)$. Let m_i be the mapped location of pixel i, for i=$0, \ldots, n$. Compute q_j, j=$0, \ldots, k$, as the linear approximation to the location of $m^{-1}(j)$, as follows:

```
for ($i = j = 0;\ j \le k;\ j$++) {
  while ($i < n - 1$ && $m_{i+1} < j$) $i$++;
  $q_j = i + (j - m_i)/(m_{i+1} - m_i)$;
}
```

Note that we are assuming above the mapping function is strictly increasing to avoid fold-over prob-

```
    Pad the input; compute $k_l$, $k_r$, and $i_l$, the indices to the leftmost and rightmost output pixels and the
    index to the leftmost input pixel that contributes to the output; and compute the linear approximation
    to the location of $m^{-1}(j)$, for $j = k_l, \ldots, k_r + 1$.
normalizingfactor = $q_{k_l+1} - q_{k_l}$;              /* set up for normalization                      */
$q_{k_l}$ = MAX($q_{k_l}$, 0);                           /* ensure that $q_{k_l}$ is nonnegative          */
inseg = 1.0 − FRACTION($q_{k_l}$);                       /* fraction of input pixel left to be consumed   */
outseg = $q_{k_l+1} - q_{k_l}$;                          /* #input pixels mapped onto one output pixel    */
acc = 0.0;                                               /* reset accumulator for next output pixel       */
for ($j = 0$; $j < q_{k_l}$; $j$++) out[$j$++] = 0;      /* zero out the garbage at left end              */
for ($i = i_l$, $j = k_l$; $j <= k_r$; ) {               /* while there is output to produce              */
        Use the current pixel (in[$i$]) and its neighbors to update $R()$, the
        integral of the restoration $r()$.
  leftpos = 1.0 − inseg;                                 /* get left endpoint for integration             */
  if (inseg < outseg) {                                  /* if we will consume input pixel first          */
    acc += $R(1) - R$(leftpos);                          /* add integral to end of output pixel           */
    $i$++;                                               /* index into next input pixel                   */
    if ($i == n$) {                                      /* check end condition                           */
      if (normalize) acc /= normalizingfactor;           /* normalize the output, if appropriate          */
      out[$j$] = acc;                                    /* init output                                   */
      break;                                             /* exit from the loop                            */
    }
    outseg −= inseg;                                     /* inseg portion has been filled                 */
    inseg = 1.0;                                         /* new input pixel will be available             */
  }
  else {                                                 /* Else we will produce output pixel first       */
    acc += $R$(leftpos + outseg) − $R$(leftpos);         /* add integral to end of output pixel           */
    if (normalize) acc /= normalizingfactor;             /* normalize the output, if appropriate          */
    out[$j$] = acc;                                      /* init output                                   */
    $j$++;                                               /* index into next output pixel                  */
    acc = 0.0;                                           /* reset accumulator for next output pixel       */
    inseg −= outseg;                                     /* outseg portion of input has been used         */
    outseg = $q_{j+1} - q_j$;                            /* new output size                               */
    normalizingfactor = outseg;                          /* need for normalization                        */
  }
}
for ($j = k_r + 1$; $j < k$; $j$++) out[$j$++] = 0;      /* zero out the garbage at right end             */
```

Figure 2: The Integrating Resampler assuming a Rect PSF Filter, see text for discussion.

lems, see [Wolberg and Boult-1989] for a more general approach.

In order to allow efficient computation of the integral as well as to perform proper antialiasing, the algorithm, given as pseudo code in Figure 2, runs along the input and output determining in which image it will next cross a pixel boundary. To do this, we have two variables: *inseg* $\in [0, 1]$, which represents the fraction of the current input pixel left to be consumed, and *outseg*, which specifies the amount of input pixel(s) required to fill the current output pixel. If we assume the output PSF is a Rect filter, the closed form definite integral $R(t)$ of the restoration $r(t)$ from point a to b can be derived similar to equation (1). Other imaging-consistent algorithms yield different forms of R. Fant's original work can be viewed as imaging-consistent with a piecewise constant image resulting in a linear form for $R(t)$.

Whenever *inseg* $<$ *outseg*, we know that the input pixel will finish first, so we can consume it and update our state. If, on the other hand, it happens that *inseg* $\geq$ *outseg*, the output pixel will finish first, so we produce an output and update our state. Thus we process in a loop such that each time we either consume one input pixel or produce one output pixel; the algorithm requires at most $k + n$ iterations.

The underlying idea of this integrating resampler can be found in the work of Fant [Fant-1986] who proposed a similar algorithm for the special case of linear reconstruction. Our contribution is twofold: the generalization to deal with more advanced reconstruction algorithms, and providing for a modeling of real lens effects by using a modeling of real warps that affects the image radiance. In Fant's original work, the goal was to warp images for graphic (or visual) effects, and hence to affect geometry without disturbing the intensities. To do this, the algorithm maintained knowledge of the input size and normalized the integral to account for this size, giving a normalized intensity. Thus, if a constant image was stretched to twice its normal width, it would just change shape but retain the same intensities. Since we are interested in the modeling of lens characteristics, we realized that we do not want to do this normalization. If a lens was placed into an imaging system so as to double the width of the image on the sensor plane, then the value measured would be halved.

If an input scene f is warped according to some mapping function $m(t)$, then the measured intensity at location I_j will depend only on the energy from f that is warped according to m to reach the pixel. This assumes no blurring effects while in reality whatever caused the geometric distortion would likely have also resulted in some blurring. It also presumes the sensor is linear with respect to its input, which may not be the case near the extremes of the cameras dynamic range, or in cameras with non-unity gamma (gain), automatic gain control, or auto irises. Still, it is a better approximation than just resampling which assumes that the intensity is independent of the geometric distortions.

6 Examples and Applications

We demonstrate these ideas on two low-level computer vision applications. The first is correlation-based stereo with a pre-processing warp to enhance registration and correct for radial lens distortion. The second is warping-based data fusion for polarization computations in a small-baseline multi-camera systems.

6.1 Stereo Matching after Lens Distortion Correction

When doing stereo matching with correlation-based matchers it is usually assumed that the difference between images is a simple translation. Pre-warping the image to remove distortions can significantly improve the matching speed by allowing epi-polar constraints to result in a 1-D search. This idea is not new, however, previous work has presumed that bi-linear interpolation was sufficient for the warping. In this section we briefly show how the integrating resampler can improve the quality of the matching results.

The test data used an inexpensive Sun video camera which has noticeable radial distortion. Calibration and test images were from two converging views. The test images are shown in Figure3.

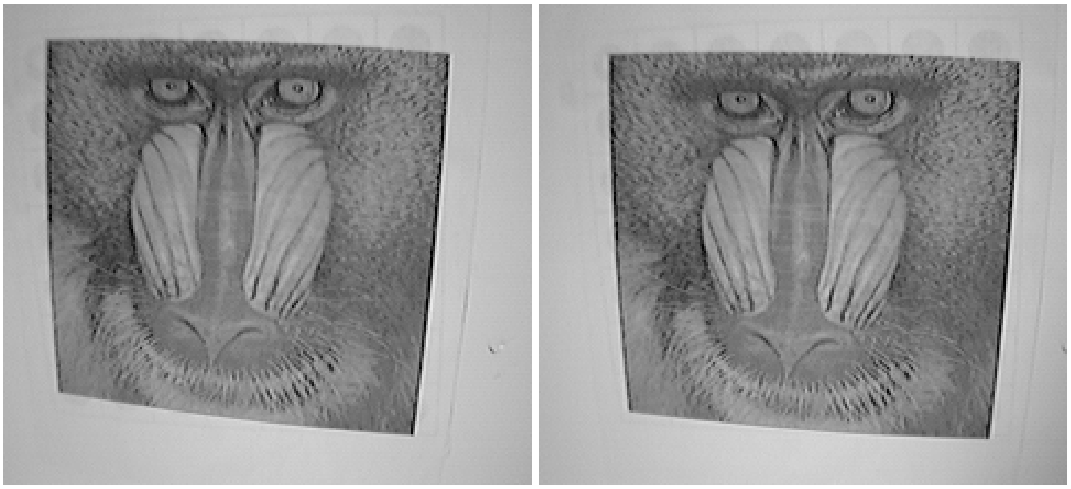
Geometric correction refers to the process of recovering undistorted images from distorted images. Though there exist several kinds of distortions, only radial lens distortion is considered in this example. The parameters that characterize the underlying camera model are computed using the two-stage camera calibration method described in [Tsai-1986, Tsai-1987, Chiang and Boult-1995].

Given the camera parameters, the input images were warped to remove the lens distortions. Two different warpings were applied, one using bi-linear warping (resampling), the other based on the integrating resample. These "corrected" images were then used as input to a sum-of-square distance stereo matching algorithm (7x7 template window). To reduce any dependency on calibration error, a 2D search window was used. The result of the matching is a dense disparity surface. In this demonstration, the test images were from a planar object and hence we evaluate the quality of warping by considering the best plane fit to the disparity data. For the images warped with the integrating resampler we found an average absolute error of 3.48, and a RMS error of 5.07. Using bi-linear interpolation the average absolute error was 4.0, and a RMS error was 5.62. The integrating resampler showed a 10-20% improvement over pre-warping with bi-linear interpolation. This appears to be a sufficient gain to warrant the slightly larger computational cost.

6.2 Warping-based Image Fusion for Polarization Computations

We turn now to the use of the imaging-consistent algorithms and the integrating resamplers for our second application, warping-based image fusion for polarization computations. These polarization computations might be used as a preprocessing step for other algorithms, e.g. as a preprocessing step for removal of specularities using color and polarization.

In our previous work, e.g. [Wolff and Boult-1991, Nayar *et al.*-1993], the experimental setup used a linear polarizing filter mounted on a precision rotation ring and placed in front of the camera lens. Images are taken at various orientations of the polarizer. This setup requires manually adjusting the orientation of



Left

Right

Figure 3: Left and right images for stereo example.

the linear polarizing filter, which is impossible for applications involving motion. More recent work by L. Wolff, [Wolff-1994], allows a LCD shutter to automate the polarization orientations, but it still requires 1/5 of a second to acquire the polarization information, limiting its use to static scenes. The accuracy of the computation of the percent polarization images depends, to a large extent, on how accurately the images are aligned and the accuracy of the intensity values after warping. The goal of our warping-based technique is to support polarization computations on moving objects.

Rather than using a single camera and a single polarizing filter, we have four Sony XC-999/999P color cameras tightly bundled together. Each has a linear polarizing filter mounted in front of its lens. Thus in a single frame time we can acquire 4 images at different polarization orientations. However, this setup introduces new problems. How do we align the images taken from these cameras? Warping! And, how do we ensure that the intensity values are accurate after the images are aligned? The integrating resampler, of course. (Note that with a very-short baseline (approximately 2.5cm) relative work area (approximately 25cm), and a moderate scene distance (approximately 2m) the variations due to scene depth are expected to be quite small. We calibrate the warping between the images using a fixed pattern. Polarization computations objects are simply placed into the scene.

Again, we use the integrating resamplers described herein to solve this problem. For this experiment we choose one of the four images as a reference image and warp the others to that reference image. Camera calibration, especially radiometric calibration, is needed for polarization computations (even with a single camera).

Region	RMS	Avg PP	%Error
$\bigtriangledown P < 10, \quad 35 < P$	1.12	41	2.7%
$\bigtriangledown P < 10, 10 < P < 35$	0.44	16	2.7%
$\bigtriangledown P < 10, \ P < 10$	0.36	3	12%
$10 < \bigtriangledown P < 20$	0.96	17	5.6%
$20 < \bigtriangledown P$	1.62	21	7.7%

Table 1: Errors in warping-based polarization computions.

To evaluate its use in polarization, we set up the cameras using a rotating polarizer as in our earlier work. Using this we could compute polarization using a single camera and via warping and thus compute the difference, which we have considered "errors". We note that "random" variations on the order of 5-10% of the computed polarization values occur when using a single camera setting. While there are numerous ways we could analyze this, we broke the error computations into 5 regions, shown in Table 1, see [Zhou-1995, Table3.12]. The regions consider high gradient of polarization as well as high and low polarization. In that table P refers to the percent polarization at a pixel, $\bigtriangledown P$ means the gradient of the percent polarization image at that point.

As can be see in table 1, the errors in warping-based polarization are quite acceptable. The larger errors in regions of very low polarization and regions of high polarization gradient are consistent with variations in polarization from a single camera, where those two regions have larger errors. We have done a few such experiments, with varying degrees of success, for example in a close-up experiment on curved objects, some of the the polarization errors were closer to 25%. While the integrating resampler can handle the warping with sufficient accuracy, warping is just a part of the polarization computation problem. Further work is needed before the multi-camera polarization computations are robust enough for general use. Difficulties arise because of:

- camera calibration (we need accurate radiometric calibration between the 4 cameras),
- lighting variations,
- object with high curvature (which cause specularities to move significantly for small changes in viewpoint) and
- objects with regular high frequency textures that can increase aliasing.

We are currently investigating techniques to address these issues.

7 CONCLUSIONS

This paper introduced integrating resamplers as an efficient method for warping with the imaging-consistent algorithms. Examples show the usefulness of integrating resamplers in two two low-level computer vision applications. Evaluations were made by comparing the resulting images using the integrating resamplers and those using bilinear interpolation. If accuracy in the underlying intensity values matter, the integrating resample offers advantages over traditional bi-linear mapping.

References

[Boult and Wolberg, 1993] Terrance E. Boult and George Wolberg. Local image reconstruction and subpixel restoration algorithms. *CVGIP*, 55(1):63–77, Jan 1993.

[Chiang and Boult, 1995] Ming-Chao Chiang and T.E. Boult. A public domain system for camera calibration and distortion correction. Tech. rep., Lehigh Univ. Department of EECS, and Columbia Univ., Dept of CS, Dec 1995. In preperation.

[Fant, 1986] Karl M. Fant. A nonaliasing, real-time spatial transform technique. *IEEE Computer Graphics and Applications*, 6(1):71–80, January 1986. See also “Letters to the Editor” in Vol.6 No.3, pp. 66-67, Mar. 1986 and Vol.6 No.7, pp 3-8 July 1986.

[Keys, 1981] Robert G. Keys. Cubic convolution interpolation for digital image processing. *IEEE Trans. Acoust. Speech, Signal Process.*, ASSP-29:1153–1160, 1981.

[Nayar *et al.*, 1993] Shree K. Nayar, XiSheng Fang and Terrance E. Boult. Removal of specularities using color and polarization. *CVPR*, pages 583–590, 1993.

[Park and Schowengerdt, 1983] Stephen K. Park and Robert A. Schowengerdt. Image reconstruction by parametric cubic convolution. *Computer Vision, Graphics, and Image Processing*, 23:258–272, 1983.

[Simon, 1975] K.W. Simon. Digital image reconstruction and resampling for geometric manipulation. *Proc. IEEE Symp. on Machine Processing of Remotely Sensed Data*, pages 3A–1–3A–11, 1975.

[Tsai, 1986] Roger Y. Tsai. An efficient and accurate camera calibration technique for 3D machine vision. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–374, 1986.

[Tsai, 1987] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344, August 1987.

[Wolberg and Boult, 1989] George Wolberg and Terrance E. Boult. Image warping with spatial lookup tables. *Computer Graphics*, 23(3):369–378, July 1989. (SIGGRAPH ’89 Proceedings).

[Wolff and Boult, 1991] L. B. Wolff and T. E. Boult. Constraining object features using a polarization reflectance model. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(7):635–637, 1991.

[Wolff, 1994] L.B. Wolff. Advances in polarization vision. In *Proc. of the ARPA Image Understanding Workshop*, pages 605–616. ARPA, 1994.

[Zhou, 1995] Yanhong Zhou. *Research in Applying Image Warping to Polarization Parameter Fitting*. PhD thesis, M.S. thesis (advisor T. Boult), Dept. of Ele. Eng. and Comp. Sci., Lehigh Univ., 1995.