Remote High Speed Measurement of Small Diameters by Optical Diffractometry and Interferometry

M. W. Siegel

CMU-RI-TR-88-15

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

September 1988

Contents

Illustrations

Tables

Abstract

The characteristics of pulsed diode lasers and linear CCD array optical detectors are explored in the context of high speed precision dimension and location measurements by optical diffraction and interference. It is suggested that pixel based metrology may, with appropriate signal extraction and processing methods, be made *more* accurate by making edges *less* well defined: if the mechanism of definition loss is well understood, *e.g.*, diffraction, the intensity distribution over many pixels can be matched to a template to within a small fraction of a pixel, obviating the need to make sub-pixel interpolations based on local intensities. The specific example of making remote flash measurements of the diameter and location of an opaque cylindrical object, *e.g.*, a fast-moving textile strand, is examined. A model is developed for calculating the diffraction pattern given the target and instrument parameters. The results are examined and compared with experimental data. Algorithms for finding the diameter and location given the diffraction and interference pattern and the instrument parameters are described. The performance statistics of a prototype instrument are measured in a data run of 1000 laser shots. Location and diameter measurement standard deviations are 0.13 pixel-widths (1.69 μm) and 0.36 pixel-widths (4.68 μm) respectively, implying that sub-micron precisions are obtainable by averaging a few dozen measurements. Sources of drift and scatter, and their reduction, are discussed in the context of designing and building an improved second generation instrument.

Introduction

Pulsed semiconductor diode lasers and CCD detectors invite low cost, high speed, high precision optical diffraction and interference measurements of mechanical locations and dimensions. A diode laser, a high current pulsed power supply, and a triggering circuit [8] can be built into a penlight-size module capable of delivering fast (≈ 200 nsec) pulses of high power (≈ 10 watts) in the near infra-red (≈ 0.9 μm) at high repetition rates (≈ 2 kHz) inexpensively (≈ $200). A linear CCD array of 256 elements, its clocking circuits, and output signal amplifiers can be purchased as an evaluation board [4] or user-assembled into a compact module for another ≈ $200. The dimensional accuracy of the CCD pixelation (typically on 13 μm centers) makes it an attractive ruler. Given these inexpensive sources and detectors the designer can imagine dozens of optical measurement configurations, each adapted to the needs of a specific application.

The dimensional accuracy of CCD pixelation *must* be extremely high: the only practical way to maintain layer-to-layer alignment of large scale integrated circuit devices is for each process step to be dimensionally accurate in an absolute sense [6]. On the other hand, despite the accuracy of the ruler, in the absence of a good sub-pixel interpolation scheme the precision of location and dimension measurements made with a 13 μm ruler can be at best mediocre relative to the sub-μm measurement requirements of practical interest in many contexts. Sub-pixel interpolation is usually done by a local gray-level method, *e.g.*, a gray-registering pixel straddled by a white-registering pixel and a black-registering pixel is defined to be partially obscured by a step-function edge whose location is linearly interpolated via the gray-level intensity [9].

The models and experiments described in this report suggest that, with several provisos, CCD based location and dimension measurements can be made *more* accurate by making edges *less* well defined:

- the method of edge definition loss is well understood, *e.g.*, diffraction;
- the parameters that dictate the dimensional details of the spread-out edge, *e.g.*, optical wavelength and apparatus geometry, are known with adequate accuracy;
- model-based signal extraction and processing methods are utilized.

Under these circumstances, the intensity distribution of an edge *spread out by diffraction over 100 or more pixels* can be matched to a template to within a small fraction of a pixel. I suggest that the result can be substantially more precise than sub-pixel interpolation based on local gray-levels.

In this report I describe a system that uses a pulsed diode laser, a CCD evaluation board, and a few inexpensive optical components to make flash measurements of the diameter and location of a stranded material composed of several hundred fibers (≈ 10 μm fiber diameter, ≈ 200 μm strand diameter) moving at high speed (> 50 m-sec^{-1}). The laser and the CCD are controlled by a single-board microcomputer[1] that communicates with a computer workstation that runs the inversion algorithms and reports the results and statistics.

The harsh environment of the application practically precludes an imaging system. This environment, and a requirement for unobstructed operator access to the strand, dictate an instrument with a large standoff (> 150 mm). At unit magnification with this standoff an image would be only about 15 pixels across. Thus even carefully implemented gray-level based sub-pixel interpolation methods would

[1]Actually the apparatus uses several lasers and CCDs to make a set of synchronous diameter and location measurements that are fused to give a measurement of higher level interest, *e.g.*, for process diagnostics and control. The details will be discussed in another report.

yield relatively low precision. Higher than unit magnification is in principle a solution, but it would require an impractically long optical system, *e.g.,* magnification of 5x with a 150 mm standoff would put the detector 900 mm (3 feet) from the strand. Folded optical paths are of course a possibility, but prohibitively expensive in the application's context.

Laser diffraction methods for measuring diameters have developed rapidly in the last few years, primarily in response to the process control requirements of the fast growing optical fiber communication industry, and possibly also in response to the similar requirements of specialty metal wire drawing industries, *e.g.,* for integrated circuit wire bonding. The optical fiber case, at least, is clearly amenable to diffraction measurements at large scattering angles, where due to refraction and internal reflection a large angle signal is available in transmission [2], [3].

However strands composed of multiple fibers are practically opaque, and only diffusely reflecting, precluding large angle scattering methods. Having thus ruled out imaging and large angle scattering, what is left is small angle scattering in the forward direction, *i.e.,* analysis of the shadow that the strand casts when interposed between the laser and the detector. This shadow, actually a Fresnel diffraction pattern[2], contains, I will show, two approximately independent sources of precise (albeit computationally expensive to extract) information about the strand diameter and location. The method requires virtually no optics: an inexpensive (≈ $40) and uncritical microscope objective simplifies the analysis by collimating the laser, and an equally inexpensive and uncritical cylindrical lens increases signal-to-noise by focusing the longitudinal direction onto the linear array. This simplicity and robustness is essential in the application's environment, which includes severe vibration, severe electrical interference from the switching of kiloampere currents, high temperature, humidity in excess of 100%, and a high density of particulates that cling tenaciously to lenses and other glass optical elements. These constraints preclude a finicky optical system: the precision of my measurement has to come from someplace other than the precision and stability of the mechanical and optical alignment.

Theory Behind the Model

Modeling the shadow of an opaque wire-like object is straightforward, although a straightforward implementation of the spatial integrals is computationally expensive. Since I work in an environment in which computational expense only indirectly translates into monetary expense, my model does the integrals by brute-force. This slow-but-sure approach achieves flexibility and an absence of pitfalls that might not be achieved by a clever implementation, *e.g.,* one employing approximations to the phase factors to facilitate symbolic rather than numerical integration.

Since the light scatterer is wire-like, I will model the problem for cylindrical waves, so that all planes normal to the scatterer are equivalent, and only one needs to be considered. The resulting geometry of the scattering (or "source") plane and the detector plane is illustrated in Figure 1. A coordinate axis labeled z_{src} coincides with the wire-like scatterer. An orthogonal coordinate axis y_{src} passes through $z_{src} = 0$. The optical electric-field in the $y_{src} z_{src}$-plane has amplitude and phase given by $E(y_{src})$ and $\phi(y_{src})$ in

$$\mathbf{E}(y_{src}) = E(y_{src})\, e^{i(\phi(y_{src}) - \omega t)}$$

---

[2]Diffraction patterns are broadly classed as Fresnel or Fraunhoffer based on the simplifying approximations that can be made about the phase factors under the integral signs in, *e.g.,* Huygens' Principle [10] or Green's Function [7] models of optical field propagation. For the purposes of this report, it is reasonable to say that a Fresnel pattern is a shadow (no lens used) and a Fraunhoffer pattern is an image (lens used). In both cases, regions that in geometrical optics would contain abrupt transitions between light and dark acquire oscillatory structure via diffraction and interference effects.

where ω is the optical frequency. The wavenumber $k = \frac{\omega}{c} = \frac{2\pi}{\lambda}$, where c is the speed of light and λ the optical wavelength, is the magnitude of the propagation vector, which lies along a third axis x orthogonal to the $y_{src}z_{src}$-plane and passing through its origin. The polarization vector does not have to be specified, but to be concrete let us say it is in the z_{src} direction. A detector, realized as a linear CCD array, lies at $x = d$, along coordinate axis y_{det} which is parallel to y_{src}. At any point on the y_{det} axis the optical electric-field is given by a Huygens' Principle integration over y_{src}:

$$\mathbf{E}(y_{det}) = [\int_{-\infty}^{y_L} + \int_{y_R}^{+\infty}] \frac{E(y_{src})}{\sqrt{d^2 + (y_{src} - y_{det})^2}} [1 + \frac{d}{\sqrt{d^2 + (y_{src} - y_{det})^2}}] e^{i(\phi(y_{src}) + k\sqrt{d^2 + (y_{src} - y_{det})^2} - \omega t)} dy_{src} \qquad (1)$$

where y_L and y_R are the y_{src} coordinates of the left and right edges of the strand.

The factors of this equation are understood as follows:

1. $\frac{E(y_{src})}{\sqrt{d^2 + (y_{src} - y_{det})^2}}$: The electric-field amplitude at y_{det} attributable to the Huygens' wavelet originating at y_{src}. The radiated electric-field falls off *linearly* with distance between source and detector, not quadratically: it is the energy density, proportional to the square of the electric-field, that falls off as the inverse square of distance.
2. $[1 + \frac{d}{\sqrt{d^2 + (y_{src} - y_{det})^2}}]$: Kirchoff's modification of Huygens' Principle, the "obliquity factor," which assures that Huygens' wavelets propagate only in the forward direction. The factor is usually written $1 + \cos\theta$, where θ is the angle of wavelet propagation with respect to the forward direction.
3. $e^{i(\phi(y_{src}) + k\sqrt{d^2 + (y_{src} - y_{det})^2} - \omega t)}$: the optical phase at y_{det} lags the phase at y_{src} by the wavenumber times the path distance between y_{src} and y_{det}.

Photodetectors in general, and CCDs specifically, respond to incident optical energy, which is proportional to $|\mathbf{E}(y_{det})|^2$ integrated over the detector area and multiplied by the exposure time. The proportionality constant is $\frac{\kappa\varepsilon_o c}{2}$, where κ is the dielectric constant of the medium, *e.g.*, air, and ε_o is the permittivity of free space, times factors describing the detection efficiency and the proportionality between incident energy and output signal voltage. Since all these factors, and also the pixel height, will ultimately be normalized away, I will simply dispense with them now and write, for the signal from the n-th pixel:

$$S(n) = \frac{\int_{y_{det}^n}^{y_{det}^n + w_{pixel}} |\mathbf{E}(y_{det})|^2 dy_{det}}{w_{pixel}}$$

where y_{det}^n is the y_{det} coordinate of the "left edge" of the n-th pixel, and w_{pixel} is the width of the unmasked active area of the pixel. Typical values are $13\,\mu m$ pixel-to-pixel spacing, and $5\,\mu m$ active width.

Implementation of the Model

Doing the integrals over y_{src} and y_{det} requires:

- replacing the integration limits of $-\infty$ and $+\infty$ on y_{src} with appropriate finite limits;

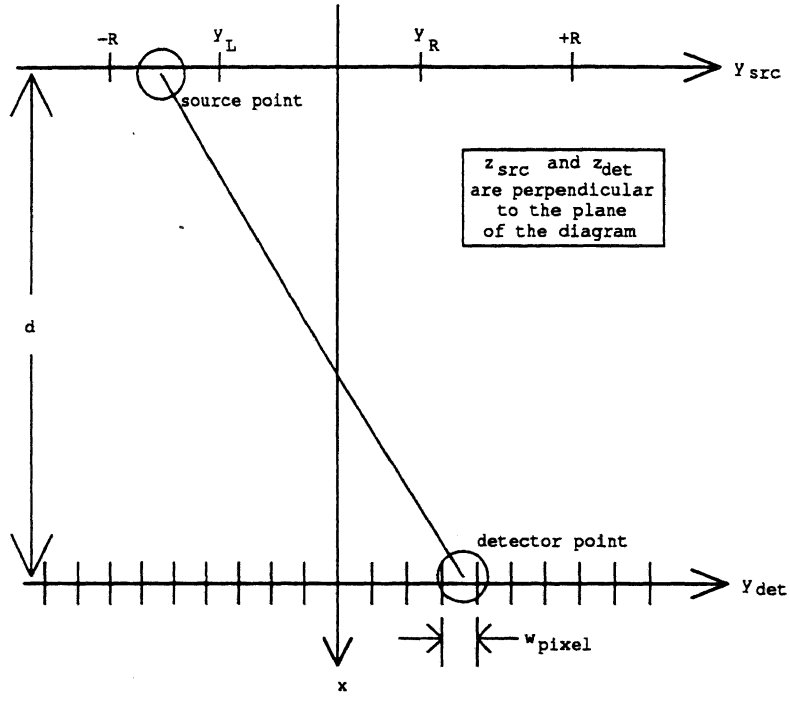

Figure 1. GEOMETRY OF THE APPARATUS.

Not to scale: d is approximately 167 mm, w_{pixel} is 13 μm, and the total width of the CCD (256 cells) is approximately 3.3 mm

- specifying appropriate forms for $E(y_{src})$ and $\phi(y_{src})$;
- specifying appropriate step sizes.

Many of the choices are heuristic, justified by the absence, when they are made, of artifacts in the result, robustness of the result against small changes in any of the forms and parameters selected, and ultimately, agreement between the model and experimental data. The following short paragraphs outline a successful course.

Doing the integral over y_{det} requires specifying an appropriate step size. Some computational efficiency, especially with respect to being able to reuse partial results from example to example, is greatly enhanced if we assume, and justify by comparison between the resulting calculation and the data, that the optical phase changes sufficiently little over the active width of a pixel that it is valid to use:

$$S(n) = \left| \frac{\int_{y_{det}^n}^{y_{det}^n + w_{pixel}} E(y_{det})\, dy_{det}}{w_{pixel}} \right|^2$$

Using this approximation is equivalent to saying that the phase changes slowly enough over a pixel width that it is adequate to average the electric-field vector components over a pixel and take the square-magnitude of the result instead of averaging the square-magnitude over a pixel.

The problem of "replacing the integration limits of $-\infty$ and $+\infty$ on y_{src} with appropriate finite limits" is intimately connected with "specifying appropriate forms for $E(y_{src})$ and $\phi(y_{src})$." The steps are:

- Assume a collimating lens, the microscope objective mentioned in the Introduction, that transforms the laser beam pattern into plane waves in the scattering plane: thus $\phi(y_{src}) = constant$, and we might as well call the constant zero;
- Recognize that ideal laser beams have a gaussian radial structure, so we would like to be able to write:

  $$E(y_{src}) \sim e^{-\left(\frac{y_{src}}{r_o}\right)^2}$$

  where r_o is a scale length proportional to an appropriate transverse linear dimension of the active laser channel;
- Reason that the appropriate scaling length is given by

  $$r_o = R\sqrt{\frac{ln\,2}{\tan(\sin^{-1}NA)}}\,\tan\theta_{FWHM}$$

  where R is the half aperture (radius) of the microscope objective, NA is the numerical aperture of the microscope objective, and θ_{FWHM} is the angular full-width-at-half-maximum of the laser beam divergence perpendicular to the long dimension of the scatterer;
- Recognize that unless the integral is somehow cut off, *e.g.*, for $|y_{src}| > R$, the integration is both physically unreasonable and numerically cumbersome. Also, discover empirically that unless the cutoff is made smoothly, artifacts that are very sensitive to the details appear in the result. Heuristically, a way not to generate artifacts is to integrate y_{src} between $-R$ and $+R$ while smoothly reducing $E(y_{src})$ to zero at the limits by parabolically truncating the gaussian, *i.e.*, taking:

  $$E(y_{src}) \sim [1-\left(\frac{y_{src}}{R}\right)^2]e^{-\left(\frac{y_{src}}{r_o}\right)^2}$$

- Finally, learn empirically that step sizes for the integrals over y_{src} and y_{det} of $1\,\mu m$, giving 13 steps per inter-pixel spacing, and 5 steps across each active pixel width, are a reasonable compromise between smaller step sizes resulting in excessive computational time and larger step sizes resulting in artifacts such as spurious modulation.

Appendix A is the program *huygens* that implements this model.

Comparison of Model with Data

In this section I compare data collected with prototype apparatus of approximately the geometry described in Table 1 with calculations generated by the *huygens* program using these parameters. In the following figures illustrating the model the calculation is degraded to 6-bits of signal amplitude resolution, corresponding to the 6-bits of actual resolution available from the analog-to-digital flash converter in the apparatus; data and model are each subjected to a linearly weighted five point sliding average before plotting.

Table 1

Parameters of the apparatus generating the data
and of the model simulating the data

Parameter	Value	Comment
CCD cells	256	Number of cells in the linear CCD detector
Pixel spacing	13.0 μm	Pixel-to-pixel spacing
Pixel size	5.0 μm	Active width of each pixel
Distance	167 mm	Target to detector distance
Wavelength	0.895 μm	Near infra-red pulsed diode laser
Lens radius	2.5 mm	Half-diameter of collimating lens
NA	0.1	Numerical Aperture of collimating lens
FWHM	30.0 deg	Full-Width-at-Half-Maximum laser divergence
ADC range	6-bits	Resolution of analog-to-digital converter

Sources of error that should be borne in mind when comparing data and model include

- the actual wire diameter is uncertain because on the one hand the diameter is reduced by stretching, and on the other hand it is increased by enamel insulation;
- the distance is uncertain by mechanical measurement error, and by imperfect collimation of the laser beam, which may be slightly convergent or divergent.

Figure 2 compares experimental data for a 26 awg[3] wire, nominal diameter 404 μm, with the model for a target diameter of 400 μm. This diameter is about twice the anticipated strand diameter in the application of interest. The graphs for data and model are slightly offset from each other horizontally to correct for the wire being inexactly centered on the CCD. The contrast in the data is somewhat less than the contrast in the model, due no doubt to stray light, scattering of the laser beam by optical surfaces, optical imperfections, dust, etc. The correlation between data and model are clearly extremely high.

Figure 3 compares experimental data for a 28 awg wire, nominal diameter 320 μm, with the model for a target diameter of 320 μm. This diameter is about 50% larger than the anticipated strand diameter in the application of interest. The remarks made with respect to Figure 2 again apply. Systematic diameter related differences between Figures 2 and 3 are easily discerned. For example, the prominent wiggle between each toe and shoulder of the shadow has moved outward in both data and model. But the wiggle that in Figure 2 is midway between each toe and shoulder of the shadow has in both the data and model of Figure 3 moved all the way up to each shoulder in the model, but only part way there in the data. We can probably conclude that the 26 awg wire is slightly oversize, *i.e.*, in the direction of 26 awg. Since the thickness of the enamel insulation has not been taken into account, this is reasonable.

Figure 4 compares experimental data for a 32 awg wire, nominal diameter 202 μm, with the model for a target diameter of approximately 200 μm. This diameter is about equal to the anticipated strand

[3]American Wire Gauge (awg) diameters increase logarithmically with decreasing gauge number. In this report the tabular relationship [1] is represented by $diameter\,(\mu m) = exp(m\ awg + b)$ where $m = (log\,(79.87/404.0)) / (40.0 - 26.0)$ and $b = log\,(79.87) - 40.0 \times m$

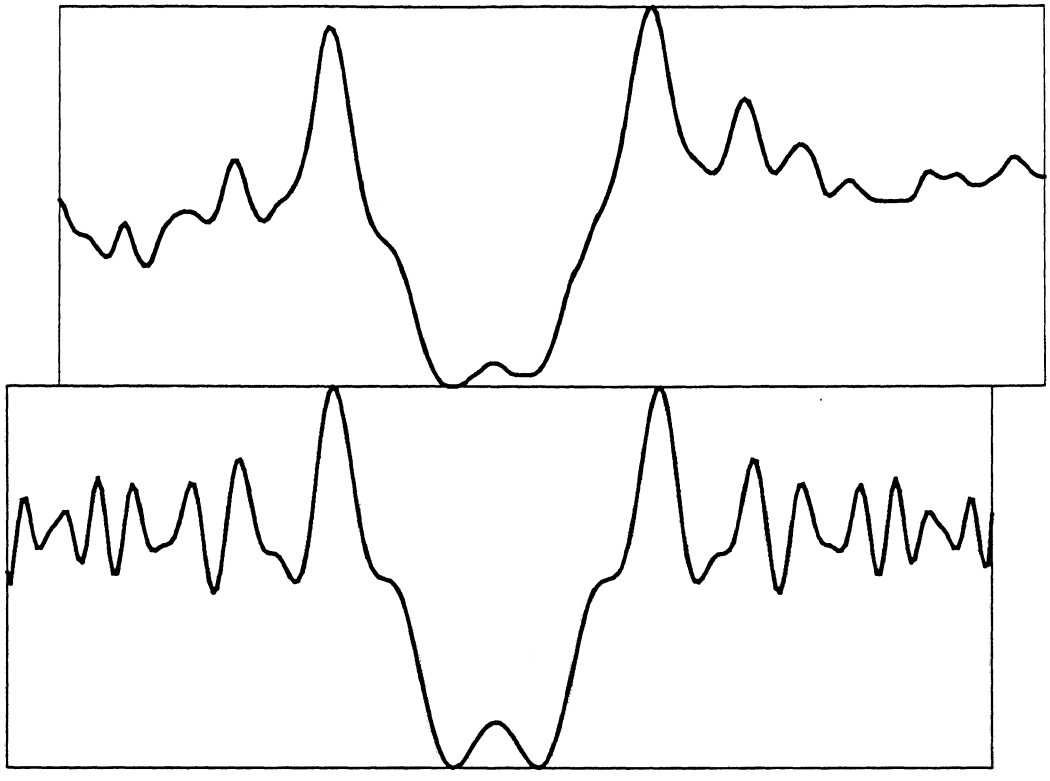

Figure 2. 26 AWG WIRE DATA

Top, data for 26 awg wire, approximately 404 μm diameter;
bottom, model for 400 μm diameter.

diameter in the application of interest. As in Figures 2 and 3, except for degraded contrast in the experimental data the agreement between data and model is excellent. The wiggle that moved outward between Figure 2 and 3 has moved still farther out, somewhat more so in the model than in the data. This probably indicates that the 32 awg wire is also somewhat oversize, again probably due to the enamel insulation.

Figure 5 compares experimental data for a 38 awg wire, nominal diameter 101 μm, with the model for a target diameter of approximately 100 μm. This diameter is about half the anticipated strand diameter in the application of interest. The trends identified in Figures 2, 3, and 4 are seen to continue in a smooth and predictable fashion.

There are three obvious conclusions of this comparison:

1. the model and the data agree well;
2. the data are rich in diameter dependent features;
3. over the range of target diameters between 100 and 400 μm the dominant feature of the data, *i.e.,* the central dip that intuitively might be associated with the geometrical shadow, in fact changes only slightly in width; thus the width of this dip is *not* a straightforward measure of target diameter;
4. the wiggle seen midway between each toe and shoulder of the dip in Figure 2, diameter ≈ 400 μm, has by Figure 5, diameter ≈ 100 μm, moved, quite systematically, halfway across the CCD.

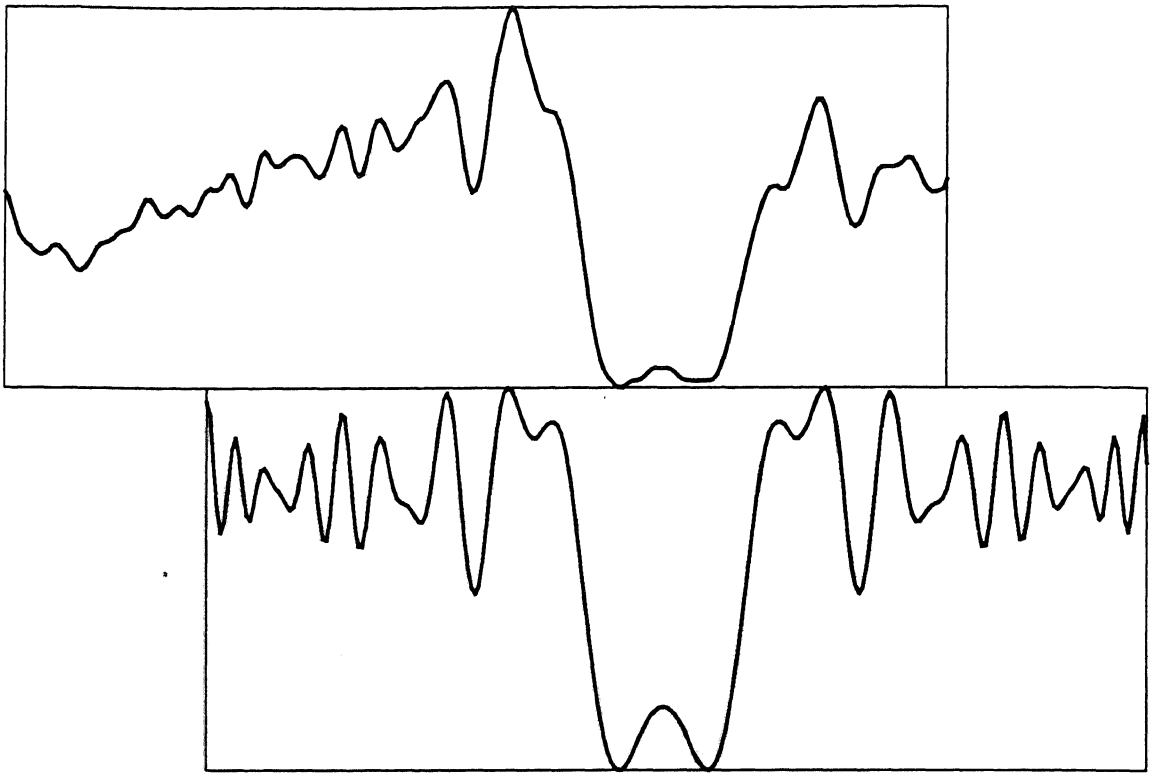

Figure 3. 28 AWG WIRE DATA

Top, data for 28 awg wire, approximately 320 μm diameter;
bottom, model for 320 μm diameter.

Figure 6 aids in understanding the two distinct diameter dependent features in the shadow by showing the model result for a 200 μm diameter target with the *distance* parameter set to 1 mm, 10 mm, and 100 mm respectively. These three shadows can also be compared with Figure 4, which corresponds to the prototype apparatus *distance* of 167 mm. In Figure 6, as in Figures 2–5, the model has been subjected to the same 6-bit digitization and five-point linearly weighted smoothing as the data. At 1 mm, the top of Figure 6, the width of the central dip is essentially the diameter of the target[4], *i.e.,* it is essentially a geometrical shadow with some fine structure determined by diffraction and interference. At 10 mm, the middle of Figure 6, the width of the central dip has increased significantly, and diffraction effects extend for several tens of pixel-widths to each side of each shoulder. At 100 mm, the bottom of Figure 6, diffraction and interference effects extend at least a hundred pixel-widths beyond each shoulder.

It is especially apparent in the bottom member of Figure 6, as well as in Figure 5, that there are two phenomena contributing to the structure of the data. Most of the signal power is contained in a pattern of wiggles whose spacing and amplitude *decrease* systematically with distance from the shoulders of the geometrical shadow. The peak and valley locations in this structure do not obviously depend on the wire diameter, although there is a distinct dependence of the oscillation amplitude on the diameter. Superimposed on this weakly diameter dependent structure is a pattern of *uniformly* spaced wiggles whose *spacing increases dramatically as the diameter decreases.* This second feature contains regrettably little signal power, and thus while it is easily visible in the model, parts of it are sometimes difficult to identify in the noise of the experimental data.

[4]The measurement can be verified simply by multiplying the ratio of the dip width to the figure width by the actual width represented by the figure width, 256 CCD channels times 13 μm per channel.

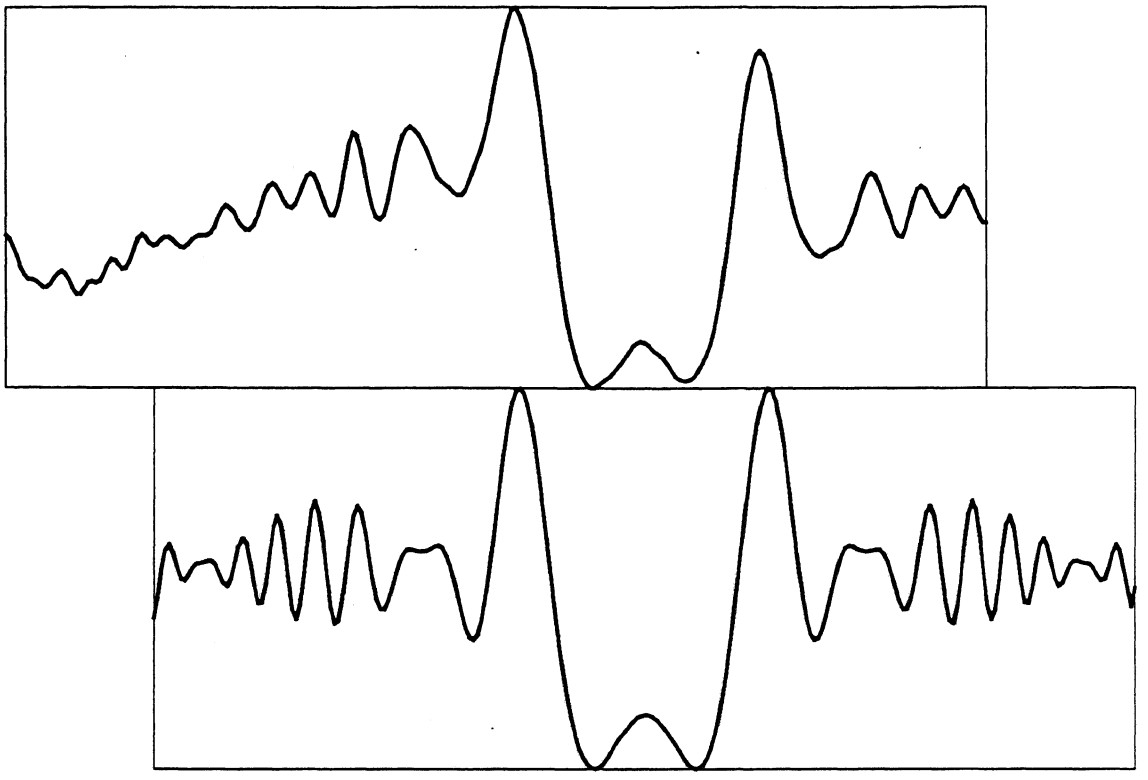

Figure 4. 32 AWG WIRE DATA

Top, data for 32 awg wire, approximately 202 μm diameter;
bottom, model for 200 μm diameter.

The conclusion of these experiments is that there is a wealth of diameter-sensitive information in the data. The question now is whether the inverse problem, determining the diameter from the data, is possible, and if possible, if it is practical. Answering these questions affirmatively is aided by an understanding of the physical origins of the features this section has identified.

Building Intuition about the Method

The real and simulated data are essentially identical. We conclude that we know, at least to the level of our measurement precision, how to calculate what the apparatus will tell us. I will now show that we can also *understand* the physical origin of the major features of the data as characterized when the target-to-CCD distance exceeds a few millimeters:

1. There is a dip in light intensity that is
    - centered on the geometrical shadow, and
    - much broader than the geometrical shadow;
2. The center of the dip is somewhat brighter than the region a little off center;
3. The light intensity on each side of the dip is oscillatory with a broad structure that:
    - seems to the eye to be *independent of strand diameter,* (in spacing, although not so independent in intensity), and

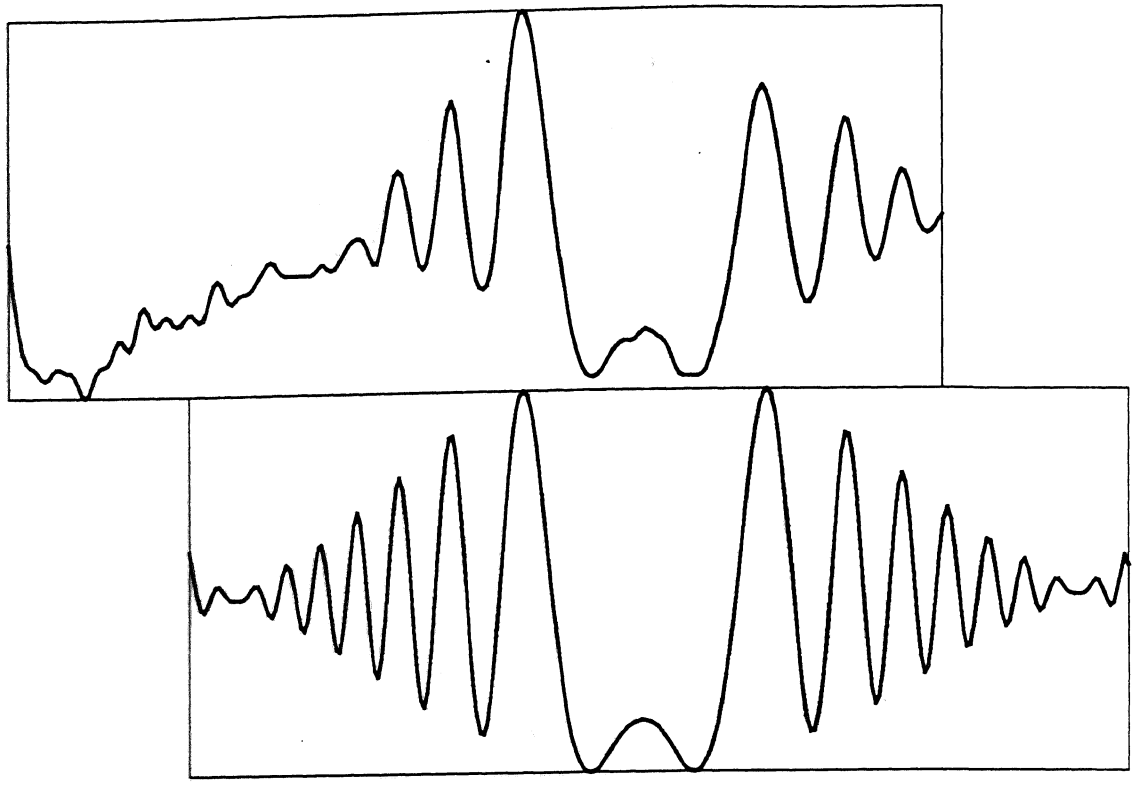

Figure 5. 38 AWG WIRE DATA

Top, data for awg 38 wire, approximately 101 μm diameter;
bottom, model for 100 μm diameter.

- in which the spatial frequency of the oscillations increases (the spacing decreases) with distance from the strand center;

4. Superimposed on the oscillatory structure is a fine structure that is:
   - *extremely sensitive to strand diameter,* and
   - whose spatial frequency (and thus the distance between manifestations) is a *constant* that depends inversely on diameter.

Of these features, the third one (the broad oscillations of increasing spatial frequency) is most important for the interim inversion procedure I have implemented. The fourth feature, the fine structure of constant, strongly diameter dependent spacing, promises to be most important for future, higher precision implementations.

Insight into the origin of the broad oscillations is obtained by studying the shadow of a knife edge, the bottom of Figure 7. The top of Figure 7 is the same as the model result previously shown in Figure 5. The broad oscillations of increasing spatial frequency seen in the data and model are thereby easily recognized as the shadows of the edges of the wire-like target: the primary structure is due to *diffraction,* and it depends on each edge essentially independently. The horizontal offset between the top and bottom parts of Figure 7 is essentially the measure of the right edge location with respect to the wire center. The offset shown was obtained by visually aligning the top and bottom parts of the figure. The offset measures 2 mm in a total of 128 mm figure width, corresponding to 256 pixel-widths of 13 μm each, or a wire radius of 52 μm, compared with the nominal value of 50 μm.

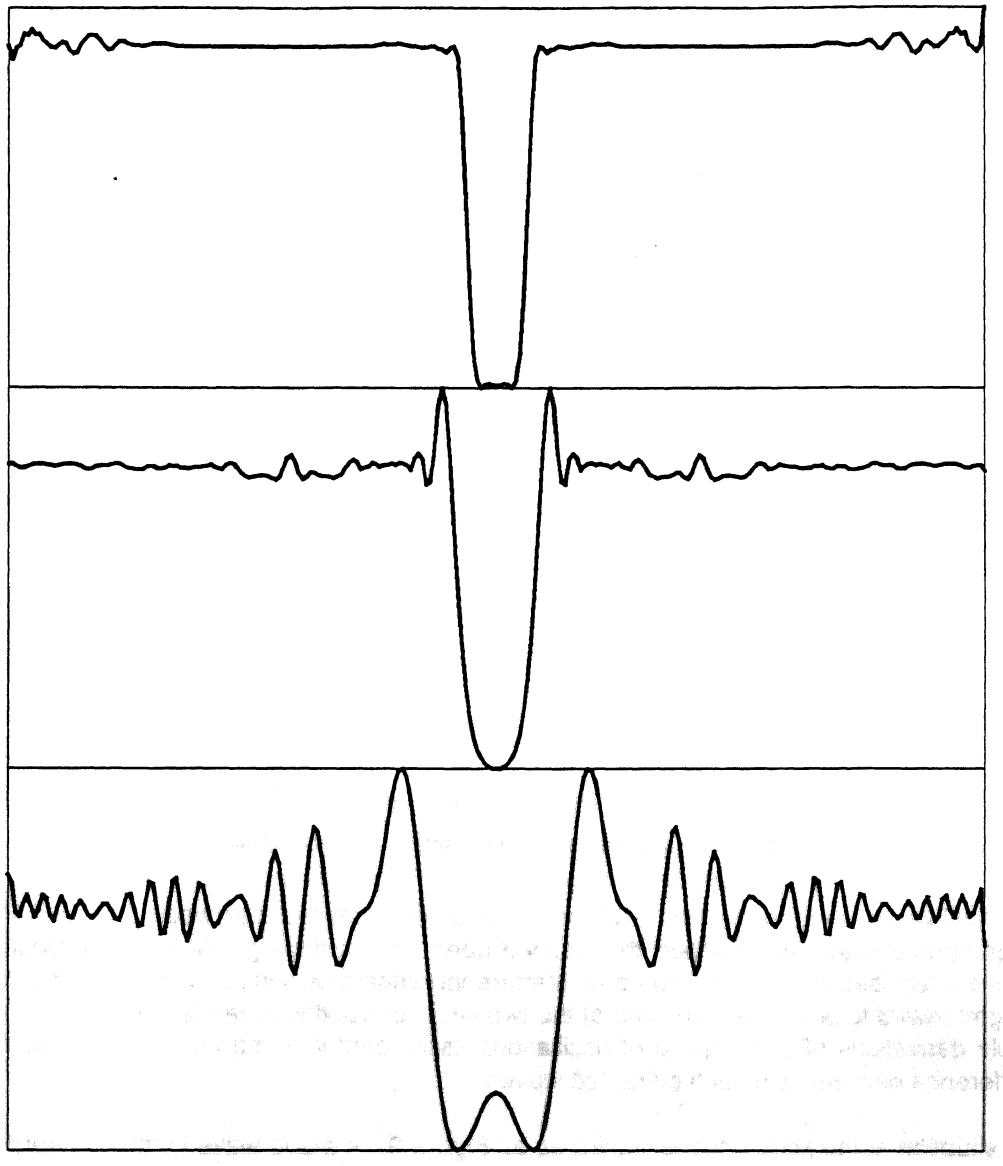

Figure 6. DIAMETER DEPENDENCE MODEL

The model for 200 μm diameter for target-to-CCD distance of *top*, 1 mm; *middle*, 10 mm; and *bottom*, 100 mm.

The remaining fine structure is an *interference* effect that depends on both edges simultaneously. Figure 8 shows the model result for two 1 μm wide strips 200 μm apart, *i.e.*, a two slit interference pattern with a slit separation the same as the target diameter, compared with the model result for a 200 μm diameter wire. Although generated for this demonstration by the *huygens* program, the two-slit part of Figure 8 is really just an offset cosine function whose wavelength is the product of distance and optical wavelength divided by target diameter. Comparing it with the model for the 200 μm wire, it is easily seen that each peak in the two-slit pattern corresponds precisely to an interference "wiggle" perturbing the broad diffraction structure of the wire. The fine structure, because it depends on both edges simultaneously, is exquisitely sensitive to the target diameter. But this sensitivity is difficult to exploit because the signal power associated with it is small relative to the signal power associated with the broad oscillations.

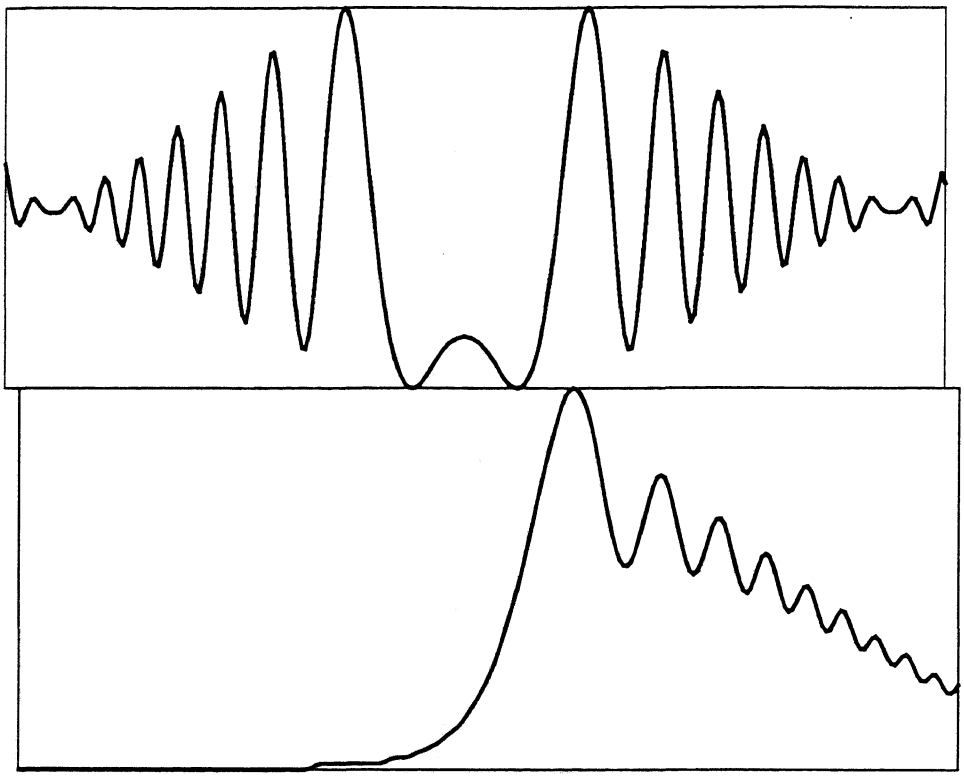

Figure 7. DIFFRACTION SHADOWS

Diffraction shadow of *bottom*, a knife edge, and *top*, a 100 μm diameter wire.

The intuitive picture I have given is plausible, and the agreement between the brute force numerical integrations and the data support the claimed deep understanding. While it is difficult to predict the relative intensities of the diffraction and interference effects by symbolic calculation, it is nevertheless straightforward to predict the spacing of the two kinds of oscillatory terms. I conclude this section with simple derivations of the spacing of oscillations associated with diffraction at a knife edge, and with interference between two such diffracted waves.

The situation at the knife edge is illustrated by Figure 9. A plane wave is moving from left to right. It is interrupted by the knife-edge toward the left side of the picture. In the upper half of the picture the plane wave continues on to the detector plane at the far right. A cylindrical Huygens' wavelet propagates into both halves of the space to the right of the knife-edge. The results are

- The cylindrical wavelet carries energy into the region that would be in the dark shadow space were optics geometrical;
- The interference between the plane waves and the cylindrical wave produces an oscillatory structure in the region that would be in the completely unshaded space were optics geometrical.

The geometry is simple: constructive interference, resulting in increased light intensity, occurs in the vicinity of regions where the path difference between plane and cylindrical waves is an even number of half wavelengths; destructive interference, resulting in decreased light intensity, occurs in the vicinity of regions where the path difference is an odd number of half wavelengths. For the small angles that are applicable the path difference between plane and cylindrical waves reaching y_{det} is

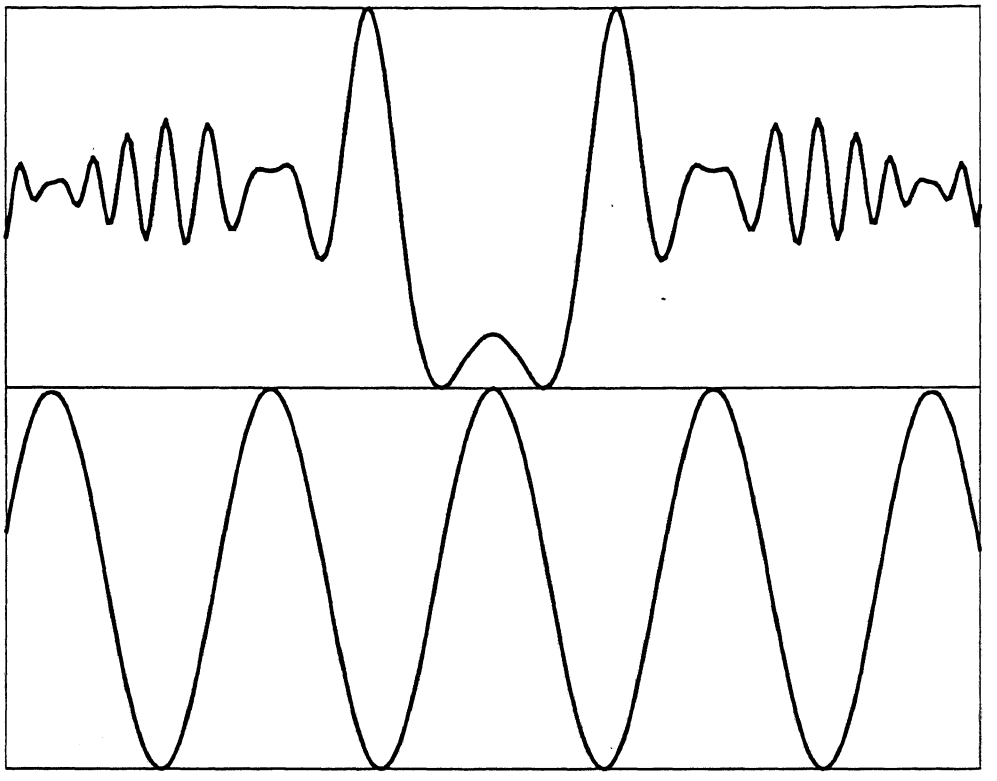

Figure 8. TWO SLIT INTERFERENCE PATTERN

Interference *bottom*, between two 1 μm wide slits 200 μm apart, compared with *top* model for a 200 μm diameter wire.

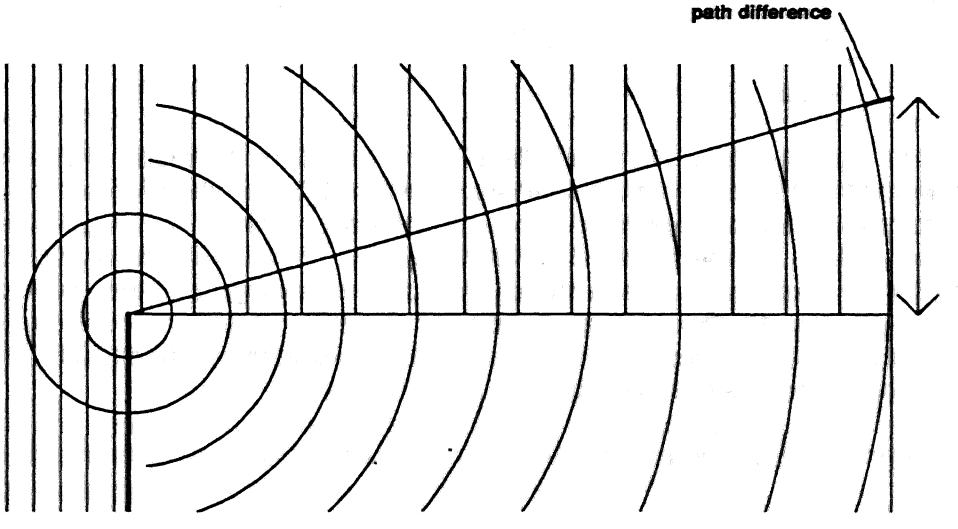

Figure 9. KNIFE-EDGE GEOMETRY

Geometry for calculating knife-edge diffraction pattern.

$$\Delta p = \frac{(y_{det}-y_R)^2}{2d}$$

for y_{det} to the right of a right edge, and

$$\Delta p = \frac{(y_L - y_{det})^2}{2d}$$

for y_{det} to the left of a left edge, where d is, as previously, the distance between the target plane and the detector plane. To the left of a right edge and to the right of a left edge there is only the cylindrical wave, no plane wave, so there are no interference oscillations in these regions, only a rapid falloff with angle corresponding more-or-less to the 1–cos θ obliquity factor.

The maxima and minima of the oscillatory structure outside the geometrically shaded region occur when $\Delta p = n\lambda/2$, at

$$\Delta y_{det} = \sqrt{2dn\lambda} \tag{2}$$

where Δy_{det} symbolizes either $(y_{det} - y_R)$ or $(y_L - y_{det})$ as the case may be. Even values of n are the peaks, odd values of n are the valleys, *measured from the location of the geometrical shadow* in the detector plane. Measuring, more naturally, from the center of the shadow of a wire-like target, half the target diameter is added to Δy_{det} for locations outside the geometrical shadow, and Δy_{det} is subtracted from half the target diameter for locations inside the geometrical shadow. Measurements of the peak and valley locations on data and simulations agree well with this model.

This treatment re-affirms that the primary oscillatory structure is a weak source of target diameter information, in that it is only the offset of this structure from target center location that measures diameter. On the other hand, it is a better measure than would be the location of a geometrical shadow, in that the edge location information is spread over many pixels, and can be measured to within a small fraction of a pixel-width by a mask or template matching algorithm. In contrast, the location of a geometrical shadow with sub-pixel-width precision has to be inferred from the gray level interpolation, a method that is probably in general less precise.

The geometry of two slit interference is illustrated by Figure 10. We consider the interference of the two cylindrical wavelets originating at the left and right edges of the target. This is the Young's two slit interference pattern problem for slit widths much smaller than slit spacing, where the "slit spacing" is our target diameter. For distances Δy_{det} not too far from the center of the interference pattern

$$\frac{\Delta y_{det}}{d} = \frac{\Delta p}{D} = \frac{n\lambda}{2D} \tag{3}$$

where in this case the target diameter D is an explicit factor in the result. Peaks and valleys are located at integer values of n, peaks at even n and valleys at odd n. What we see in reality, when this interference pattern is superimposed on the edge diffraction patterns, is a wiggle at each even value of n. At $n = 0$, *i.e.*, at the center of the diffraction pattern, interference is also constructive; this accounts for the bright spot at the center of each pattern, analogous to the well known bright spot found at the center of the shadow of a small disk or sphere [5]. The locations of the regularly spaced small wiggles in the real and simulated data are in excellent agreement with the prediction of Equation 3 for consecutive even values of n.

This interference structure is a tantalizing source of diameter measuring information for at least two reasons:

1. The peaks and valleys have a fixed spacing, corresponding to a fixed spatial frequency: their fourier transform is a uniformly spaced line spectrum, in contrast to the continuum spectrum generated by the edge diffraction pattern;
2. The spacing depends inversely on the diameter D, with correspondingly increasing relative sensitivity to smaller diameters.

The prototype instrument was built with neither sufficient shot-to-shot laser pattern reproducibility nor

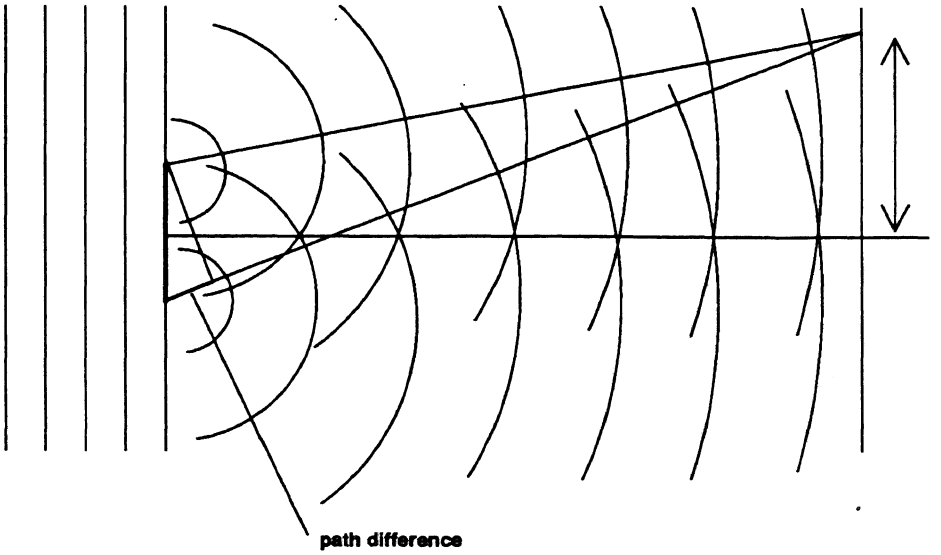

Figure 10. TWO SLIT GEOMETRY

Geometry for calculating two slit interference pattern.

sufficient analog-to-digital conversion resolution to be able to demonstrate this potential for precision interferometric measurement of diameter. I hope to remedy this in the future.

Some final words of caution are in order. First, the treatments of both single edge diffraction and two edge interference ignore issues of phase shifts on scattering; these complicated effects, which depend on the materials involved, *e.g.*, may be different for metals and insulators, sometimes have effects such as reversing the roles of even and odd values of n in Equations 2 and 3. Second, strictly speaking it is incorrect to separate single edge diffraction and two edge interference: there is a single scattering process, more-or-less described by Equation 1. Because the detection process destroys phase information, it is fundamentally impossible to dissect the result into two or more separate effects. Nevertheless, if one does not ask too many questions about details, particularly details of the intensity distribution, but instead restricts the discussion to the location of significant features such as peaks and valleys, this treatment is heuristically valid. In practice, the intuition generated by the model that separates single edge diffraction and two edge interference was essential to discovering an algorithm to extract the diameter and location information from the data.

The Diameter-Center Algorithm

The diameter and center are obtained via an algorithm that independently locates the left and right edges. The diameter is defined as the difference in edge locations and the center is defined as the mean of edge locations. The algorithm operates in several steps:

1. The digitized CCD data are smoothed, and in the process converted from integer[5] to floating point values, by a five point linearly weighted sliding average, as illustrated in Figure 11 for a 30 awg wire target.

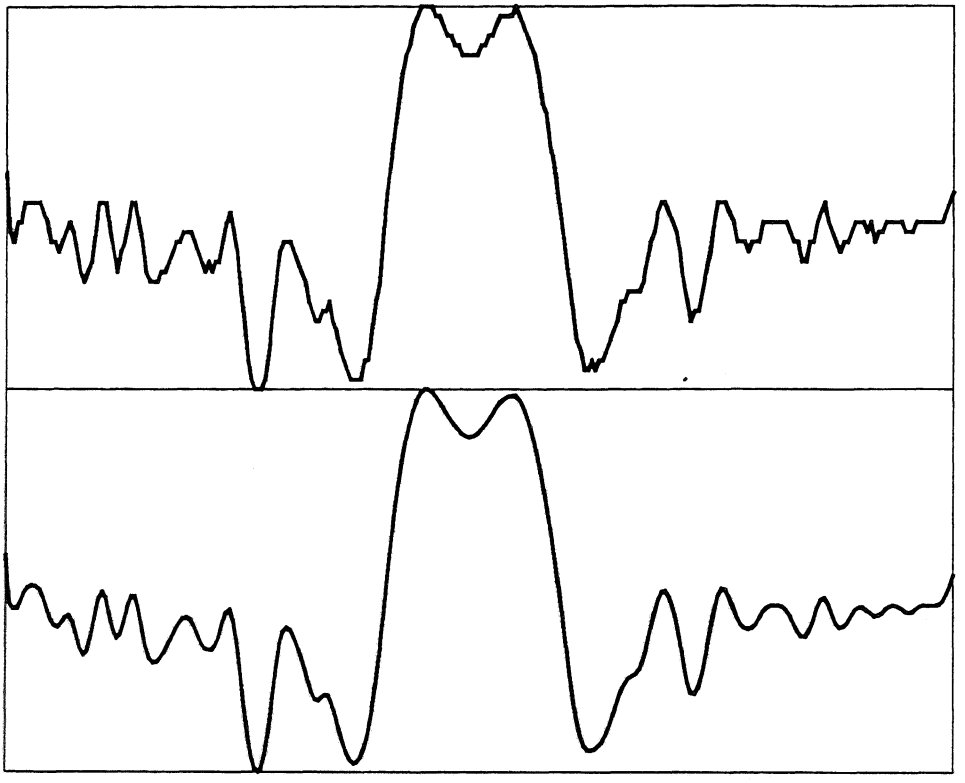

Figure 11. 30 AWG WIRE DIGITIZED DATA

Top, raw digitized CCD data, 30 awg wire, approximately 254 μm diameter (transmitted as twos-complement, therefore down is light, up is dark). *Bottom,* same data after five point linearly weighted sliding average.

[5]In the present 6-bit implementation, and the anticipated 8-bit implementation, it is efficient to actually store the raw data as character (byte) sized variables.

2. The smoothed data are subtracted from a stored floating point background[6] or *no wire* pattern obtained by averaging the pattern obtained from several (≈ 16) laser shots as illustrated in Figure 12 which shows the stored background, the data after background subtraction, and the model result for a 260 μm wire.

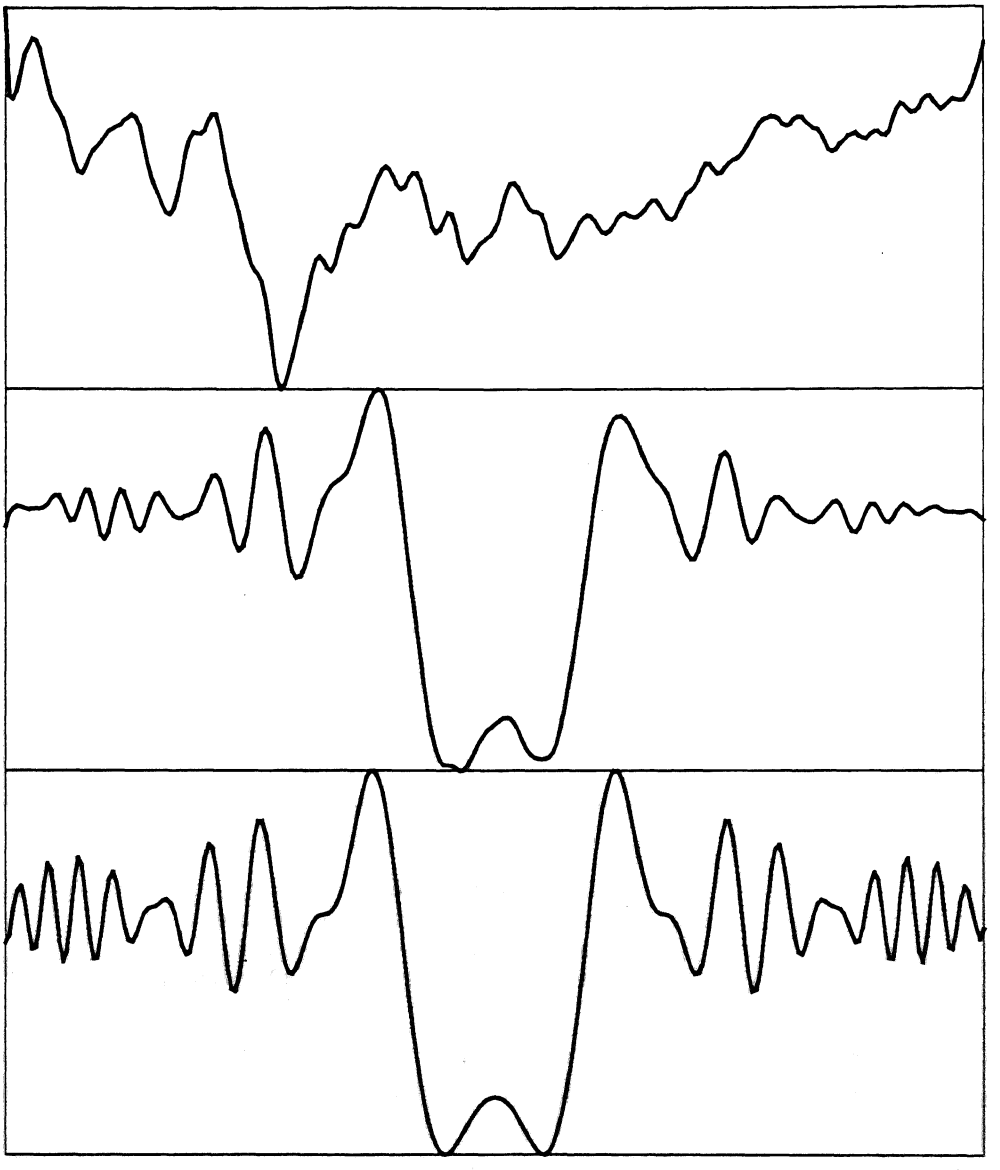

Figure 12. 30 AWG WIRE PROCESSED DATA

Top, stored background (*no wire*) file.
Center, data of Figure 11 after subtraction from the background.
Bottom, model result for 260 μm diameter wire.

[6]The data are subtracted from the background rather than *vice versa* because the digitizer is set up to deliver twos-complement data.

3. The background-subtracted data are convolved with a mask or template that is a heuristically selected portion of the diffraction shadow of a knife edge, Figure 7; the portion that is used begins two pixels to the right (higher pixel-numbers) of center and extends for 64 pixels. These convolutions, and the convolutions with the mirror-image of the mask, are illustrated in Figure 13.

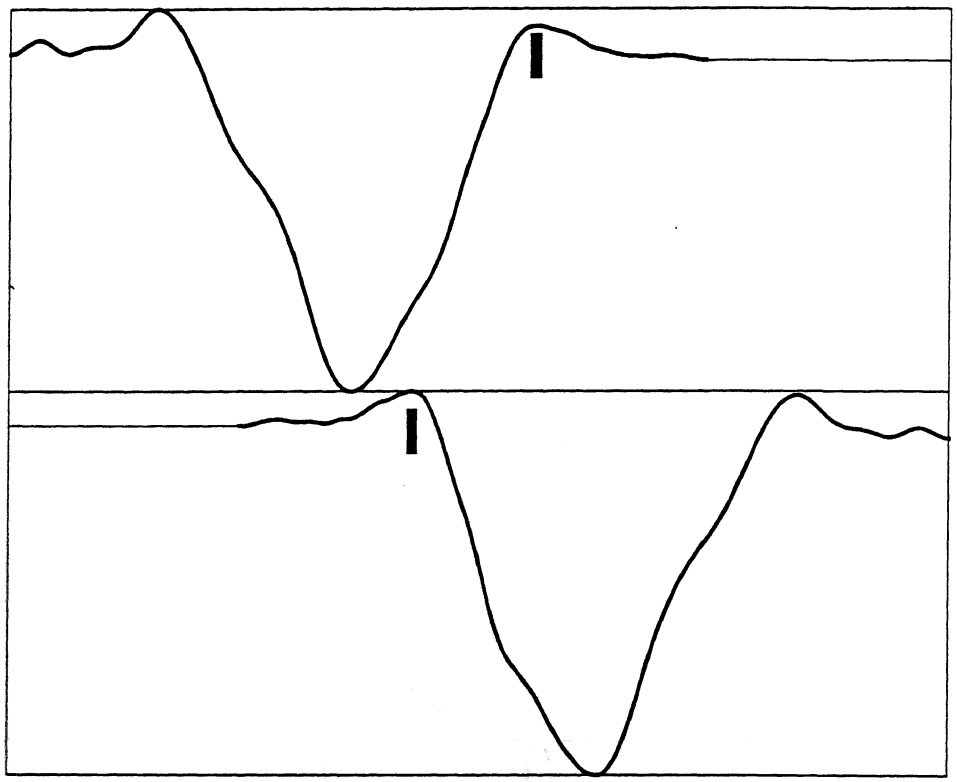

Figure 13. 30 AWG WIRE CONVOLUTIONS

Top, convolutions of a portion of the mask of Figure 7
with the smoothed, normalized data of Figure 12.
Bottom, same, with a mirror-image of the mask.
The heavy vertical bars mark nominal edge locations found by the algorithm.

4. The absolute minimum of the convolution is found, and then the first maximum to the right of the minimum is found; this maximum is marked by a heavy vertical bar in Figure 13.

5. A polynomial is fit to a seven-pixel-wide region centered on this maximum.

6. The right edge is defined as the fractional pixel-number at which this polynomial maximizes.

7. The left edge is found by a procedure that is the mirror-image of the four previous steps, as shown in the bottom part of Figure 13.

8. The diameter and center are obtained by respectively subtracting and averaging the two edge locations.

This procedure fails if the right edge falls within 64 pixels of the right end of the CCD or the left edge falls within 64 pixels of the left end of the CCD; in either of these cases an alternative, less accurate, procedure is invoked. The alternative procedure fits a polynomial to a seven-pixel-wide region

centered on the minimum of the convolutions. In a moderate range of diameters this minimum has an *approximately* fixed offset from the maximum actually sought.

Even the alternative procedure fails if an edge is off the CCD; when this happens the particular measurement is abandoned.

At the end of the procedure a preliminary calibration correction is invoked by a procedure that adjusts the diameter via a low order polynomial mapping of the diameter-center plane. The polynomial coefficients are found by running the algorithm on simulated data generated by the *huygens* program. The largest effect of the correction is to subtract a constant, approximately 12 pixel-widths.

Appendix C is the function *diacens* (and several functions that it uses) that implements this model, and Appendix B is the *include* file of instrument constants, program parameters, and structure definitions needed to understand parts of Appendix C.

We end this section by explicitly noting that the background subtraction procedure is heuristic; on strict theoretical grounds it is wrong. Equation 1 gives the electric-field amplitude and phase in the detector plane as an integral of the electric-field amplitude and phase over the un-occluded portion of the target plane. The background or *no wire* picture is essentially the square magnitude of this integral when there is no occlusion of any portion of the target plane. But taking the square magnitude, *i.e.*, translating from vector amplitudes (which add) to scalar intensities (which, for coherent illumination, cannot properly be added, but which are what the detector reports) permanently destroys the phase information that is needed to do a proper background correction. In fact, if the detection process *did not* destroy this phase information, the integral of Equation 1, with subscripts *det* and *src* interchanged, could be used to do the inversion *exactly*. The result would be the laser intensity pattern in the un-occluded part of the target plane, and blackness across the diameter of the target. As a practical matter, the procedure I have described is effective despite the fact that it casually treats intensities as if they could engage in the arithmetic that nature in fact has reserved for amplitudes.

Instrument Performance

The raw data, their trend line, and the standard deviation of their distribution for 1000 diameter-center measurement made in a single run lasting between 10 and 11 hours are shown in Figures 14 and 15. The target was a 30 awg wire.

The mean of the raw diameter measurements is 21.504 pixel-widths, or 279.5 μm, in contrast to the actual diameter of 254.2 μm; a multiplicative calibration factor having no significant bearing on any performance measures would correct this discrepancy. The spread in the raw measurements is from a maximum of 22.67 pixel-widths to a minimum of 20.66 pixel-widths; most of the skew in the distribution is due to the trend, which could (but will not) be subtracted out. The standard deviation is 0.36 pixel-widths, which is the appropriate error to assign to a single measurement. The relative standard deviation is 1.67%. Averaging n measurements would reduce these values by $\sqrt{n}$, *e.g.*, the average of 100 measurements would be expected to be correct within 0.036 pixel-widths, or 0.167%.

The mean of the raw center measurements is pixel-number 128.00; since there is no independent measure of the accuracy of the centering this value is entirely nominal; its only interest is as a fiducial value with respect to which trend and scatter can be evaluated. The spread in the raw measurements is from pixel-number 127.57 to pixel-number 128.38; as in the case of the diameter measurement, most of the skew in the distribution is due to the trend, which could (but will not) be subtracted out. The standard deviation is 0.131 pixel-widths. This is the appropriate error to assign

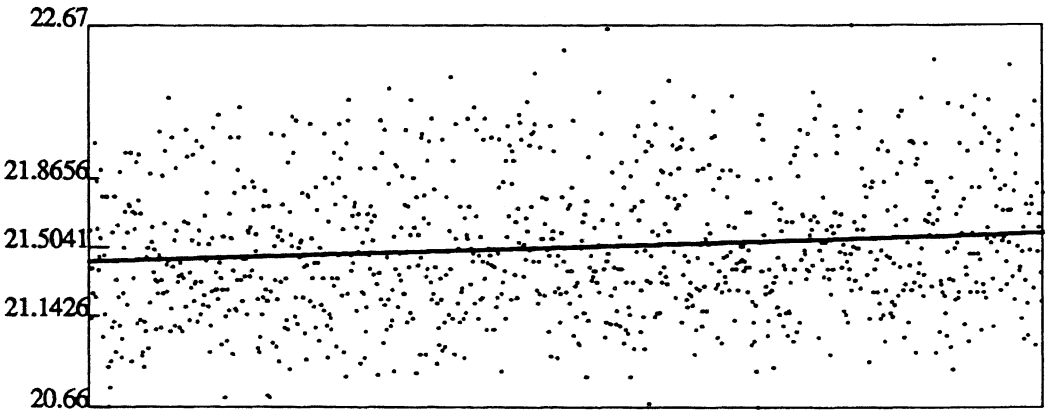

Figure 14. DIAMETRIC SCATTER PLOT

Scatter plot for 1000 diameter measurements over approx. 10 hrs.

to a single measurement. Averaging n measurements would reduce this value by $\sqrt{n}$, *e.g.*, the average of 100 measurements would be expected to be correct within 0.0131 pixel-widths.

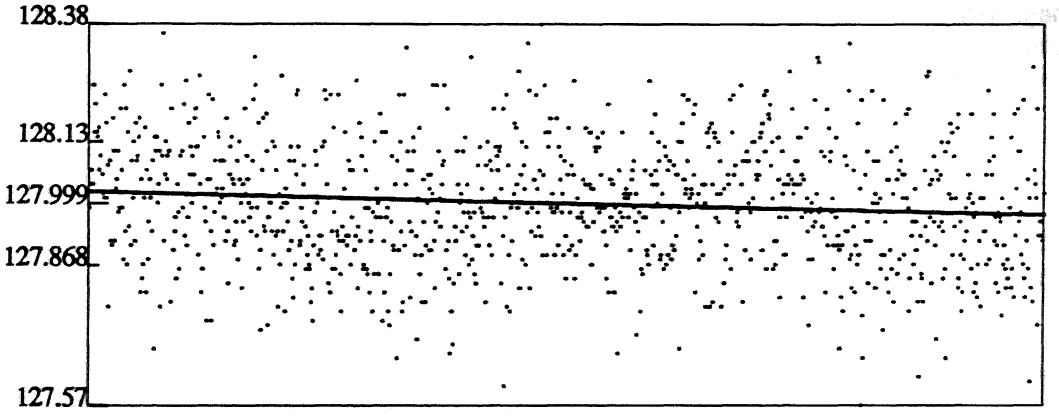

Figure 15. CENTRIC SCATTER PLOT

Scatter plot for 1000 center measurements over approx. 10 hrs.

For both the diameter and the center measurements the trend, or long term drift, accounts for about 20% of the the total error, and the remaining 80% is, to visual inspection, reasonably random shot-to-shot noise. Both the long term drift and the shot-to-shot noise, I believe, are due primarily to the laser. Drift in the mean laser pattern (the stored background or "no wire" pattern, Figure 12), which in the present implementation is recorded only once before the start of data collection, I believe accounts for most of the trend. Shot-to-shot fluctuation in the laser pattern is substantial, as much as 25% in some parts of the pattern, presumably because the laser is operated fairly close to threshold, and I believe that the shot-to-shot noise is almost entirely due to this fluctuation.

The long term drift could be substantially reduced by periodically moving the wire out of the laser beam and revising the stored background pattern; this could be automated easily. Alternatively, if the mean value of the measurement were known via other measurements to be time invariant, the trend line could simply be subtracted out. The background revision alternative is obviously preferable.

The shot-to-shot noise could be substantially reduced by introducing a spatial filter (focusing lens, pinhole, and re-collimating lens); the pattern emerging from the spatial filter would continue to show

global intensity fluctuations, but its spatial pattern would be stabilized to essentially the gaussian TEM_{00} mode. Furthermore, to compensate for the light lost in the spatial filter the laser would have to be operated at higher power, thus reducing the global intensity fluctuations.

The noise is so strongly dominated by these laser-related factors that it is impossible to say what are the next largest sources of technical noise. It seems likely that adding a spatial filter will reduce the drift and noise levels by an order-of-magnitude or more.

Summary and Next Steps

The diffraction shadow examined in this report can be productively though of as containing two distinct sources of metrological information: the independent shadows of the right and left edges, and the interference pattern between the two diffraction patterns. The shadows have been shown to be suitable for finding the edges via a mask or template matching operation with the calculated (or alternatively, measured) shadow of a knife edge. This method, since it uses the information from the two edges independently, is error prone and of relatively low precision compared with what is potentially available in the interference pattern, whose spatial frequency depends explicitly and delicately on the diameter itself. However substantially more signal power is available in the edge shadows than in the interference pattern. With the low signal-to-noise and coarse analog-to-digital converter implemented in the prototype described, using the edge shadows has proven to be the expedient alternative.

Using the edge shadow method, the prototype apparatus has demonstrated the ability to make diameter measurements with a standard deviation (expected error of a single measurement) of 0.36 pixel-widths, or 4.7 μm, for wire-like targets in the 100 to 400 μm diameter range. The corresponding standard deviation for the center location measurement is 0.13 pixel-widths, or 1.7 μm. The laser flash duration is approximately 200 nsec, the CCD readout time could be as short as 100 nsec per channel, or 25.6 μsec per line; video flash digitizers that could follow this 10 MHz data rate are easily available. Thus, computation times aside, measurements could be repeated at a rate approaching 40,000 sec^{-1}. While a general purpose computer could not begin to keep up with this repetition rate, special purpose hardware, *e.g.*, an array processor card, could easily come within an order of magnitude of keeping up in real time. When n measurements are averaged the uncertainty of the average is the uncertainty of a single measurement divided by $\sqrt{n}$; thus with a 0.25 second averaging time we could ideally obtain a factor of 1000 improvement, *i.e.*, 4.7 nm diameter measurement and 1.7 nm center location measurement. In short, sub-μm precision is easily obtainable by averaging a few dozen measurements.

A more interesting alternative to the brute-force approach of making and averaging many measurements in a short time is to look for opportunities to extract and use the interference information that is now inaccessible. I propose to attack this problem with a combined hardware and software approach. First the shot-to-shot reproducibility should be stabilized, *e.g.*, by spatial filtering of the laser beam, modest improvements in optics, and modestly improved mechanical stability; after this is done it will become productive to increase the digitization resolution. Second, residual drift due to temporal changes in the laser intensity pattern can be minimized by periodically updating the stored background picture, in contrast to the practice to date of storing this picture once at the beginning of a data run. Finally, with increased signal amplitude resolution, several approaches to extracting the interference signal will be worth trying. These would be primarily fourier transform methods, in which, perhaps after preliminary processing to subtract the edge shadow signal components, we would look for the comb of spatial frequency peaks in the fourier spectrum that correspond to the target diameter.

Acknowledgements

Much of this work was supported by PPG Industries, Inc., Fiber Glass Research Center, Pittsburgh, PA 15238. Reed Grundy of PPG-FGRC provided valuable constructive criticism that continually re-sharpened our approach to the practical problem. The implementation was by Alan Guisewite, Mark Hahn, Thomas Chanak and Usha Swaminathan at CMU. Roman Kuc of Yale checked our model via an alternative computational route, and contributed to many valuable discussions during his sabbatical at CMU.

References

1. Robert C. Weast (Ed.). Wire Table, Standard Annealed Copper. In *Handbook of Chemistry and Physics*, CRC Press, Boca Raton, FL, 1979.

2. Benincasa, D., Barber, P., Zhang, J., Hsieh, W., Chang, R. "Spatial distribution of the internal and near-field intensities of large cylindrical and spherical scatterers". *Applied Optics 26*, 7 (April 1987), 1348-56.

3. Gilliar, W., Bickel, W. S., Videen, G., and Hoar, D. "Light scattering from fibers: an extension of a single-slit diffraction experiment". *Am. J. Phys. 55* , 6 (June 1987), 555-9.

4. Fairchild Camera and Instrument Company. CCD: The Solid State Imaging Technology. Manufacturer's catalog.

5. E. U. Condon and H. Odishaw. Optics: Diffraction and Interference. In *Handbook of Physics*, McGraw-Hill, New York, 1958, Chap. 6.

6. Sorab K. Ghandhi. Lithographic Processes: Printing and Engraving. In *VLSI Fabrication Principles: Silicon and Gallium Arsenide*, John Wiley & Sons, New York, 1983, Chap. 10, pp. 544-6.

7. Joseph W. Goodman. Foundations of Scalar Diffraction Theory. In *Introduction to Fourier Optics*, McGraw-Hill, San Francisco, 1968, Chap. 3, pp. 34-7.

8. Sanders Associates, Inc. High Power Laser Diodes. Manufacturer's data sheet.

9. O. Robert Mitchell, Edward P. Lyvers, Kirk A. Dunkelberger and Mark L. Akey. Recent Results in Precision Measurements of Edges, Angles, Areas and Perimeters. Automated Inspection and Measurement, SPIE, Cambridge, MA, October, 1986, pp. 123-34.

10. Francis T. S. Yu. The Fresnel-Kirchoff Theory or Huygens' Principle. In *Introduction to Diffraction, Information Processing, and Holography*, The MIT Press, Cambridge, MA, 1973, Chap. B (appendix), pp. 359-60.

Appendix A: The *HUYGENS* Program

```
/*************************************************************************
  huygens.c: huygens [arg1 arg2]
  calculate amplitudes using Huygens' Principle as modified by Kirchoff

  CALCULATES THE RELATIVE COMPLEX AMPLITUDE SEEN AT EACH CELL OF A
  LINEAR CCD THAT INTERCEPTS A COLLIMATED LASER BEAM INTERUPTED BY A
  WIRE-LIKE OPAQUE OBJECT.  THE CORRESPONDING SIGNAL IS THE SQUARE
  MAGNITUDE OF THE OUTPUT OF THIS PROGRAM: (ampl.r^2 + ampl.i^2)

  M. W. Siegel - Robotics Institute - Carnegie Mellon University
  May 29, 1986
  Copyright, the author and maybe PPG Industries, pending decision as to
  the author's right to designate it as being in the public domain,
  which is where he wants it to be.

  (1) no  arguments: create wavefcns and distfcns,
                     integrate from -LENS_RADIUS to +LENS_RADIUS,
                     producing output file "unobstru";

  (2) two arguments: read back wavefcns and distfcns,
                     subtract out the region between (and including)
                     the two arguments;

  (3) [add later]: if wavefcns are symmetric, take this into account;

  (4) [add later]: if wavefcns are pure real or pure imaginary, take this
      into account.

  Note: It is useful to dimension arrays of (2*n-1), i.e.,
        x[-(n-1):(n-1)], as follows:

<type> *x;
x = (<type> *)calloc(2*n-1, sizeof(<type>)) + n - 1;

  Given:
  (1) a lens of radius LENS_RADIUS, from which emerges
  (2) a collimated beam of gaussian profile, exp(-(y_src/y00)^2),
  (3) times a parabolic shading (1 - (y_src/LENS_RADIUS)^2)
  (4) to make a smooth transition to zero intensity, and thus to eliminate
      structure from the diffraction pattern of the lens aperture;
  and
  (5) a ccd consisting of CELLS cells, CELL_SIZE microns center-to-center,
  (6) located such that Center_Cell = atoi(argv[1]) is on the
      axis of the lens, and thus on the center line of the beam,
  and
  (7) a wire that obstructs the beam DISTANCE microns from the plane of
      the ccd,
  (8) whose left edge is at Left_Edge = atoi(argv[2]) microns from
      the axis,
  (9) and whose right edge is at Right_Edge = atoi(argv[3]) microns from
      the axis;

 (10) FOR EXAMPLE:  huygens 116 -50 150
      calculates the line image response of the ccd whos 116th cell is on
      the lens axis, when obstructed by a wire whose left edge
```

```
      is 50 microns to the left of the lens axis, and whose right edge is
      150 microns to the right of the lens axis.

The program works as follows:

(1) physical and geometrical parameters of the apparatus are DEFINEed;
(2) some more constants are calculated;
(3) the wavefront amplitude is specified at 1 micron intervals from the
    lens axis to the lens radius: wavefcn[LENS_RADIUS];
(4) a phase, 1/r distance, and (1 + cos(angle)) obliquity product function
    is specified at 1 micron spatial increments of the offset between
    source point (y_src) and detector point (y_ccd); since this offset
    can be between zero (y_src = y_ccd) and LENS_RADIUS + CELLS * CELL_SIZE
    microns (from the left edge of the lens to the right edge of the
    ccd when Center_Cell = 0, or vice versa, from the right edge of the
    lens to the left edge of the ccd when Center_Cell = CELLS - 1):
    distfcn[LENS_RADIUS + CELLS * CELL_SIZE].
(5) the wavefcn[] and distfcn[] are independent of the specific location
    of the ccd, or the location of the ccd or the wire, in any application;
    thus once the arithmetic of calculating these is done, the results are
    written onto files whose names are hard-wired to "wavefcns" and
    "distfcns" in the directory from which huygens is executed;
(6) if the files "wavefcns" and "distfcns" exist in the at run time, they
    are read into wavefcn[] and distfcn[], thus eliminating tedious
    re-calculation.

CAUTION: obviously, I hope, if the DEFINEs are changed and the program
         recompiled, it is important that the files "wavefcns" and
         "distfcns" are ABSENT from the local directory the first time
         the recompiled program is run!

(7) for each ccd cell, located at y_ccd,
(8) integrate in one micron intervals over CELL_SIZE microns in the
    ccd plane
(9) and in one micron intervals from -LENS_RADIUS to +LENS_RADIUS,
(10) except for the values of y_src obstructed by the wire.

Then print the result in some useful format, e.g.,
an amplitude file and a signal file.

*************************************************************************/

#include <stdio.h>
#include <math.h>

#define DISTANCE                167000.0        /* microns, wire to CCD */
#define WAVELENGTH              0.895           /* microns */
#define LENS_RADIUS             (int)2500       /* microns */
#define CELL_SIZE               (int)13         /* microns */
#define SUB_CELL_SIZE           (int)5          /* microns */
/*********************************
  The active width of each pixel
*********************************/
#define CELLS                   (int)256
/*************************************************************************
  The calculation will be done between -(CELLS - 1) and +(CELLS - 1)
*************************************************************************/
```

```
#define NA                      0.1             /* numerical aperture */
#define FWHM                    30.0            /* deg laser divergence */

main    (argc, argv)
int     argc;
char    *argv[];
{
COMPLEX *wavefcn, *distfcn, *amp;
/*********************************************************
 Note typedef of COMPLEX should be struct {double r, i}
*********************************************************/
int     scratch,
        Lens_Radius = (int) LENS_RADIUS,
        Cells       = (int) CELLS,
        Cell_Size   = (int) CELL_SIZE,
        Left_Edge   = atoi(argv[1]),
        Right_Edge  = atoi(argv[2]),
        y_elems     = Lens_Radius + Cells * Cell_Size;
char    *calloc(),
        out_name[32];
FILE    *fopen(), *out_fp;

if (argc != 1 && argc != 3)
  {
  fprintf(stderr,
  "Usage: huygens [Left_Edge Right_Edge]\n");
  exit();
  }
scratch = (argc == 1) ? 1 : 0;

wavefcn = (COMPLEX *)calloc(2*Lens_Radius - 1, sizeof(COMPLEX)) +
                                Lens_Radius - 1;
get_wavefcns(wavefcn);

distfcn = (COMPLEX *)calloc(2*y_elems - 1, sizeof(COMPLEX)) +
                                y_elems - 1;
get_distfcns(distfcn);

amp = (COMPLEX *)calloc(2*Cells - 1, sizeof(COMPLEX)) +
                            Cells - 1;
calculate_amp(wavefcn, distfcn, Left_Edge, Right_Edge, amp, scratch);

if (scratch) sprintf(out_name, "unobstru");
else         sprintf(out_name, "L%dR%d.amp", Left_Edge, Right_Edge);
out_fp = fopen(out_name, "w");
fwrite(amp - Cells + 1, sizeof(COMPLEX), 2*Cells - 1, out_fp);
fclose(out_fp);
}

get_wavefcns(wavefcn)
COMPLEX *wavefcn;
{
FILE    *wave_file, *fopen();

if ((wave_file = fopen("wavefcns", "r")) == NULL)
  {
  double        lens_radius     = (double) LENS_RADIUS,
```

```
                fwhm            = (double) FWHM,
                na              = (double) NA,
                pi, y00, yy0, yy1, yy2;
  int           Lens_Radius     = (int) LENS_RADIUS,
                y_src;

  pi  = 4.0 * atan(1.0);
  y00 = lens_radius*tan(0.5*fwhm*(pi/180.0))*sqrt(-log(0.5))/tan(asin(na));

  for (y_src = -Lens_Radius + 1; y_src < Lens_Radius; ++y_src)
    {
    yy0 = (double)y_src;
    yy1 = yy0/y00;
    yy2 = yy0/lens_radius;
    (wavefcn + y_src)->r = exp(-yy1 * yy1) * (1.0 - yy2 * yy2);
    (wavefcn + y_src)->i = 0.0;
    }
  wave_file = fopen("wavefcns", "w");
  fwrite(wavefcn-Lens_Radius+1,sizeof(COMPLEX),2*Lens_Radius-1,wave_file);
  }
else
  {
  int   Lens_Radius = (int)LENS_RADIUS;

  fread(wavefcn-Lens_Radius+1,sizeof(COMPLEX),2*Lens_Radius-1,wave_file);
  }
fclose(wave_file);
}

get_distfcns(distfcn)
COMPLEX *distfcn;
{
FILE    *dist_file, *fopen();

if ((dist_file = fopen("distfcns", "r")) == NULL)
  {
  int           Lens_Radius     = (int) LENS_RADIUS,
                Cells           = (int) CELLS,
                Cell_Size       = (int) CELL_SIZE,
                y_elems         = Lens_Radius + Cells * Cell_Size,
                y_src_ccd;
  double        distance        = (double) DISTANCE,
                wavelength      = (double) WAVELENGTH,
                pi, k00, pathlen, dist, dsq, cosphi, sinphi, kirchf;


  pi  = 4.0 * atan(1.0);
  k00 = 2.0 * pi / wavelength;
  dsq = distance * distance;

  for (y_src_ccd = 0; y_src_ccd < y_elems; ++y_src_ccd)
    {
    dist = (double)y_src_ccd;
    pathlen = sqrt(dsq + dist * dist);
    cosphi = cos(k00 * pathlen);
    sinphi = sin(k00 * pathlen);
    kirchf = (1.0 + distance/pathlen)/pathlen;
```

```
    (distfcn + y_src_ccd)->r = (distfcn - y_src_ccd)->r = cosphi * kirchf;
    (distfcn + y_src_ccd)->i = (distfcn - y_src_ccd)->i = sinphi * kirchf;
    }
  dist_file = fopen("distfcns", "w");
  fwrite(distfcn - y_elems + 1, sizeof(COMPLEX), 2*y_elems-1, dist_file);
  }
else
  {
  int           Lens_Radius     = (int) LENS_RADIUS,
                Cells           = (int) CELLS,
                Cell_Size       = (int) CELL_SIZE,
                y_elems         = Lens_Radius + Cells * Cell_Size;

  fread(distfcn - y_elems + 1, sizeof(COMPLEX), 2*y_elems - 1, dist_file);
  }
fclose(dist_file);
}

calculate_amp(wavefcn, distfcn, Left_Edge, Right_Edge, amp, scratch)
COMPLEX *wavefcn, *distfcn, *amp;
int     Left_Edge, Right_Edge, scratch;
{
int     Cells           = (int) CELLS,
        Cell_Size       = (int) CELL_SIZE,
        Sub_Cell_Size   = (int) SUB_CELL_SIZE,
        Sub_Left        = -(Sub_Cell_Size/2),
        Sub_Right       =  Sub_Cell_Size/2 + 1,
        Lens_Radius     = (int) LENS_RADIUS,
        cell, subcell, y_ccd, y_src;
COMPLEX ampl, *d_y_ccd;

if (scratch)
  {
  Left_Edge  = -Lens_Radius + 1;
  Right_Edge =  Lens_Radius - 1;
  }
else
  {
  FILE *unobstru_fp, *fopen();
  unobstru_fp = fopen("unobstru", "r");
  fread(amp - Cells + 1, sizeof(COMPLEX), 2*Cells - 1, unobstru_fp);
  }

for (cell = -Cells + 1; cell < Cells; ++cell)
  {
  ampl.r = 0.0;
  ampl.i = 0.0;
  y_ccd = cell * Cell_Size;
  for (subcell = Sub_Left; subcell < Sub_Right; ++subcell)
    {
    d_y_ccd = distfcn - y_ccd - subcell;
    for (y_src = Left_Edge; y_src <= Right_Edge; ++y_src)
      {
      ampl.r += (d_y_ccd + y_src)->r * (wavefcn + y_src)->r -
                (d_y_ccd + y_src)->i * (wavefcn + y_src)->i;
      ampl.i += (d_y_ccd + y_src)->r * (wavefcn + y_src)->i +
                (d_y_ccd + y_src)->i * (wavefcn + y_src)->r;
```

```
        }
      }
    if (scratch)
      {
      (amp + cell)->r = ampl.r;
      (amp + cell)->i = ampl.i;
      }
    else
      {
      (amp + cell)->r -= ampl.r;
      (amp + cell)->i -= ampl.i;
      }
    }
}
```

Appendix B: The *INCLUDE* File of Constants and Definitions

```
#include <stdio.h>
#include <math.h>

#define BIGG            1.0e9
#define TINY            1.0e-9
#define PI              3.141592654

#define CCDS            4
#define CELLS           256
#define AMPCELLS        511     /* for making the mask from huygens calc */
#define ADC_RANGE       63      /* 63 dark, 0 saturated light */

#define MIN_AREA        1.0     /* square-ccds */
#define DIA_ERR         10.0    /* ccd */
#define CEN_ERR         2.0     /* ccd */

#define MASKLEN         64      /* convolution mask length */
#define MASKZER         2       /* mask offset */
#define HOW_GUTSY       16      /* willingness to risk */
                                /* intgr math overflow */
#define YWEIGHT         0.001   /* relative weight of center */
                                /* vs. diameter errors in */
                                /* the correction algorithm  */

#define CAL_TABLE       "cal"   /* root name of calibration files */
#define CEN_TABLE       "cen"   /* root name of centering files */
#define UNOBSTRU_FILE   "nw"    /* "no wire"  */
#define NO_LASER_FILE   "nl"    /* "no laser" */
#define MASK_FILE       "mask"  /* need I say more? */

#define TTY             "/dev/tty00"
#define BAUD            B9600
#define ENQ             5       /* CELLS data points sent on receipt */

#define SPACING         13.0            /* microns */
#define DISTANCE        167000.0        /* microns */

#define GM_PER_CM3      2.53
#define PACKING_FRAC    0.91

#define GRAPH_NONE      0       /* these nine are enum-s in the  */
#define GRAPH_SOME      1       /* main program,                 */
#define GRAPH_ALL       2       /* but I had difficulty passing  */
                                /* them to subroutines as such   */
#define PAUSE_NONE      0       /* and rather that fight it now  */
#define PAUSE_SOME      1       /* I pass them as integers       */
#define PAUSE_ALL       2

#define CALIB_LIVE      0
#define CALIB_AUTO      1
#define CALIB_MANU      2

struct  DC      {float  d, c;};
struct  POINT   {float  x, y;};
struct  LINE    {float  m, b;};
```

```
struct  PLANE   {float cx, cy, cz;};          /* z = cx*x + cy*y + cz */
struct  PTPAIR  {struct POINT corr, meas;}; /* correct and measured */
struct  DPP     {float dist;                  /* (corr - meas)^2      */
                 struct POINT corr, meas;};

typedef struct {double r, i;} COMPLEX;
```

Appendix C: The *DIACENS* Diameter-Center Algorithm Program

```
/*
  diacens()
  given (char) cdata, returns (dc) diameter-center pairs
*/

/*************************************************************************
*  M. W. Siegel - Robotics Institute - Carnegie Mellon University        *
*                                                                        *
*  Copyright 1987 and earlier development versions, the author and       *
*  possibly PPG Industries, pending decision as to the author's right    *
*  to designate it as being in the public domain, which is where he      *
*  wants it to be.                                                       *
*************************************************************************/

#include        "yardage.h"

diacens(cdata, dc, graph_mode, pause_mode, calib_mode)
unsigned char   *cdata;
struct  DC      *dc;
int             graph_mode, pause_mode, calib_mode;
{
register int    leng, conv, *inte, *mask, cell, convmin;
char            caption[80];
int             ccd, rcell, rmin, lcell, lmin,
                INTENS [CELLS],
                CONV   [CELLS],
                VNOC   [CELLS];
double          redge, ledge, centr, xpeak();
struct DC       precal();
static double   radix;
static int      nevercalled = 1,
                MASK   [AMPCELLS];
static float    NO_WIRE[CCDS][CELLS],
                NO_LASE[CCDS][CELLS],
                FNTENS [CELLS];

if (nevercalled)
  {
  FILE  *fp_in;
  float ftemp;

  nevercalled = 0;

  radix = HOW_GUTSY *  /* HOW_GUTSY: willingness to risk overflow */
          pow(2.0, 0.5 * 8.0 * sizeof(int)) /
          (ADC_RANGE * sqrt((double)(CELLS - MASKLEN) * MASKLEN));

  /* get and optionally show the mask */
  fp_in = fopen(MASK_FILE, "r");
  for (cell = 0; cell < AMPCELLS; ++cell)
    {
    fscanf(fp_in, "%f", &ftemp);
    MASK[cell] = ftemp * radix + 0.5;
    }
  fclose(fp_in);
```

```
  if (graph_mode == GRAPH_ALL)
    {
    graph(&MASK[AMPCELLS/2], 'i', AMPCELLS/2, '*', 1, "mask (right half)");
    await_cr(pause_mode);
    }

  /* get and optionally show the background (no_laser) file */
  fp_in = fopen(NO_LASER_FILE, "r");
  for (ccd = 0; ccd < CCDS; ++ccd)
    for (cell = 0; cell < CELLS; ++cell)
      fscanf(fp_in, "%f", &NO_LASE[ccd][cell]);
  fclose(fp_in);
  if (graph_mode == GRAPH_ALL)
    for (ccd = 0; ccd < CCDS; ++ccd)
      {
      sprintf(caption, "ccd %1d: no_laser file", ccd);
      graph(NO_LASE[ccd], 'f', CELLS, '*', 1, caption);
      await_cr(pause_mode);
      }

  /* get and optionally show the unobstructed (no_wire) file */
  fp_in = fopen(UNOBSTRU_FILE, "r");
  for (ccd = 0; ccd < CCDS; ++ccd)
    for (cell = 0; cell < CELLS; ++cell)
      fscanf(fp_in, "%f", &NO_WIRE[ccd][cell]);
  fclose(fp_in);
  if (graph_mode == GRAPH_ALL)
    for (ccd = 0; ccd < CCDS; ++ccd)
      {
      sprintf(caption, "ccd %1d: no_wire file", ccd);
      graph(NO_WIRE[ccd], 'f', CELLS, '*', 1, caption);
      await_cr(pause_mode);
      }
  }

/*
   WARNING: does not include any NO_LASER corrections,
   because I am not really sure how it should be done,
   or even if it should be done at all!
 */
for (ccd = 0; ccd < CCDS; ++ccd, ++dc, cdata += CELLS)
  {
  /* smooth, convert to floats, normalize the input data (chars) */
  fsmooth(cdata, FNTENS, graph_mode, pause_mode, ccd);

  /* correct for background */
  subnorm(FNTENS, NO_WIRE[ccd], graph_mode, pause_mode, ccd);

  /* convert to integer arithmetic */
  for (cell=0; cell<CELLS; ++cell) INTENS[cell]=FNTENS[cell]*radix+0.5;

  /* take convolutions to find the right edge */
  for (cell = 0; cell < CELLS - MASKLEN; ++cell)
    {
    conv = 0;
    leng = MASKLEN;
    mask = &MASK[AMPCELLS/2] + MASKZER;
```

```
  inte = INTENS + cell;
  while ( leng-- ) conv += *inte++ * *mask++;
  CONV[cell] = conv;
  }
for (cell = CELLS - MASKLEN; cell < CELLS; ++cell) CONV[cell] = conv;

convmin = CONV[0];
for (cell = 1; cell < CELLS - MASKLEN; ++cell)
  if (CONV[cell] < convmin) convmin = CONV[rmin = cell];
cell = (rcell = rmin) + 1;
while (CONV[rcell] < CONV[cell++]) ++rcell;
redge = rcell + xpeak(&CONV[rcell]);

/* take convolutions to find the left edge */
for (cell = MASKLEN; cell < CELLS; ++cell)
  {
  conv = 0;
  leng = MASKLEN;  /*cell<MASKLEN ? cell : MASKLEN;*/
  mask = &MASK[AMPCELLS/2] + MASKZER;
  inte = INTENS + cell;
  while ( leng-- ) conv += *inte-- * *mask++;
  VNOC[cell] = conv;
  }
conv = VNOC[MASKLEN];
for (cell = 0; cell < MASKLEN; ++cell) VNOC[cell] = conv;

convmin = VNOC[0];
for (cell = MASKLEN; cell < CELLS; ++cell)
  if (VNOC[cell] < convmin) convmin = VNOC[lmin = cell];
cell = (lcell = lmin) - 1; while (VNOC[lcell] < VNOC[cell--]) --lcell;
ledge = lcell + xpeak(&VNOC[lcell]);

/* do some special stuff if close to a ccd end */
if      ( rmin <= MASKLEN )        /* then the LEFT edge is garbage */
  {
  dc->c = 0.5 * (rmin + xpeak(&CONV[rmin]) + lmin + xpeak(&VNOC[lmin]));
  dc->d = 2.0 * (redge - dc->c);
  *dc = precal(*dc);
  }
else if ( lmin > CELLS - MASKLEN) /* then the RIGHT edge is garbage */
  {
  dc->c = 0.5 * (rmin + xpeak(&CONV[rmin]) + lmin + xpeak(&VNOC[lmin]));
  dc->d = 2.0 * (dc->c - ledge);
  *dc = precal(*dc);
  }
else                               /* this is the normal case */
  {
  dc->c = 0.5 * (redge + ledge);
  dc->d = redge - ledge;
  *dc = precal(*dc);
  }

centr = dc->c;

if ( graph_mode || calib_mode == CALIB_MANU )
  {
  sprintf(caption, "ccd %1d: diameter %7.3f, center %7.3f",
```

```
                          ccd,          dc->d,        dc->c);
    graph(  VNOC, 'i', CELLS, '-', 1, "");
    graph(  CONV, 'i', CELLS, '+', 0, caption);
    graph_mark((float)ledge, -1, '>');
    graph_mark((float)centr, -1, '|');
    graph_mark((float)redge, -1, '<');
    await_cr(pause_mode);
    if ( calib_mode == CALIB_MANU )
      {
      graph(INTENS, 'i', CELLS, '*', 0, caption);
      manu_cal(*dc, ccd);
      }
    }
  }/* end of ccd counting loop */
return;
}

fsmooth(cdata, intens, graph_mode, pause_mode, ccd)
/*************************************************************************/
/* smooth character data to float, and perhaps graph it                  */
/*************************************************************************/
unsigned char   *cdata;
float           *intens;
int             graph_mode, pause_mode, ccd;
{
int     cell;

*(intens + 0) = *(cdata + 0);
*(intens + 1) = (*(cdata + 0) +
          2.0 * *(cdata + 1) +
                *(cdata + 2))/4.0;
for (cell = 2; cell < CELLS - 2; ++cell)
  *(intens + cell) = (*(cdata + cell - 2) +
                2.0 * *(cdata + cell - 1) +
                3.0 * *(cdata + cell    ) +
                2.0 * *(cdata + cell + 1) +
                      *(cdata + cell + 2))/9.0;
*(intens + CELLS - 2) = (*(cdata + CELLS - 3) +
                  2.0 * *(cdata + CELLS - 2) +
                        *(cdata + CELLS - 1))/4.0;
*(intens + CELLS - 1) = *(cdata + CELLS - 1);

if (graph_mode == GRAPH_ALL)
  {
  char  caption[80];
  sprintf(caption, "ccd %1d: smoothed data", ccd);
  graph(intens, 'f', CELLS, '*', 1, caption);
  await_cr(pause_mode);
  }
return;
}

subnorm(intens, nowire, graph_mode, pause_mode, ccd)
/*************************************************************************/
/* subtract from a background pattern, and perhaps graph it              */
/*************************************************************************/
float   *intens, *nowire;
```

```
int     graph_mode, pause_mode, ccd;
{
register int cell;

cell = CELLS;
while ( cell-- )
  {
  *intens = *nowire - *intens;
  ++intens; ++nowire;
  }
intens -= CELLS;

if (graph_mode == GRAPH_ALL)
  {
  char  caption[80];
  sprintf(caption, "ccd %1d: background subtracted and normalized", ccd);
  graph(intens, 'f', CELLS, '*', 1, caption);
  await_cr(pause_mode);
  }
return;
}

*/
  xpeak(): interpolation for maximum or minimum in an array

  deriv(): derivatives

  fib(): fibonacci numbers
*/

double xpeak(y)
int *y;
/*************************************************************************
  Given *(y - 1) < *y > *(y + 1)  or  *(y - 1) > *y < *(y + 1),
  xpeak(y) returns the value of x (-1.0 < x < 1.0) at which the
  maximum or minimum "actually" occurs, based on an expansion
  fitting the region under the curve from *(y - DY) to *(y + DY),
  i.e., it will consider derivatives of y up to d(2*DY)y/dx(2*DY)
  and powers of x up to x^(2*DY).

  The procedure (Newton's x(n+1) = x(n) - y'(n)/y''(n)) is iterated until
  successive values of x differ by less than DX.

*************************************************************************/
{
#define DX 0.001
#define MI 6
double y1, y2, y3, y4, y5, y6, a, b, x, x1, x2, x3, x4, deriv();
int iters;

y1 = deriv(y, 1, 1.0);
y2 = deriv(y, 2, 1.0);
y3 = deriv(y, 3, 1.0);
y4 = deriv(y, 4, 1.0);
y5 = deriv(y, 5, 1.0);
y6 = deriv(y, 6, 1.0);
```

```
if ( y1 == 0.0 || y2 == 0.0 ) return 0.0;

x = -y1/y2;
iters = MI;
do {
   x1 =    x;
   x2 = x1*x;
   x3 = x2*x;
   x4 = x3*x;
   if ((a = y1 + y2*x1 + y3*x2 + y4*x3 + y5*x4) == 0.0) break;
   if ((b = y2 + y3*x1 + y4*x2 + y5*x3 + y6*x4) == 0.0) break;
   x -= a/b;
   } while ( fabs (x - x1) > DX && iters-- );
return x;
}
/* generalized version of xpeak():
{
#define DY 3
#define DX 0.001
double x, dx, deriv(), taylor(), der[2*DY + 1], tay1, tay2; int    i;

for (i = 0; i <= 2*DY; ++i) der[i] = deriv(y, i, 1.0);

x = dx = 0.0;
while ( (tay2 = taylor(der, 2*DY+1, 2, x)) != 0.0 &&
         fabs(dx = -taylor(der, 2*DY+1, 1, x)/tay2) > DX ) x += dx;
return x;
}
*/

double deriv(y, n, dx)
int *y;
int           n;
double       dx;
/*****************************************************************************
  Takes y, a pointer to an element of an array of doubles, and
  returns the n-th derivative in the vicinity of that element.
  The elements of y are separated by dx, i.e., the horizontal
  distance between (y + N) and (y + N - 1) is dx.

  if ( dx <= 0.0 dx == 1.0 ) then the result is reported without
  normalizing by dx^n, making the arithmetic a little faster.

  REQUIRES: fib(n, m), a Fibonacci number generator.

  WARNING: it is the caller's responsibility that (y - (n+1)/2) through
  (y + (n+1)/2) be valid addresses containing members of array y.

  WARNING: severe rounding errors may start to appear around n = 8.
*****************************************************************************/
{
int *y1, *yr;
double dny_dxn;
int    fib(), m = 0, sign = 1, n_2;

if ( n < 0  ) err_exit("deriv: cannot take derivative of order < 0.");
```

```
if      ( n == 0 ) dny_dxn = *y;
else
  {
  if ( n %  2 ) /* odd n, even number of points, so symmetrize */
    {
    n_2 = (n+1)/2;
    dny_dxn = (double) *(yr = y + n_2) - (double) *(yl = y - n_2);
    while (sign = -sign, ++m < n_2)
      dny_dxn += sign * (fib(n, m - 1) - fib(n, m)) *
                        ((double) *++yl - (double) *--yr);
    dny_dxn *= 0.5;
    }
  else                /* even n, odd number of points, is symmetric */
    {
    n_2 = n/2;
    dny_dxn = (double) *(yl = y - n_2) + (double) *(yr = y + n_2);
    while (sign = -sign, ++m < n_2)
      dny_dxn += sign * fib(n, m) * ((double) *++yl + (double) *--yr);
    dny_dxn += sign * fib(n, m) * (double) *y;
    }
  }
if ( dx > 0 && dx != 1.0 ) while ( n-- ) dny_dxn /= dx;
return dny_dxn;
}


int fib(n, m)
int n, m;
/*****************************************************************************
  Fibonacci number generator; on the assumption that the first few
  values are needed most often, these are stored in a static array.
*****************************************************************************/
{
#define EIGHT 8
static int Fib[EIGHT][EIGHT] = {{   1,   0,   0,   0,   0,   0,   0,   0},
                                {   1,   1,   0,   0,   0,   0,   0,   0},
                                {   1,   2,   1,   0,   0,   0,   0,   0},
                                {   1,   3,   3,   1,   0,   0,   0,   0},
                                {   1,   4,   6,   4,   1,   0,   0,   0},
                                {   1,   5,  10,  10,   5,   1,   0,   0},
                                {   1,   6,  15,  20,  15,   6,   1,   0},
                                {   1,   7,  21,  35,  35,  21,   7,   1}};

if (n < 0 || m > n) err_exit("fib: called with n < 0 or m > n.");

return (m == 0 || m == n) ? 1          :
       (n <  EIGHT      ) ? Fib[n][m] :
                            fib(n-1, m-1) + fib(n-1, m);
}

struct DC precal(dc)
struct DC dc;
{
struct DC dc_;
double x1, x2, x3, x4, y1, y2, c0, c1, c2, c3, c4;

y1 = dc.c;
```

```
y2 = y1*y1;

c0 = -2.7583e01 + 2.1286e-01 * y1 - 8.3575e-04 * y2;
c1 =  1.5949e00 - 8.1579e-03 * y1 + 3.2038e-05 * y2;
c2 =  0.0       + 0.0        * y1 + 0.0        * y2;
c3 =  0.0       + 0.0        * y1 + 0.0        * y2;
c4 =  0.0       + 0.0        * y1 + 0.0        * y2;


x1 = dc.d;
x2 = x1*x1;
x3 = x1*x2;
x4 = x1*x3;

dc_.d = c0 + c1 * x1 + c2 * x2 + c3 * x3 + c4 * x4;
dc_.c = dc.c;

return dc_;
}
```