Real-Time Vergence Control

Thomas J. Olson   Robert D. Potter

The University of Rochester
Computer Science Department
Rochester, New York 14627

Technical Report 264

November 1988

Abstract

Binocular robots whose cameras can be independently directed require some mechanism for aiming both cameras at the same world point. We describe a mechanism for verging the cameras of the Rochester Robot in real time. The mechanism consists of a discrete control loop driven by an algorithm that estimates a single disparity from the two cameras. We present two algorithms for disparity estimation. The first uses the cepstral-transform approach of Yeshurun and Schwartz [1987]. We argue that in this application the cepstrum is best understood as autocorrelation with an adaptive filter that acts to sharpen peaks in the autocorrelation image. We show that qualitatively similar filters have similar effects, with the limiting case being equivalent to deconvolution. We describe efficient real-time implementations of the cepstral and deconvolution approaches.

This work was supported in part by the U.S. Army Engineering Topographic Laboratories under research contract no. DACA76-85-C-0001, by NSF research grant no. DCR-8602958, by NIH Public Health Service research grant no. 1 R01 NS22407-01, and by ONR research contract no. N00014-82-K-0193. In addition, this work was supported by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700, and the Air Force Office of Scientific Research, Bolling AFB, DC 20332 under Contract No. F30603-85-C-0008. This contract supports the Northeast Artificial Intelligence Consortium (NAIC).

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TR 264	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Real-Time Vergence Control		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Thomas J. Olson and Robert D. Potter		8. CONTRACT OR GRANT NUMBER(s) DACA76-85-C-0001 N00014-82-K-0193
9. PERFORMING ORGANIZATION NAME AND ADDRESS Dept. of Computer Science 734 Computer Studies Bldg. Univ. of Rochester, Rochester, NY 14627		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS DARPA / 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE November 1988
		13. NUMBER OF PAGES 20
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* Office of Naval Research Information Systems Arlington, VA 22217		15. SECURITY CLASS. *(of this report)* Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Distribution of this document is unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Vergence	Phase Correlation
Cepstrum	
Deconvolution	
Autocorrelation	
Disparity	

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Binocular robots whose cameras can be independently directed require some mechanism for aiming both cameras at the same world point. We describe a mechanism for verging the cameras of the Rochester Robot in real time. The mechanism consists of a discrete control loop driven by an algorithm that estimates a single disparity from the two cameras. We present two algorithms for disparity estimation. The first uses the cepstral-transform approach of Yeshurun and Schwartz (1987). We argue that in this application the cepstrum is best understood as autocorrelation with an adaptive filter that acts to sharpen peaks in

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20. ABSTRACT (Continued)

the autocorrelation image. We show that qualitatively similar filters have similar effects, with the limiting case being equivalent to deconvolution. We describe efficient real-time implementations of the cepstral and deconvolution approaches.

1 Introduction

Recently a significant amount of work in computer vision has focused on the problems of acting, behaving systems, and in particular on how "active vision" differs from analysis of static scenes or vision with fixed cameras [Aloimonos *et al.*, 1987; Bandopadhay, 1986; Ballard, 1987]. In many cases it turns out that giving a vision system the power to move around in its environment simplifies many previously intractable problems. Over the summer of 1988 the Rochester vision group began development of an integrated facility for the study of vision, AI and systems issues related to active vision. The facility will be described in some detail in section 2. Briefly, it consists of an industrial robot arm bearing a custom-built "head". The head has two CCD television cameras which can be moved together in altitude (pitch) and independently in azimuth (yaw). The head, arm and cameras are connected to a pipelined image processor, a workstation and a set of large-scale parallel processors.

A major goal of our research is the development of a real-time gaze control system. We believe that the robot must be able to maintain fixation on world points and change gaze to new ones without the intervention of high level "cognitive" faculties. To this end, we are developing a set of quasi-reflexive gaze control mechanisms that compensate for known ego motions (similar to the human vestibulo-ocular reflex), maintain fixation on moving objects, make saccadic movements to targets selected by some external process, and verge the eyes so that their optic axes intersect at a surface.

We envision the gaze control modules forming a layered control structure along the lines described by Brooks [1987; 1986]. The details of the control structure and module interactions are a current research topic, but the individual modules are fairly well developed. This paper describes the internal workings of the module responsible for maintaining vergence. In the next section we describe the robot's environment and capabilities. We then discuss reasons for verging and describe our general approach. In section 4 we describe a verging algorithm based on the cepstrum. Section 5 contains a discussion of why the cepstral approach works, developing conclusions which are used to derive a better algorithm. We conclude by discussing plans for extending the vergence system.

2 The Rochester Robot

The philosophy and goals of the Rochester Robot project are detailed in [Brown, 1988], along with a detailed description of the hardware and preliminary research results. Here we will summarize this material in just enough detail to motivate the work to be described in later sections.

The centerpiece of the University of Rochester Robotics and Vision Laboratory is the robot head/body assembly, shown in figure 1. The "body" is a PUMA 761 six degree-of-freedom robot arm made by Unimation, Inc. It can place the head virtually anywhere inside a sphere with a radius of two meters, and can move it at a maximum speed of 1 meter per second. The head (figure 2), built as a joint project with the University's Mechanical Engineering Department, has two RS-170 CCD cameras feeding a DataCube MaxVideo™

pipelined image processor. The cameras are moved by three stepper motors, one of which controls the pitch of the camera platform while the other two control the yaw angles. The cameras thus have roughly the same degrees of freedom as human eyes. The stepper motors have an angular resolution of 0.14 degrees per step and a maximum speed of 400 degrees per second.

Figure 3 shows the computational hardware and communication links that make up the laboratory. Operators typically control the robot from the console of the Sun 3/260, from which they can watch the robot through a large plexiglas window[1]. Standard UNIX^{TM} network tools make it possible to run programs on the Butterflies, LISP machines or other Suns from the Sun 3/260 if desired. One of the Butterflies has direct access to the Sun's VME bus, and hence can access the image processor and motor drivers directly.

The MaxVideo system [Datacube, 1987] consists of a set of VME-based boards that can be cabled together via a separate digital image bus to form an image processing pipeline. Functions available as pipeline stages include digitization and storage, convolution, histogramming, and point operations on one or two images. The MaxVideo system also includes EUCLID, a single-board computer based on the ADSP-2100 digital signal processing microprocessor [Analog Devices, 1987]. We used EUCLID extensively for the work presented here, so we will describe it in some detail.

In addition to the CPU, the EUCLID board contains 32K words of data memory, 16K words of program memory and various bus interfaces. The ADSP-2100 runs at 8 Mhz and has separate data and instruction busses; most instructions execute in one 125 nS cycle. Its instruction set and functional units are heavily optimized for digital signal processing applications. Half of the data memory is dual-ported to the VME bus, and supports zero wait state access by the ADSP-2100 concurrently with access by the Sun. The other half of the data memory is dual-ported to the digital image bus, so that regions of interest can be transferred in at video rates.

An important limitation of the ADSP-2100 is that the data memory address space is only 16K 16-bit words long. On the EUCLID board this space is divided into 16 1K-word pages. Fifteen of these pages can be mapped to 1K-word blocks in either half of local data memory, or to 1K-word or 1K-byte blocks of VME memory. The remaining page contains mapping and control registers and scratchpad memory. Because of the small address space, it is difficult to manipulate large data objects. For the disparity computations described below, this limits sample density to 32×32 samples per image. Higher sample densities could be implemented by using remote memory, but this would slow the system down significantly.

The EUCLID board is programmed using a C cross-compiler, cross-assembler and linker running on the Sun host. Unfortunately the C compiler, while very useful for prototyping, does not make efficient use of the ADSP-2100's rather exotic architecture. In order to obtain real-time performance it is usually necessary to convert all inner loops to microcode subroutines. Fortunately the DataCube libraries provide a number of examples of well-written microassembly codes, which can be used as models.

[1]For safety reasons, the room containing the robot is normally kept locked, and the doors to the room are equiped with switches that cut power to the robot when the doors are opened.

3 The Vergence Problem

For primates like ourselves the need for a vergence mechanism is obvious. Human eyes have non-uniform resolution, so we need a way to direct both foveas at the same world point so as to extract the greatest possible amount of information about it. The human brain has an extraordinary ability to extract depth information from stereo pairs, but only if the disparities fall within a limited range. Verging on surfaces usually constrains points near the fixation point to fall inside this range.

The Rochester Robot's cameras do not have foveas. Even so, there are good reasons to have a low-level mechanism that maintains vergence. As Ballard argues [Ballard, 1987], having a unique fixation point defines a coordinate system which is related as much to the object being observed as it is to the observer, and hence is a step in the direction of an object-centered coordinate system. Verging the eyes also provides an invariant that may be useful to higher level processes. It guarantees that the depth of at least one world point is known, even if we do not attempt stereo reconstruction in the usual sense.

3.1 Vergence on the Rochester Robot

The vergence control mechanism of the Rochester Robot assumes that one eye is designated as dominant. That is, one of the two eyes is assumed to be under the control of the saccadic or tracking systems. The vergence system's job is to control the other eye so as to make the optic axes of the two eyes intersect at or near a surface. To do this, it repeatedly executes the simplest possible control loop: it grabs left and right images using the digitizer, estimates the disparity in pixels, converts the disparity to an angular error using an empirically derived calibration table, and commands the non-dominant eye motor to null out the error. It waits for the camera motion to complete and then repeats. In all tests to date the "dominant" eye has in fact been fixed. When the gaze control system is finally integrated, some changes in the control mechanism will be necessary; these will be discussed in section 6.

Disparity measurement has been studied extensively in the context of stereo depth reconstruction [Barnard and Fischler, 1982]. Unfortunately the best stereo methods are not very suitable for the real-time vergence application. The most sophisticated methods tend to be based on a search through matching space for an assignment of depths that is optimal under some cost metric. They are usually expensive to compute, and (what is perhaps worse for a real-time application) their running time is usually data-dependent. Since they depend on optimizing global criteria, it is hard to talk about estimating single disparities.

For real-time vergence what is needed is a simple algorithm that estimates a single disparity in a fixed amount of time. This narrows the field by and large to image processing methods such as cross-correlation. Past attempts to use such methods for stereo depth recovery have uncovered many problems (see [Horn, 1986] for a review.) We have found, however, that two operators that are closely related to correlation work quite well for vergence. These operators are described in sections 4 and 5.

Correlation-like disparity estimators work by comparing sample windows taken from the left and right images. The first step in using them is to decide what size sample windows to compare. If the sample windows are too large they are likely to contain objects at different depths, which may lead to ambiguous results. If they are too small, two problems arise. First, they may not contain enough detail to provide an adequate basis for the disparity computation. Second, they may not overlap at all, in which case no disparity computation will be possible.

After choosing sample windows one must decide how densely to sample them. A low sampling rate will limit the angular resolution of the disparity estimate. Furthermore, if the sampling rate is not high enough to satisfy the Nyquist criterion then the left and right images may contain spurious frequencies whose phase depends on the position of the sample window. In this case the disparity operator may produce meaningless results. In general, denser sampling will improve the signal-to-noise ratio and angular resolution at the cost of more work for the algorithm.

We have so far made no attempt to optimize the window size and sample density parameters for either of our algorithms. Window size selection is an interesting research problem, which we plan to explore in a later study. Any solution will certainly involve an adaptive algorithm that adjusts the window size continously to suit changing conditions. In the experiments described below, the sample windows were usually the entire left and right camera images. As for sample density, early empirical studies showed that reducing the 512×512 images to 32×32 by subsampling produced adequate results. This was fortunate, since using larger sample arrays would have greatly complicated the task of porting the algorithms to EUCLID.

Our work to date, then, has consisted of developing fast disparity measures for 32×32 subsampled versions of the original image. The next two sections describe our results.

4 Computing disparity using the cepstral filter

Yeshurun and Schwartz [1987] describe a one-pass algorithm for estimating the disparity between two image patches using the cepstral filter. We have successfully used this algorithm to verge the cameras of the Rochester Robot. A single disparity estimate for the left and right camera images is used to adjust the camera yaw angles to minimize the horizontal disparity. In theory the vertical disparity should be zero, but small errors can arise due to camera miscalibration, imprecision in the camera positioning system et cetera.

To compute a single disparity value, the cepstral procedure is:

- extract sample windows of size $h \times w$ from the left and right images. Splice them to form a single image of size $h \times 2w$.
- Compute the 2–D Fourier transform of the resulting image.
- Use the result to compute the power spectrum of the image, and take the log of each pixel.
- Take the 2–D Fourier transform of the result.
- Find the peak value in a subregion of the resulting image.

The peak values occur at $(\pm(w + d_h), \pm d_v)$, where d_h and d_v are the estimated horizontal and vertical disparities.

Since the vergence control loop requires only one disparity estimate per image pair, the cost of the cepstral filtering approach is not necessarily prohibitive. However, even single cepstral disparities are hard to compute at anything close to real-time rates. For example, the Sun 3/260 that serves as system controller for the Rochester Robot requires approximately 2.6 seconds to compute one disparity estimate. By analysing the algorithm and implementing carefully on special-purpose hardware we have been able to compute disparities in less than a tenth of a second, which is more than adequate for our system. The remainder of this section describes our approach and experiences.

4.1 Efficient computation of the cepstrum

The derivation of the cepstral filter given in [Yeshurun and Schwartz, 1987] is phrased in terms of continuous functions. As stated earlier, it is based on Fourier transforms of an $h \times 2w$ image formed by splicing two windows together. However, simply applying a 2-D FFT to this image will not work. A discrete Fourier transform of size $h \times 2w$ can only handle frequencies in the range $-h/2$ to $h/2$ vertically and $-w$ to w horizontally. Positive horizontal disparities will produce peaks falling outside this range. Since discrete FFTs are circular, positive disparities will wrap around to the opposite ends of the horizontal frequency axis. The result will be that the magnitude of the disparity can be recovered, but not its sign.

The simplest solution to the ambiguity problem would be to widen the spliced image by padding with zeros. For 32×32 sample windows, for example, one might pad the image to 32×128 by appending 32×32 arrays of zeros to each end. However, given that time and memory are scarce resources, this solution is less than satisfactory. The extra work does nothing to improve the signal-to-noise ratio; its sole purpose is to provide enough frequency range to handle the expected output.

It turns out that the range problem can be gotten around in another way, at least for this application. Recall the previous discussion of the effect of applying a 2D FFT to an unpadded cepstral input image. If the horizontal disparity is positive, the output peak that should have appeared at $(w + d_h, d_v)$ will wrap around to the other end of the spectrum, appearing instead at $(-(w - d_h), d_v)$. At the same time, the peak that should have appeared at $(-(w + d_h), -d_v)$ will appear at $(w - d_h, -d_v)$. The transform output will then be indistinguishable from that produced by disparities of $(-d_h, -d_v)$. But now suppose that (as in the case of vergence or stereopsis) the vertical disparity is known to be small – specifically, that it is in the interval $\pm h/4$. The disparity peaks can be "tagged" by introducing an artificial vertical disparity of $+h/4$. To do this, transform the right-hand sample window by moving row m to row $(m+h/4) \bmod h$. Discrete Fourier transforms wrap around at their borders, so no samples will be lost and the signal-to-noise ratio will not suffer. After the cepstral transform, locate peaks in the usual way. Since the vertical disparity should be positive, a negative vertical disparity indicates that the horizontal disparity has wrapped around, and that the correct disparities can be obtained by flipping the signs of the measured disparities.

The main body of the cepstral algorithm consists of a 2-D FFT, a point transform (the log of the power spectrum), and a second 2-D FFT. A multidimensional FFT is performed by applying 1-D FFTs along each dimension of the input array. In-place FFT algorithms require either the inputs or the outputs of the transform to appear in scrambled (bit-reversed) order. The overhead of scrambling inputs or unscrambling outputs can be avoided by adopting a strategy used in linear filtering. First transform each column of the input using a decimation-in-frequency (DIF) FFT that expects normally ordered input and produces scrambled output. Since each column is scrambled identically, this leaves the data correctly ordered within each row; the rows simply appear in a different order. Next, transform each row by the same DIF FFT. This completes the first 2-D FFT, but leaves the data scrambled in a very complex way. However, taking the log of the power spectrum is a point operation, so the fact that the points are out of order is unimportant. The second 2-D FFT is performed using a decimation-in-time algorithm that expects scrambled inputs and produces normally ordered outputs. Applying this transform first to the rows and then to the columns undoes the scrambling produced by the first FFT, leaving us with a correctly ordered output image.

Computing the power spectrum from the Fourier transform is trivial, but taking the pointwise logarithm at first seems likely to be expensive – especially if it is to be done on an integer machine with a small address space, such as EUCLID. The standard derivation of the cepstral filter uses the natural log (base e), but the base is immaterial to the algorithm; all that matters is that $\log ab = \log a + \log b$, which is true for any base. It is possible to estimate $\log_2 n$ (in fixed point, for any integer $n > 2$) to within 2% by counting the bits in n and then linearly interpolating between the two nearest powers of two. In fact the cepstral disparity algorithm works quite well even if the interpolation step is omitted, so the EUCLID implementation simply counts bits. The ADSP-2100 has a NORMALIZE instruction, so counting the bits requires only two or three instructions per point.

As a final optimization, note that the final set of column transforms only needs to be done in the region that will be searched for peaks. This region consists of the columns representing horizontal frequencies between $w/2$ and w. This makes it possible to eliminate 3/4 of the column transforms.

All of the optimizations described above were incorporated into the EUCLID implementation of the cepstral disparity estimator. Sixteen-bit fixed-point arithmetic was used throughout. The DIT and DIF FFTs were implemented as assembly language subroutines, as was the procedure that computes the power spectrum and takes its log. All other code was written in C. Using the optimizations described above, EUCLID computes the cepstral disparity estimate for 32×32 windows in 58.6 milliseconds, not including the 3-4 ms required to acquire the VME bus and copy the sample arrays from the frame buffer into local memory.

A final optimization (which we have not done) would be to take advantage of the fact that the FFTs in the cepstral filter operate on real data. This would involve using a real-data transform on the image rows, and then (since the output is known to be symmetrical about the origin) transforming only half of the columns. We expect that this would cut the run time to about 40 milliseconds, including image load time. Note that this is comparable to the RS-170 frame time, so that synchronizing with the cameras might become a significant part of the loop delay.

Figure 4 b) shows shows the result of running the implementation described above on the input shown in 4 a). The input image is synthetic in that it consists of two overlapping windows taken from the same real image. Since the two sample windows are related by a pure shift, this represents something of a "best case" for the algorithm. For display purposes all of the columns in the output have been computed, and the result has been shifted to put frequency $(0,0)$ in the center. The pixel values have also been logarithmically scaled in order to reduce the relative brightness of the central peak. The two bright dots to the upper left and lower right of the central peak are the disparity peaks. The apparent noisiness of the cepstral output is an artifact of the reproduction process; in fact the disparity peaks stand well above the noise background, and can be extracted with high reliability.

4.2 Using the Cepstral Disparity for Vergence Control

The vergence control loop for the Rochester Robot can be split into three stages: digitization, error estimation, and error correction. Digitization is done under control of the Sun host using the MaxVideo digitizer. At present the system has only one digitizer, so the Sun must select one camera, digitize a frame, switch cameras and digitize again. This stage requires 4 frame times (132 milliseconds[2].) Once the frames are available in the frame store, the Sun signals EUCLID to begin computing a disparity estimate. The EUCLID computation takes about 63 milliseconds, during which time the Sun is free to do other things. EUCLID places its disparity estimate in a well known location in shared memory and issues an interrupt to signal completion. The Sun converts the pixel disparity to motor coordinates and commands the non-dominant eye motor to make the appropriate correction. It then waits for the motor to stop, and the loop repeats. The motor correction takes about 80 milliseconds, varying slightly with the size of the correction. Thus the total loop time is approximately 275 milliseconds, reducible to about 210 by adding a second digitizer.

Figure 5 shows samples from two video sequences produced by the MaxVideo system while the robot viewed a changing scene. For both sequences the MaxVideo system was programmed to display the average of the left and right images, so that disparities would be readily visible. The left-hand series was produced with vergence disabled, the right-hand series with vergence enabled. Note the extreme disparities that occur when the cameras do not verge. Note also that the cepstral algorithm works quite well even when the target is very close, at which time the assumption that the two images are related by a pure shift is grossly violated.

5 Understanding the cepstral filter

Our work with the cepstral filter has helped us to develop some intuition about why it works so well. The standard view [Bogert *et al.*, 1963; Yeshurun and Schwartz, 1987] is that

[2]A second digitizer will be added to the system soon. This will permit simultaneous digitization of the left and right images, lowering the duration of this part of the loop to 1 to 2 frame times (33-66 milliseconds.)

the transform extracts a periodic term in the log power spectrum of the sum of the original and shifted images. This is mathematically correct, but for purposes of understanding how the algorithm works we find it more useful to stress its relation to autocorrelation. The argument is as follows:

Ignore for the moment the fact that the cepstral input consists of the left and right images spliced together. What does the rest of the algorithm do? It is well known that the power spectrum is the Fourier transform of the autocorrelation function. If one skips taking the log of each pixel, and replaces the second Fourier transform with the inverse transform, then the procedure computes exactly the autocorrelation. But the Fourier transform of any function is just the complex conjugate of the *inverse* Fourier transform of that function. Since the autocorrelation function is even-symmetric, its Fourier transform is strictly real– it is its own complex conjugate. Therefor, the second Fourier transform might as well be an inverse Fourier transform. Without the log step, then, the filter just computes the autocorrelation function.

What is the effect of taking the logarithm before the inverse Fourier transform? For any *particular* input image, it is equivalent to multiplying each point in the transformed autocorrelation function by a scalar. This implies that the cepstral filter is equivalent to a linear filter applied to the autocorrelation function – except that the filter isn't really linear, because the filter mask depends on the image. At present we have only an intuitive understanding of what this filter does. The log function makes small numbers a little smaller, but it makes large ones a lot smaller. Thus we argue that for two signals of equal energy, it will favor the one whose energy is the most evenly distributed, *i.e.* the one with the broadest spectrum. The signals with the broadest spectra, of course, are impulses; so the logarithm tends to favor spikes in the autocorrelation function. The cepstral filter can be thought of as modifying the image to make its autocorrelation function more impulse-like, hence easier to interpret.

If this intuitive argument has any validity, then any non-linear compressive function applied to the power spectrum should have a similar sharpening effect. Experiments suggest that this is in fact the case. Figures 6 a) and b) show an input image and its autocorrelation, the latter being the inverse Fourier transform of the power spectrum. Note the bright regions (marked by arrows) corresponding to the disparity match in the upper left and lower right parts of the image. Figure 6 c) shows the effect of taking the log of the power spectrum before inverting, *i.e.* the standard cepstrum. Figure 6 d) shows the effect of replacing the log with the fourth root, while 6 e) shows the effect of taking the arc tangent. Note that the peaks in the latter two cases are comparable in quality to those in the cepstrum.

The ultimate compressive operator would be one that takes all input values to a constant. The Fourier transform of a constant is an impulse at $(0, 0)$, so this operator would provide the unhelpful information that the image matches itself perfectly at a disparity of zero. In order to get useful disparity information by this method we would have to find a way to preserve the phase information which is normally destroyed when we take the power spectrum. For example, we might compute the transformed cross-correlation of the left and right images by multiplying the transform of one times the conjugate of the transform of the other, and then rescale so that all entries in the resulting complex array have the same magnitude. It turns out that this intuitively derived algorithm can be rigorously justified as a type of deconvolution, as we shall see in the next section.

5.1 Estimating Disparity by Deconvolution

The cepstral filter was originally developed to solve a particular problem in the processing of one-dimensional signals – that of analysing signals containing echos [Bogert *et al.*, 1963]. Such signals can be modelled as an original signal $S(t)$ convolved with a train of impulses, *i.e.*

$$R(t) = S(t) * (\delta(t) + a_0\delta(t - t_0) + a_1\delta(t - t_1) + \ldots)$$

Taking the log of the power spectrum transforms the received signal into a sum of two terms, one of which depends only on $S(t)$ and the other of which is a combination of distorted sinusoids with frequencies related to t_0, t_1 et cetera. If the cepstrum of $S(t)$ does not overlap the frequencies of the echo terms, conventional linear filtering techniques can be used to extract the values of the echo delays.

The Yeshurun and Schwartz method of disparity estimation begins by appending the left and right images. This creates a two-dimensional analogue of the situation described by the above equation. That is, on the assumption that the two images are related by a shift, the input can be interpreted as the result of convolving an ideal image with two impulses, one at $(0,0)$ and the other at $(w + d_h, d_v)$. One can then compute the cepstrum and look for peaks produced by the sinusoidal terms, trusting that the log power spectrum of the ideal image will have negligible energy at the frequencies corresponding to plausible disparities.

Given the assumption that the right and left images differ only by a shift, however, a more direct approach is possible. The stated assumption is equivalent to the formula

$$R(x,y) = L(x,y) * \delta(x - d_h, y - d_v)$$

In the frequency domain, this translates to

$$F_R(u,v) = F_L(u,v)e^{-j2\pi(ud_h + vd_v)}$$

from which we can derive

$$e^{-j2\pi(ud_h + vd_v)} = \frac{F_R(x,y)}{F_L(x,y)}$$

or

$$\delta(x - d_h, y - d_v) = \mathcal{F}^{-1}\left(\frac{F_R(u,v)F_L^*(u,v)}{|F_L(u,v)|^2}\right)$$

By hypothesis, however, F_L and F_R have identical magnitude spectra – they differ only in phase, because the left and right images differ only by a shift. Thus the division can be rewritten

$$\frac{F_R(u,v)F_L^*(u,v)}{|F_R(u,v)F_L^*(u,v)|} \tag{1}$$

which is the Fourier transform of the crosscorrelation of the right and left images, rescaled so that all entries have magnitude one. That is, it is exactly what we argued for on intuitive grounds in the previous section. What was described there as a peak-sharpening operation turns out to undo the convolution of $L(x,y)$ with the disparity delta function.

Deconvolution by dividing Fourier tranforms is of course not new, and it has a number of well-known difficulties [Rosenfeld and Kak, 1982]. Problems arise even if the divisor in expression (1) is non-zero. Observe that the effect of the division is to obliterate magnitude information and preserve only the phase at each frequency. But if the magnitude of the complex number at a given frequency is small, then a small amount of additive noise may radically alter the phase. Thus trouble can be expected whenever the magnitude of the transformed correlation is small. In the EUCLID implementation of deconvolution we simply return 0 whenever the magnitude is less than an empirically determined threshold. In practice, this does not seem to degrade the quality of the solutions significantly. Figure 7 shows the input images and the recovered impulse for a synthetic stereo pair, in which the assumption that the two images differ by a shift is perfectly satisfied. The location of the impulse correctly indicates that the horizontal and vertical disparities are (resp.) -5 and -2, and the impulse stands two orders of magnitude above the background noise.

The deconvolution disparity estimator performs quite well on the synthetic stereo pair of figure 7. Given deconvolution's reputation for instability, however, one might expect it to give much less satisfactory results on real images. So far, however, this does not appear to be the case. Figures 8 a), b) and c) show the inputs and the result of deconvolution for a real stereo pair taken by the robot head. Note that in this case the deconvolution produces three distinct peaks corresponding to the disparities of the book, the statuette and the bottle. Figure 8 c) shows the result of shifting the right image by the strongest disparity in the deconvolution image and adding it to the left image. We do not have "ground truth" about the correct disparity for this stereo pair, but based on figure 8 c) it appears that the estimate is quite accurate. We suspect that deconvolution's bad reputation is due in part to the fact that its paradigmatic applications are very hard problems, such as undoing camera misfocussing and motion blur. It is not surprising that deconvolution turns out to be unstable in these applications, for the applications themselves are inherently unstable.

5.2 Efficient computation of deconvolution

The steps required to compute the deconvolution disparity estimate are very similar to those needed to compute the cepstrum. They are:

- extract sample windows of size $h \times w$ from the left and right images.
- Compute the 2–D Fourier transforms of each window.
- Compute the pointwise product of one image with the complex conjugate of the other and rescale so that each point has magnitude one.
- Take the 2–D Fourier transform (forward or inverse) of the result.
- Find the peak value in the resulting image.

The result image will have a single peak at (d_h, d_v), provided that d_h is on the interval $(-w/2, w/2)$. Disparities between $w/2$ and w may still be detected, but the returned value will be incorrect by $\pm w$ – a potentially disastrous situation. This ambiguity can be avoided by the same trick that was used in the cepstral implementation. One of the images is transformed in such a way as to add a vertical disparity of $h/4$. If the peak value occurs at a negative vertical disparity, then the horizontal disparity is off by w. If the measured horizontal disparity is negative, then w must be added to the estimate, otherwise w must be subtracted.

Since the deconvolution and cepstrum algorithms are structurally similar, many of the techniques used to implement the cepstrum efficiently can also be used for deconvolution. Like the cepstrum, the deconvolution algorithm consists of two 2-D FFTs separated by a point operation. It can therefor be implemented efficiently by using DIT and DIF 1-D FFTs as described in section 4.1.

Taking the pointwise product of the two transformed images is easy, but normalizing the resulting complex numbers to magnitude one is not. The straightforward approach would be to divide the components of each complex number by the number's magnitude. This however would require a potentially expensive square root operation. This expense can be avoided by using the power spectrum of one of the input images as an estimate of the magnitude of the product image. This approximation is valid because we assume (as in section 5.1) that the two input images have identical magnitude spectra. Avoiding the square root does not solve all of the problems, however. The EUCLID microprocessor does not have a single-cycle divider – division by n bits is an n-cycle operation. Fortunately, the following very crude approximation produces excellent results: First, estimate the magnitude of the complex number by the method just described. Next, let b be the number of bits in the estimated magnitude minus the number of bits in the fixed-point representation of one. If b is negative, the complex number already has a small magnitude, so it should be skipped. If not, right-shift each component of the complex number by b. The effect of these manipulations is to preserve the phase of the complex number while bringing its magnitude to within a factor of two of the fixed-point representation of one. The fact that the algorithm continues to perform well despite the crudeness of this approximation is a testimony to its robustness.

Using the optimizations and approximations described above, EUCLID computes the deconvolution disparity estimate for 32×32 windows in 54.7 milliseconds, not including image load time. This is about 7% faster than the EUCLID cepstral filter implementation.

6 Conclusion

We have argued that vergence is important for active vision systems, and have demonstrated two algorithms for real-time vergence on the Rochester Robot. We have also presented an intuitive argument relating the cepstral method of disparity estimation to autocorrelation, and claimed that deconvolution is a logical extension of the cepstral approach. Deconvolution offers two advantages over the cepstrum. First, it is slightly faster, since it replaces one large FFT with two smaller ones. Second, its output is a pure disparity signal, containing no trace of the original input image. Traces of the original image can be removed from the cepstrum if necessary, but doing so requires a computationally expensive post-processing stage.

Our immediate plans for the vergence system include work in two major areas. One deals with the problem of selecting the right size sampling windows for the cepstral or deconvolution algorithms. Recall that a window that is too small may not have enough detail to support reliable disparity estimation. A window that is too large may have multiple disparities present, and will have lower precision for a given number of samples per window. We would like a way for the algorithm itself to decide how large the sampling window should be. In order to do this, we need to determine how our disparity estimators behave in the situations described above. If we can detect situations where the window size is too large or too small, we can hope to adapt the window size continuously to suit changing conditions. Figure 8 b) offers some hope that this will indeed be possible.

The other major task awaiting us is to integrate vergence with saccadic and tracking movements of the dominant eye. The current set of tracking algorithms uses a velocity tracking loop [Brown, 1988], so the simple positional error-nulling vergence control loop that we have used so far may not be appropriate. We may want to use the tracking velocity control signal to drive both eyes, with superimposed saccadic movements of the non-dominant eye to correct vergence errors. Alternatively, we may want to convert the positional vergence error to a velocity and combine it with the velocity error signal from the dominant eye.

A longer term project that we are considering would be to convert the entire gaze control mechanism to a foveate camera system. We have a long-standing interest in biological vision systems, and feel that working with foveate cameras might yield interesting insights about human vision. One way to implement a fovea would be to construct a "chopped" resolution pyramid using the MaxVideo image processor. For example, we might convert the 512x512 input images into sets of five 64x64 images centered on the optic axis. The first would be an appropriately filtered and subsampled version of the entire image, the second a more detailed view of the central 256×256 region, and so on. The last image would show a small region in the center of the image at full resolution. Aside from the interesting research issues, there is a brutally practical reason for doing this. The MaxVideo processor runs at ten megapixels per second, independent of the size of the input image. Compressing a 2^{18}-pixel input into 5×2^{12} pixels would greatly increase the number of computations we can do in a frame time.

Acknowledgements

The authors gratefully acknowledge the support of the granting agencies listed on the front of this report. Dana Ballard and Chris Brown provided leadership and advice, and read several drafts of this paper. Dave Tilley and Tim Becker helped make the DataCube and PUMA software usable. Hal Moroff of the DataCube corporation helped us untangle the mysteries of the EUCLID digital signal processor. Josh Diamond wrote an early version of the cepstral disparity estimator.

References

[Aloimonos *et al.*, 1987] J. Aloimonos, I. Weiss, and A. Bandopadhay, "Active Vision," In *Proc. 1st Int'l. Conf. on Computer Vision*, pages 35–54, London, June 1987.

[Analog Devices, 1987] Analog Devices, Inc., Norwood, Massachussetts, *DSP Products Databook*, 1987.

[Ballard, 1987] Dana H. Ballard, "Eye Movements and Spatial Cognition," Technical Report 218, University of Rochester Computer Science Department, November 1987.

[Bandopadhay, 1986] Amit Bandopadhay, *A Computational Study of Rigid Motion Perception*, PhD thesis, University of Rochester, 1986, also published as Computer Science Department TR 221.

[Barnard and Fischler, 1982] S. T. Barnard and M. A. Fischler, "Computational Stereo," *Computing Surveys*, 14:553–572, December 1982.

[Bogert *et al.*, 1963] B. P. Bogert, M. J. R. Healy, and J. W. Tukey, "The Quefrency Alanysis of Time Series for Echoes: Cepstrum, Pseudo-autocovariance, Cross-Cepstrum, and Saphe Cracking," In M. Rosenblatt, editor, *Proc. Symp. Time Series Analysis*, pages 209–243. John Wiley and Sons, 1963.

[Brooks, 1986] Rodney A. Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, (RA-2):14–23, April 1986.

[Brooks, 1987] Rodney A. Brooks, "Intelligence Without Representation," In *Proc. Workshop on Foundations of Artificial Intelligence*, pages 1–21, 1987.

[Brown, 1988] Christopher M. Brown, "The Rochester Robot," Technical Report 257, University of Rochester Computer Science Department, August 1988.

[Datacube, 1987] Datacube, Inc., Peabody, Massachussetts, *MaxVideo Installation and Software Manual*, 1987.

[Horn, 1986] Berthold K. P. Horn, *Robot Vision*, The MIT Press, Cambridge, Massachusetts, 1986.

[Rosenfeld and Kak, 1982] Azriel Rosenfeld and Avinash C. Kak, *Digital Picture Processing*, volume 1, Academic Press, Orlando, 1982.

[Yeshurun and Schwartz, 1987] Yehezkel Yeshurun and Eric L. Schwartz, "Cepstral Filtering on a Columnar Image Architecture: A Fast Algorithm for Binocular Stereo Segmentation," Robotics Research Technical Report 286, New York University Courant Institute of Mathematical Sciences, March 1987.

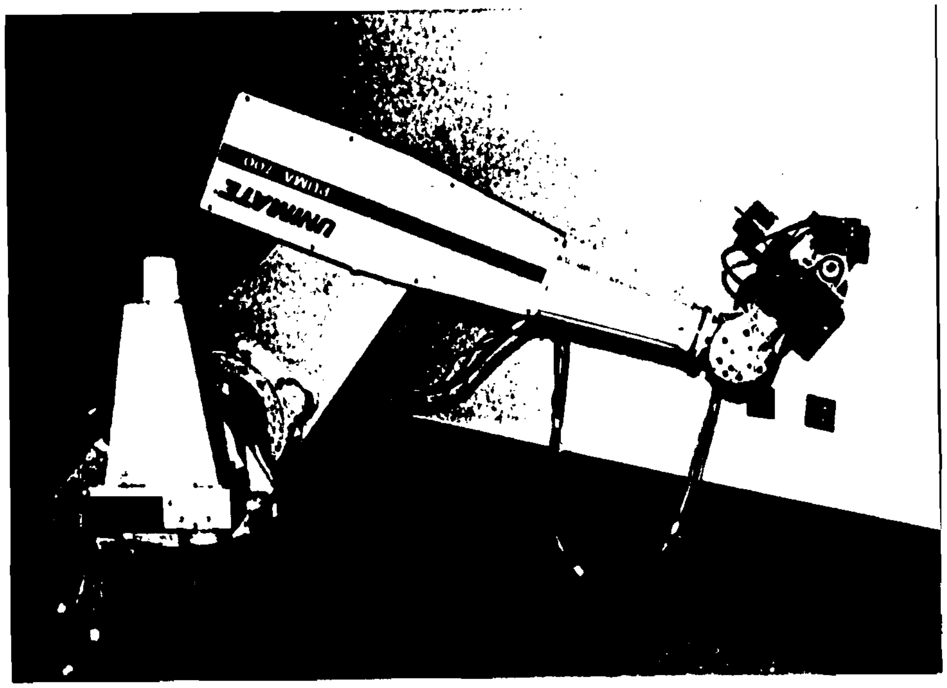


Figure 1: The Rochester Robot

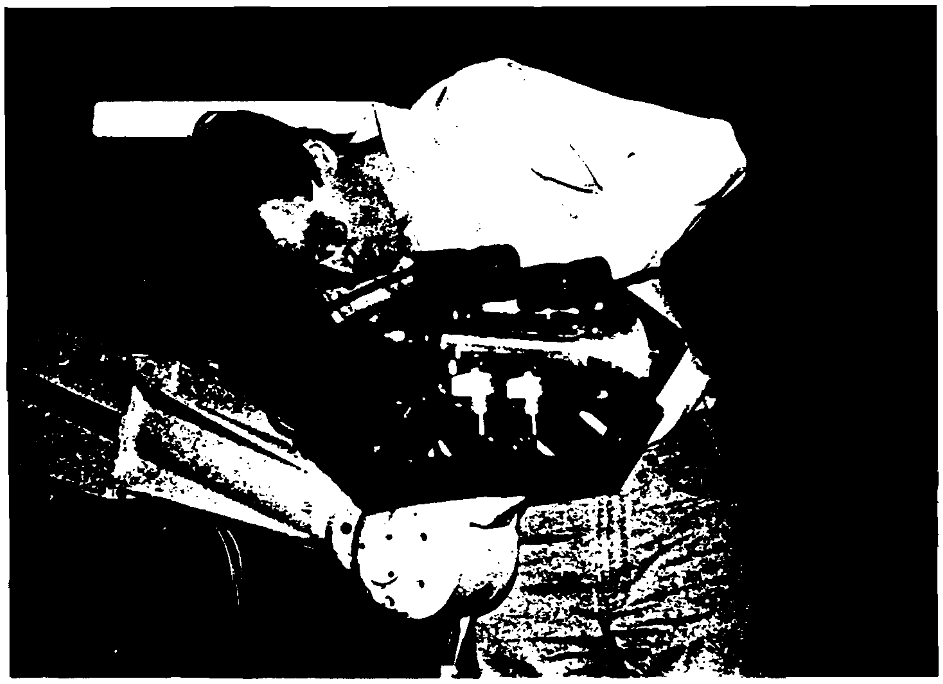


Figure 2: The robot head consists of two CCD television cameras mounted on a movable beam. The beam can rotate about its long axis, and the cameras can rotate about their points of attachment to the beam.

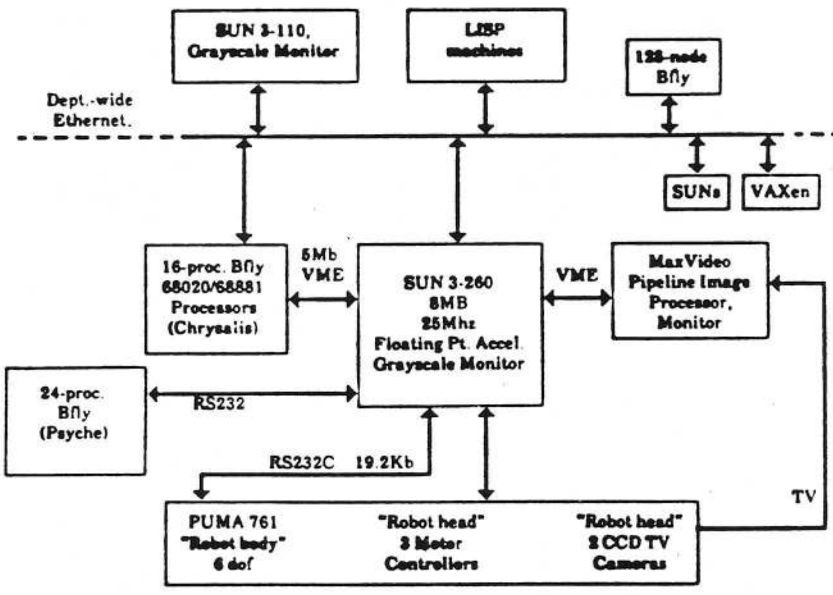

SUN 3-110, Grayscale Monitor
LISP machines
128-node Bfly
Dept.-wide Ethernet.
SUNs
VAXen
16-proc. Bfly 68020/68881 Processors (Chrysalis)
5Mb VME
SUN 3-260 8MB 25Mhz Floating Pt. Accel. Grayscale Monitor
VME
MaxVideo Pipeline Image Processor, Monitor
24-proc. Bfly (Psyche)
RS232
RS232C 19.2Kb
TV
PUMA 761 "Robot body" 6 dof
"Robot head" 3 Motor Controllers
"Robot head" 2 CCD TV Cameras

Figure 3: The University of Rochester Robotics and Vision Laboratory – computational resources and communication links.

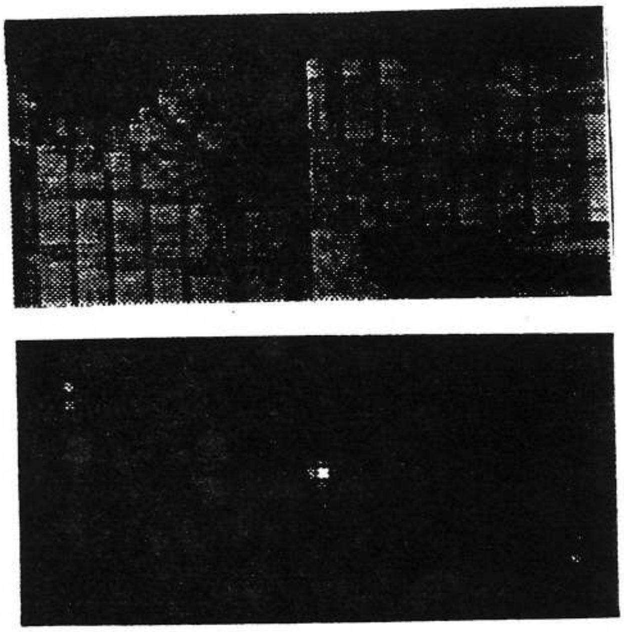


Figure 4: Inputs and outputs for the EUCLID implementation of the cepstral disparity estimator. The input consists of two 32×32 sample windows spliced together, with the right window scrolled up by one quarter the window height. The positions of the peaks to the upper left and lower right of the output array give the estimated disparity.

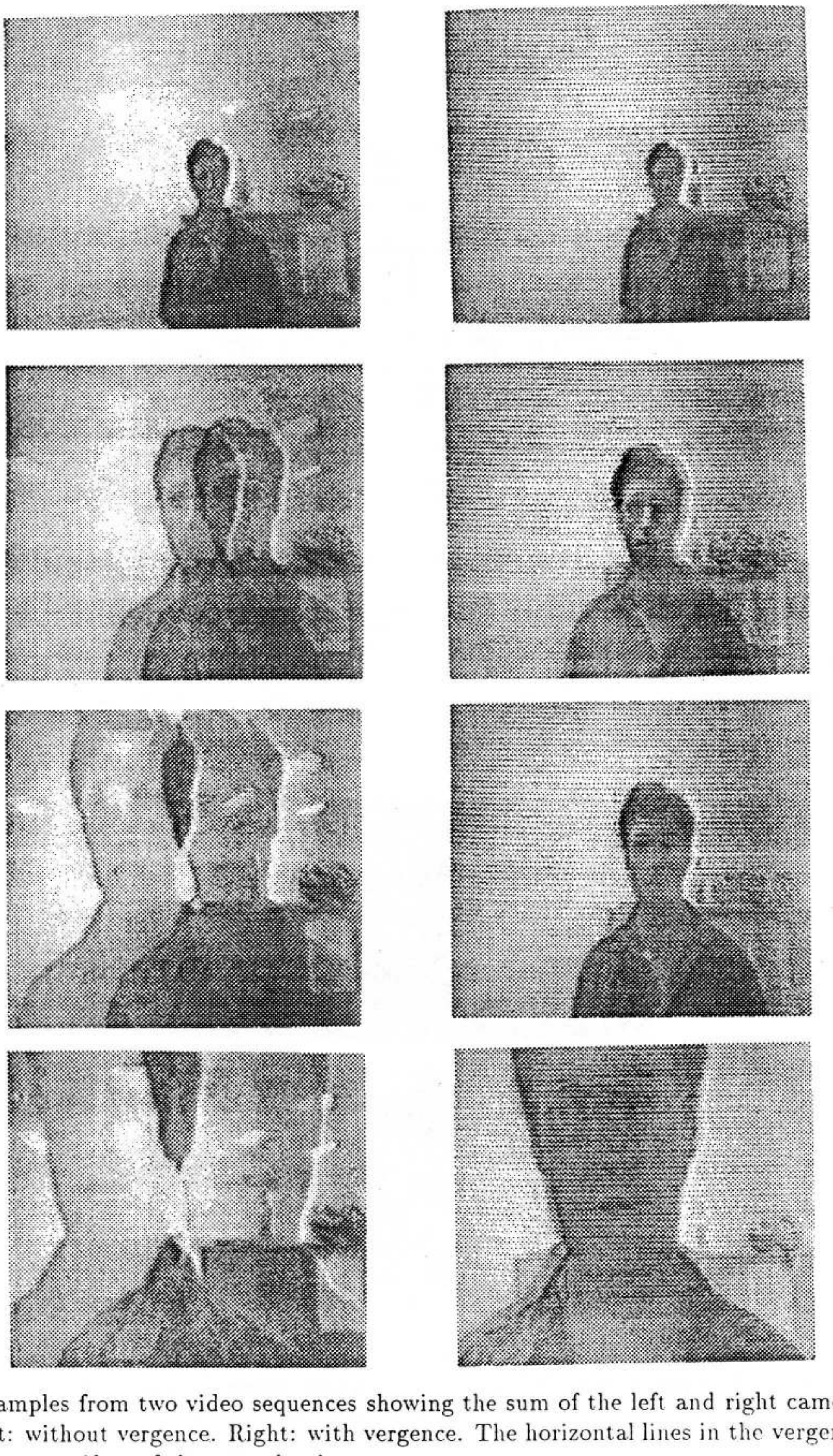


Figure 5: Samples from two video sequences showing the sum of the left and right camera images. Left: without vergence. Right: with vergence. The horizontal lines in the vergence sequence are an artifact of the reproduction process.

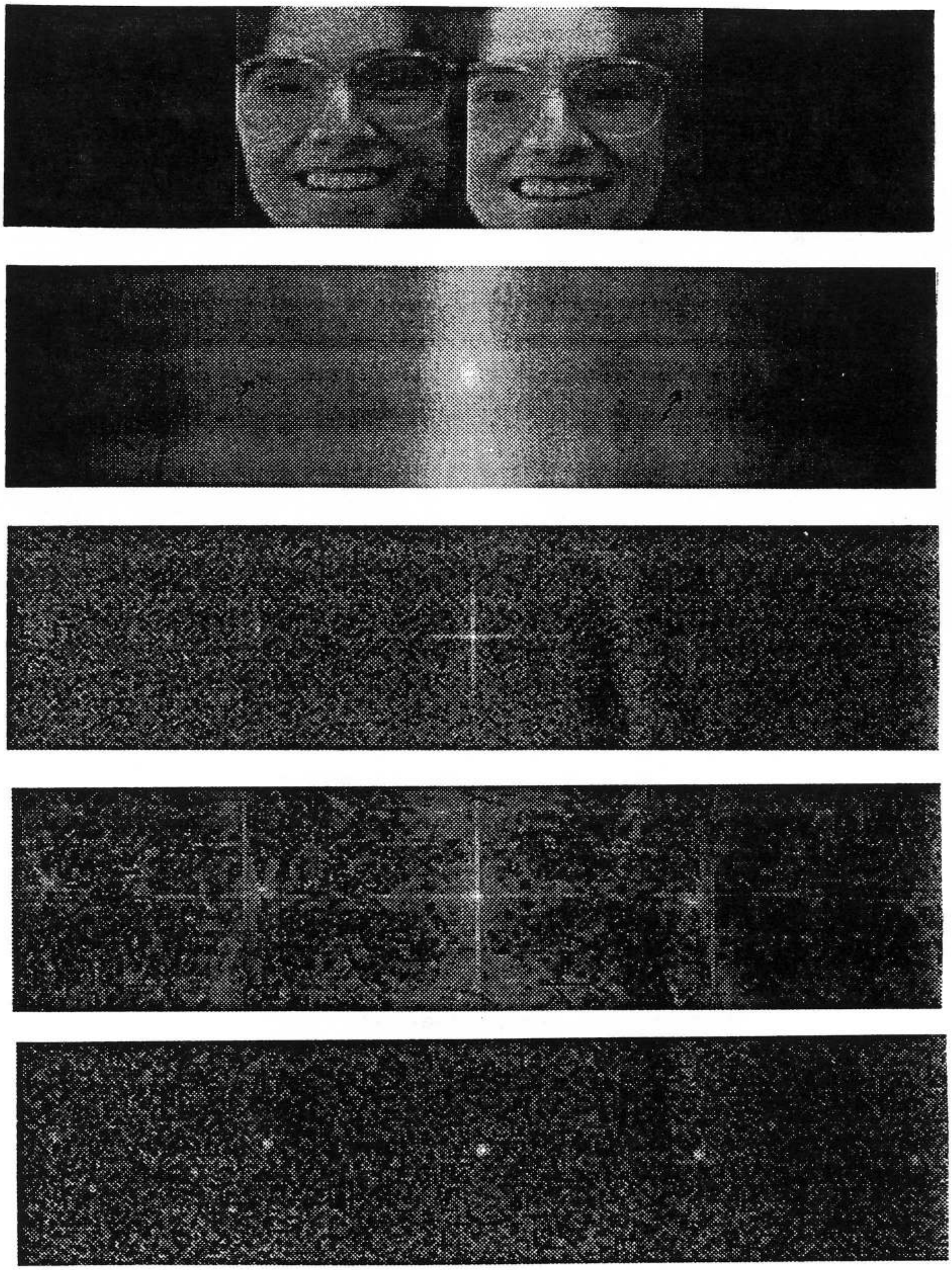
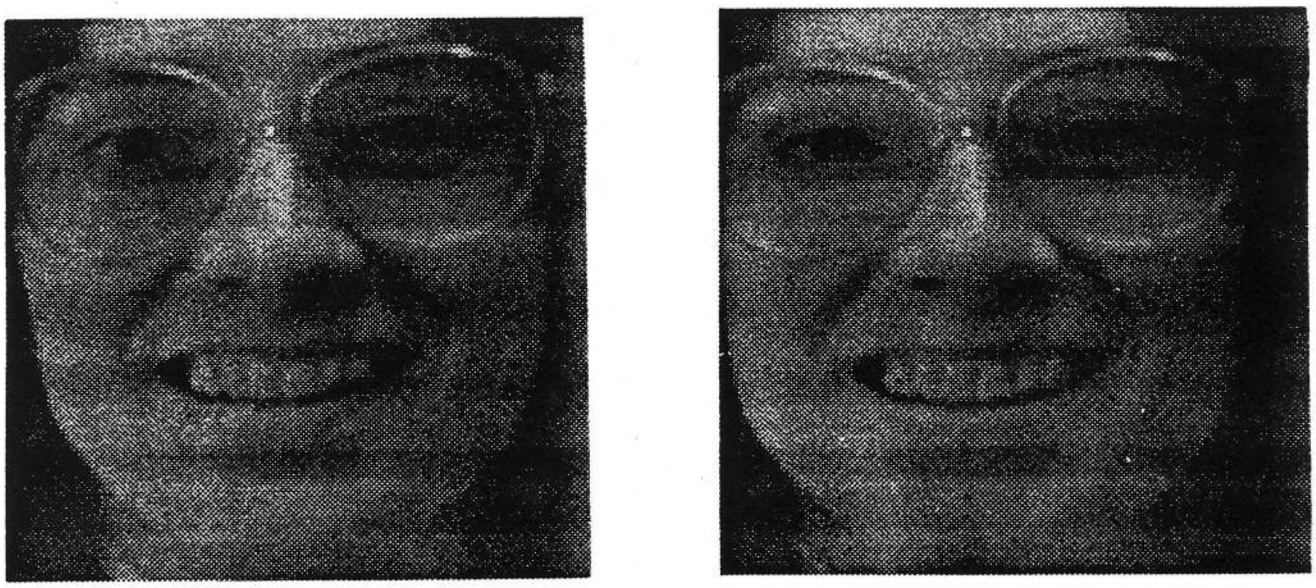


Figure 6: The cepstrum and related filters. a) A padded input image. b) The autocorrelation of the input image. Arrows show where the disparity peaks occur. c) Effect of taking the log of the power spectrum before inverting, *i.e.*the cepstrum. d) effect of taking the fourth root instead of the log. e) effect of taking the arc tangent.



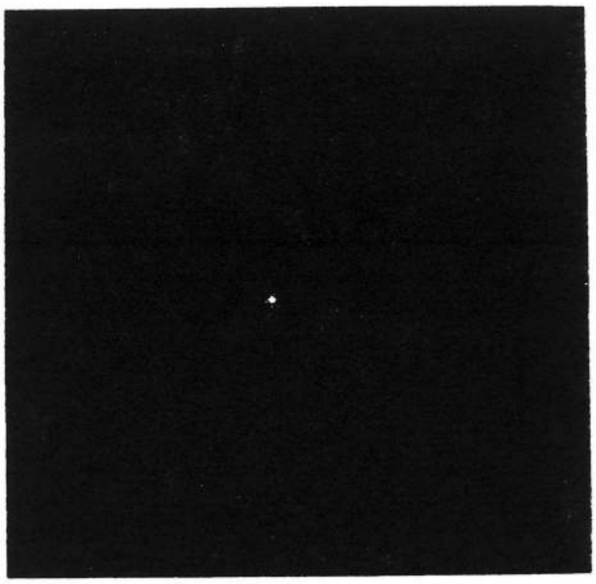


Figure 7: Estimating disparity by deconvolution. a) Two 128 $\times$ 128 overlapping windows extracted from a real image. The horizontal and vertical disparities are -5 and -2. b) The deconvolution output((0,0) at center.) The peak occurs at (-5,-2) as expected.

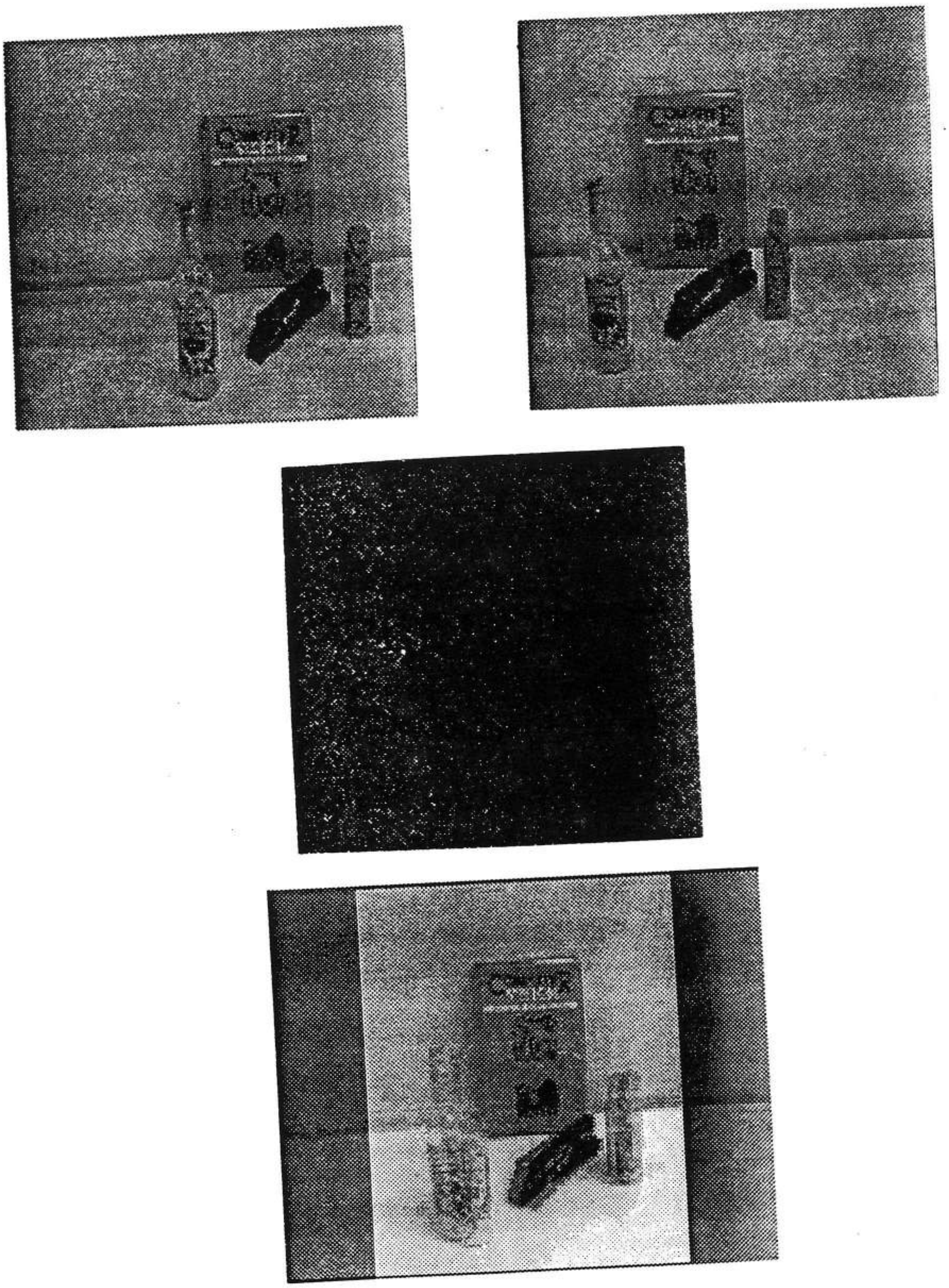


Figure 8: Deconvolution with a real stereo pair taken by the robot head. a) The input images. b) The deconvolution output. Note that we now have three peaks corresponding to the three main objects in the scene. The strongest peak is at (-28,1). c) The result of shifting the left image by the indicated amount and adding it to the right image.