System-Level Design of Specialized VLSI Hardware for Computing Relative Orientation

Lisa Dron

M.I.T. Artificial Intelligence Laboratory
545 Technology Square, Cambridge, MA 02139

Abstract

Determining the relative orientation between the coordinate systems of two cameras is central to binocular stereo, as well as to long range motion vision. In this paper I propose a system-level design for specialized hardware to compute relative orientation in real-time. The problem is difficult for two reasons: one is the nonlinearity of the equations which must be solved, and which may possess multiple solutions or local minima; the other is the difficulty of finding point correspondences. The major contribution of this paper is the theoretical analysis and simulation of simple methods, which map well to analog and digital hardware, for resolving both difficulties.

1 Introduction

Computing the relative orientation between the coordinate systems of two cameras is important for calculating depth from binocular stereo and for determining general camera motion. The problem is difficult for two reasons: one is the nonlinearity of the equations which must be solved; and the other is the difficulty of determining point correspondences between two images.

The problem is of central importance to applications involving machine vision and has been extensively studied over the last 10–15 years. A variety of techniques have been developed both for finding point correspondences as well as for solving the motion equations [3, 6, 7, 8]. However, most of these are designed in software to be executed on (relatively) large and bulky digital computers. My goal is to build a small, autonomous system with specialized analog and digital VLSI hardware for computing relative orientation in real-time. Such a system would be suitable for mounting on mobile robots or on any remote platform that cannot be tethered to a computer.

The constraints of hardware design are very different from those of software. In order to achieve real-time operation with minimal hardware, the algorithms to be executed must be made as simple as possible without compromising reliability. Because a significant part of the effort in designing the complete system is spent in choosing and adapting the algorithms, the focus of this paper is on the methods which are used. Before discussing these methods in detail, however, I first present the overall stucture and operation of the complete system.

2 System Architecture

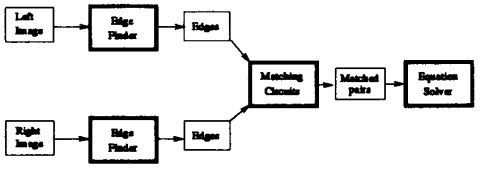

Figure 1: System block diagram

The block diagram of the complete system is shown in Figure 1. The two input images acquired at the different camera positions are referred to as *left* and *right*, regardless of whether this terminology reflects their true physical disposition. Whether the two images are acquired by separate imaging systems and processed simultaneously, as shown in the diagram, or by the same imager and processed sequentially, is unimportant.

Point correspondences are found by matching edges. The edge detector in the first stage is physically implemented by a two-dimensional CCD array with analog processing which implements the multiscale veto (MSV) algorithm. Since this algorithm is extensively described in [1], it is only briefly discussed here. The MSV algorithm differs from most edge detection methods in that it does not low-pass filter the image and then look for maxima in the first derivative. Instead, it identifies edges as occurring between two pixels wherever there is a sharp change in the image brightness function which persists over a range of spatial scales. There are two principal advantages to using the MSV algorithm in this system. The first is that it produces edges which are well-localized to feature boundaries, regardless of how the input parameters are adjusted to reject noise or small features. As a result, displacements measured in the edge maps accurately reflect displacements of the corresponding

0-8186-2840-5/92 $03.00 © 1992 IEEE

features in the original images. The second advantage of the MSV algorithm is that it can be compactly implemented with analog circuits. The circuitry can be placed on the image plane itself between the image sensors, or built into a separate unit used in conjunction with an off-line imager.

In the second stage, the binary edge maps produced by the edge detection circuits are fed into an array of matching circuits, each of which computes, for a specific patch in one edge map, the translational offset which brings it best into alignment with features in the second edge map. The search is restricted to a predefined area, or window, whose dimensions may be controlled by varying the input and shift sequences. Although it may be possible to significantly reduce this area if some external information is available, in general we expect the search to be over a large window. The reason we can obtain reliable matches despite the large search area is due to the low false-alarm probability of the scoring method. It is the speed and parallelism of the hardware which make this brute force method feasible in real-time. Because the edge signal is binary, it can be shifted with a single one-bit clocked shift register per pixel. Scoring is computed at each clock cycle by analog circuits embedded within the 2-d array of shift registers. By using analog circuits we can build a larger array on the same silicon area than would be possible with an equivalent digital implementation. Although analog processing necessitates a relatively slow, 5–10MHz, clock rate, this should not hinder real-time operation. Even at 5MHz, 10,000 shifts and compares, or the equivalent of searching a 100×100 pixel area, can be performed in 2ms. At video frame rates of 30 frames/second, this leaves over 31ms for additional processing before the next frame arrives.

In the third and final stage of the system, the point correspondences determined by the matching circuits are used to compute the relative orientation of the two camera coordinate systems. The algorithm used is is based on an adaptation of Horn's formulation [6], and represents a significant reduction in total computation—of approximately a factor 3—over the previous method. Because most of its computations involve matrix multiplications and additions, the algorithm is implemented on a digital microprocessor which is best suited to these types of operations.

3 The Matching Procedure

3.1 The scoring method

We begin with two binary edge maps, one designated as the base, and the other as the secondary. We divide the base into N, $M \times M$-sized, patches. Let i, $1 \leq i \leq N$ denote the ith patch and define

$P \equiv$ number of pixels in each patch $= M^2$
$B_i \equiv$ set of pixels in base patch i corresponding to an edge signal
$\overline{B}_i \equiv$ complement of B_i in patch i

Patch i is *admissible* if:

$$.1P < \|B_i\| < .4P \tag{1}$$

The admissibility of each patch is determined by the number of edge pixels. Nothing changes as far as the matching circuit is concerned, but the results from inadmissible patches are ignored.

Once each patch from the base is loaded into the matching circuit, it is 'swept across' a predefined area in the secondary by loading the corresponding edge pixels from the secondary and shifting them row-wise and column-wise over the base patch. Let (j,k) denote the coordinates, in the secondary image, of the center of the $M \times M$ patch currently being matched against the ith base patch. Define

$S_{jk} \equiv$ set of edge pixels in the $M \times M$ patch centered at (j,k) in the secondary
$R_{ijk} \equiv$ score of base patch i at (j,k)

The score is computed as

$$R_{ijk} = \frac{1}{\|B_i\|}\left(\|S_{jk} \cap B_i\| - \|S_{jk} \cap \overline{B}_i\|\right) \tag{2}$$

The output of the matching circuit are the coordinates (j,k) where the best score is obtained, and a binary flag which indicates if the match is reliable or not.

3.1.2. Why it works

We first observe that

$$-\frac{\|\overline{B}_i\|}{\|B_i\|} \leq R_{ijk} \leq 1 \tag{3}$$

and,

$$R_{ijk} = 1 \Longleftrightarrow \text{A perfect match} \tag{4}$$

It is unlikely that two patches from different edge maps, even corresponding to the same feature and with no perspective distortion, will be identical, since there will always be slight differences in the edge finder. However, their relative score should be close to one. On the other hand, patches corresponding to different features are likely to have very low scores. Let,

$$V_{ijk} \equiv \|S_{jk} \cap B_i\| - \|S_{jk} \cap \overline{B}_i\| \tag{5}$$

Suppose the matching is against a random collection of points, each of which may take on the values 0 or 1 with equal probability. Over a large number of such matches the expected value of V_{ijk} is:

$$\begin{aligned} E[V_{ijk}] &= \frac{1}{2}\left(\|B_i\| - \|\overline{B}_i\|\right) \\ &= \frac{1}{2}\left(2\|B_i\| - P\right) \end{aligned} \tag{6}$$

If we further assume that the pixel values of the base patch are i.i.d. Bernoulli random variables, then it can be shown that the variance of V_{ijk} is

$$Var[V_{ijk}] = P\left(\frac{1}{4} + \frac{\|B_i\|}{P} - \left(\frac{\|B_i\|}{P}\right)^2\right) \tag{7}$$

The assumption of independence is made so that we can add variances, and because we do not know the joint distribution between pixels. The values of neighboring pixels are of course correlated, but over the entire patch, independence is a reasonable assumption.

From (6) and (7) we obtain

$$E[R_{ijk}] = \frac{1}{\|B_i\|} E[V_{ijk}] = 1 - \frac{P}{2\|B_i\|} \tag{8}$$

and

$$\begin{aligned} \sigma(R_{ijk}) &= \frac{1}{\|B_i\|}\sqrt{Var(V_{ijk})} \\ &= \frac{M}{\|B_i\|}\left(\frac{1}{4} + \frac{\|B_i\|}{P} - \left(\frac{\|B_i\|}{P}\right)^2\right)^{1/2} \end{aligned} \tag{9}$$

since $M = \sqrt{P}$.

The reason for the admissibility requirement is now obvious. With

$$.1P < \|B_i\| < .4P$$

we have

$$-4 < E[R_{ijk}] < -.25$$

and

$$\frac{1.9}{M} < \sigma(R_{ijk}) < \frac{5.9}{M}.$$

Note that there is no significance to the exact values .1 and .4 for the coefficients. Other values could work as well, but these are convenient to write. The point is that for any admissible patch, the probability of obtaining a high score, say .5 or greater, from a random match is very low since it is several σ away from the mean. The decision to accept or reject matches is therefore based on thresholding the scores. For some $\tau < 1$ such that $Pr(R_{ijk} > \tau)$ is small, we assign the indicator variables

$$w_i = \begin{cases} 1 & \text{if max } R_{ijk} > \tau \\ 0 & \text{if max } R_{ijk} \leq \tau \end{cases} \tag{10}$$

By definition, $w_i = 0$ for any inadmissible patch.

3.2 Circuit design

The most attractive feature of the matching procedure is that the scores are easy to compute with analog circuits. The proposed design is to build one matching circuit per image patch, each consisting of a block of random access memory cells, to store the base edges and a two-dimensional array of one-bit clocked shift registers to shift the secondary edges across the base patch. It should be possible to put several matching circuits on the same chip to reduce the number of off-chip connections and simplify the shifting circuitry.

The matching circuits will receive the binary edge signals output by the multi-scale veto chip. The number of base edge signals is computed by generating current sources at each pixel where an edge is present and then summing the currents. Scoring will be done by generating two current sources, one proportional to the logical AND of the secondary edge signals with the base edges, and the other to the AND of the secondary edge signals with the complements of the base edges. The difference between these two currents is then converted to a voltage which is compared to the best previous score, by means of a differential sense amplifier. An off-chip counter is maintained to keep track of the number of shifts. When the maximum score changes, the counter value is latched. The final value stored is later converted to image plane coordinates from knowledge of the shift sequence.

4 Computing the Relative Orientation

Assuming that the internal geometry of the imaging systems is known, the image plane coordinates of each pair of corresponding points found by the matching circuits may be converted into ray directions emanating from the origins of their respective coordinate systems. Let $\mathbf{r}_i$ and $\boldsymbol{l}_i$ denote the unit direction vectors, $\|\mathbf{r}_i\|^2 = \|\boldsymbol{l}_i\|^2 = 1$, in the right and left camera coordinates corresponding to the ith pair of matched points. The rays from the centers of projection, which are the origins of the two coordinate systems, in the directions $(\mathbf{r}_i, \boldsymbol{l}_i)$ intersect at the same world point if and only if they are coplanar with the baseline, $\mathbf{b}$, joining the origins. This *coplanarity condition*, can be expressed as [5]

$$\lambda_i = \boldsymbol{l}'_i \cdot (\mathbf{r}_i \times \mathbf{b}) = \mathbf{b} \cdot (\boldsymbol{l}'_i \times \mathbf{r}_i) = 0 \tag{11}$$

where $\boldsymbol{l}'_i$ denotes the ith left ray rotated into the right coordinate system and $\mathbf{b}$ is a unit vector, expressed in the right camera's coordinates, in the direction of the baseline that connects the origins of the two systems.

The relative orientation problem is to compute the rotation (3 degrees of freedom) and the direction of the baseline (2 degrees of freedom) that describe the transformation between the two systems so that (11) holds for each of the N pairs of ray directions found by the matching procedure. Of course, we cannot expect (11) to hold exactly for all N pairs since there are inevitably errors in the matching process. Instead of solving the equations directly, we seek to minimize some measure of the total error. Horn [5] showed that the triple product λ_i is itself a good measure of the error in the pair $(\mathbf{r}_i, \boldsymbol{l}_i)$ and that a weighted least-squares minimization problem can be formulated as minimizing

$$S = \sum_{i=1}^{N} w_i \lambda_i^2 \tag{12}$$

over all possible rotations and directions of the baseline. The weights, w_i, in (12) are the confidence measures associated with each pair of ray directions and may in general be continuously-valued over the range $[0, 1]$. In the present case, these weights are the binary-valued indicator variables defined by (10).

No true closed form solution exists to the least-squares problem (12) when both the rotation and the direction of the baseline are unknown. Horn [6] developed an iterative algorithm to solve (12) which works well for points lying on an arbitrary non-degenerate surface and which converges to some, perhaps local, minimum of (12) regardless of the initial values used. If the starting values are not too far away from the solution, convergence is rapid, usually requiring fewer than 10 iterations. However, each iteration, which involves solving an 11×11 system of linear equations, is relatively expensive computationally. The algorithm developed for the current system is an adaptation of Horn's basic formulation in which the iterations are split into two parts with the rotation and baseline updated separately, instead of simultaneously. In this manner, it is possible both to derive closed form equations to compute the incremental improvements to the rotation and the baseline and to guarantee convergence to some stationary point. Simulations indicate that the simplified algorithm converges more slowly than Horn's method, requiring about 2.5 times as many iterations; however, each iteration involves only 1/8 as many computations. The net gain is therefore a speedup in execution by approximately a factor of 3.

4.1 Updating the baseline given the rotation

If the rotation is known, or assumed, the minimization of (12) is a straightforward eigenvalue-eigenvector problem [5] which can be expressed as

$$S = \mathbf{b}^{\mathrm{T}}\left(\sum_{i=1}^{N} w_i \mathbf{c}_i \mathbf{c}_i^{\mathrm{T}}\right)\mathbf{b} \tag{13}$$

$$= \mathbf{b}^{\mathrm{T}}\mathbf{C}\mathbf{b} \tag{14}$$

where $\mathbf{c}_i = \boldsymbol{l}'_i \times \mathbf{r}_i$. The baseline, $\mathbf{b}$, which minimizes (12) is the eigenvector corresponding to the smallest eigenvalue of $\mathbf{C}$. Since $\mathbf{C}$ is a 3×3 symmetric positive definite matrix, its eigenvalues and eigenvectors can be computed in closed form.

4.2 Updating the rotation given the baseline

If the baseline is known, or assumed, it is not possible to obtain an eigenvector equation to solve for the rotation, but it is possible to find a simple expression to compute the incremental improvement to the rotation. As suggested by Horn [6], we represent rotation with unit quaternions. Following his notation, let $\mathring{l}_i$ and $\mathring{\boldsymbol{l}}'_i$ denote unit quaternions with zero scalar part and vector part equal to $\boldsymbol{l}_i$ and $\boldsymbol{l}'_i$. If $\mathring{\mathrm{q}}$ is the unit quaternion which describes the rotation, then $\mathring{l}_i$ and $\mathring{\boldsymbol{l}}'_i$ are related by

$$\mathring{\boldsymbol{l}}'_i = \mathring{\mathrm{q}}\mathring{l}_i\mathring{\mathrm{q}}^{*} \tag{15}$$

For readers unfamiliar with quaternions and their algebraic properties, several fundamental definitions and identities are given in the Appendix.

Using the above expression and equation (44) from the Appendix, the triple product, λ_i can be written as

$$\begin{aligned}\lambda_i &= \mathring{\mathrm{r}}_i\mathring{\mathrm{b}}\cdot\mathring{\mathrm{q}}\mathring{l}_i\mathring{\mathrm{q}}^{*} \\ &= \mathring{\mathrm{r}}_i\mathring{\mathrm{b}}\mathring{\mathrm{q}}\cdot\mathring{l}_i\mathring{\mathrm{q}}\end{aligned} \tag{16}$$

Using quaternion matrices, this expression can be rearranged into a more convenient form as

$$\lambda_i = \mathring{\mathrm{q}}^{\mathrm{T}}\mathbf{F}_i\mathring{\mathrm{q}} \tag{17}$$

where $\mathbf{F}_i$ is a symmetric matrix defined by

$$\mathbf{F}_i = \frac{1}{2}(\mathcal{B}\mathcal{R}_i\underline{\mathcal{L}}_i - \underline{\mathcal{L}}_i\mathcal{R}_i\mathcal{B}) \tag{18}$$

The matrices $\mathbf{F}_i$ depend only on the value of $\mathbf{b}$, which is given, and the data $\{(\mathbf{r}_i, \boldsymbol{l}_i)\}$. The weighted sum of squares is written using (17) as

$$S = \sum_{i=1}^{N} w_i\lambda_i^2 = \sum_{i=1}^{N} w_i\left(\mathring{\mathrm{q}}^{\mathrm{T}}\mathbf{F}_i\mathring{\mathrm{q}}\right)^2 \tag{19}$$

It is not possible to remove $\mathring{\mathrm{q}}$ entirely from the summation and write (19) as a quadratic form involving $\mathring{\mathrm{q}}$ and a constant matrix. We can make S decrease, however, by taking a step in the direction of the negative gradient computed subject to the restriction $\|\mathring{\mathrm{q}}\| = 1$. Algebraically we make the gradient independent of the magnitude of $\mathring{\mathrm{q}}$ by normalizing by $\mathring{\mathrm{q}}^{\mathrm{T}}\mathring{\mathrm{q}}$. Let

$$\underline{\lambda}_i \equiv \frac{\lambda_i}{\mathring{\mathrm{q}}^{\mathrm{T}}\mathring{\mathrm{q}}} \tag{20}$$

Then

$$\begin{aligned}\frac{\partial\underline{\lambda}_i}{\partial\mathring{\mathrm{q}}} &= \frac{\partial}{\partial\mathring{\mathrm{q}}}\left(\frac{\mathring{\mathrm{q}}^{\mathrm{T}}\mathbf{F}_i\mathring{\mathrm{q}}}{\mathring{\mathrm{q}}^{\mathrm{T}}\mathring{\mathrm{q}}}\right) \\ &= \frac{2}{\mathring{\mathrm{q}}^{\mathrm{T}}\mathring{\mathrm{q}}}\left(\mathbf{F}_i\mathring{\mathrm{q}} - \left(\frac{\mathring{\mathrm{q}}^{\mathrm{T}}\mathbf{F}_i\mathring{\mathrm{q}}}{\mathring{\mathrm{q}}^{\mathrm{T}}\mathring{\mathrm{q}}}\right)\mathring{\mathrm{q}}\right) \\ &= \frac{2}{\mathring{\mathrm{q}}^{\mathrm{T}}\mathring{\mathrm{q}}}\left(\mathbf{F}_i\mathring{\mathrm{q}} - \underline{\lambda}_i\mathring{\mathrm{q}}\right)\end{aligned} \tag{21}$$

Now define

$$\underline{S} \equiv \frac{S}{4(\mathring{\mathrm{q}}^{\mathrm{T}}\mathring{\mathrm{q}})^2} = \frac{1}{4}\sum_{i=1}^{N} w_i\underline{\lambda}_i^2 \tag{22}$$

Then

$$\begin{aligned}\nabla_q\underline{S} \equiv \frac{\partial\underline{S}}{\partial\mathring{\mathrm{q}}} &= \frac{1}{2}\sum_{i=1}^{N} w_i\underline{\lambda}_i\left(\frac{\partial\underline{\lambda}_i}{\partial\mathring{\mathrm{q}}}\right) \\ &= \frac{1}{\mathring{\mathrm{q}}^{\mathrm{T}}\mathring{\mathrm{q}}}\sum_{i=1}^{N} w_i\underline{\lambda}_i\left(\mathbf{F}_i - \underline{\lambda}_i\mathbf{I}\right)\mathring{\mathrm{q}}\end{aligned} \tag{23}$$

Note that

$$\mathring{\mathbf{q}}^{\mathrm{T}} \nabla_q \underline{S} = 0 \tag{24}$$

and that evaluating the above expression at $\|\mathring{\mathbf{q}}\| = 1$ gives

$$\nabla_q \underline{S} = \sum_{i=1}^{N} w_i \lambda_i \left(\mathbf{F}_i - \lambda_i \mathbf{I}\right) \mathring{\mathbf{q}} \tag{25}$$

A necessary condition for a minimum of S is $\nabla_q \underline{S} = \mathbf{0}$, which occurs iff

$$\sum_{i=1}^{N} w_i \lambda_i \mathbf{F}_i \mathring{\mathbf{q}} = \mathring{\mathbf{q}} \sum_{i=1}^{N} w_i \lambda_i^2 \tag{26}$$

Equation (26) is not a standard eigenvector relation because λ_i depends on $\mathring{\mathbf{q}}$.

We wish to compute a new rotation quaternion, $\mathring{\mathbf{q}}'$, from $\mathring{\mathbf{q}}$ by taking the largest step possible in the direction of the negative gradient such that S is certain to decrease. Let $\delta\mathring{\mathbf{q}} = -\widehat{\nabla_q \underline{S}}$ be a unit vector in the direction of $-\nabla_q \underline{S}$. The problem is to find the coefficients (s, t) such that

$$\mathring{\mathbf{q}}' = s\mathring{\mathbf{q}} + t\delta\mathring{\mathbf{q}} \tag{27}$$

which give the largest possible decrease in the total squared error. For $\mathring{\mathbf{q}}'$ to be a unit quaternion, s and t must also satisfy

$$s^2 + t^2 = 1 \tag{28}$$

since $\mathring{\mathbf{q}} \cdot \delta\mathring{\mathbf{q}} = 0$. Substituting (27) into (19) gives

$$\sum_{i=1}^{N} w_i \lambda_i'^2 = \sum_{i=1}^{N} w_i \left[(s\mathring{\mathbf{q}} + t\delta\mathring{\mathbf{q}})^{\mathrm{T}} \mathbf{F}_i (s\mathring{\mathbf{q}} + t\delta\mathring{\mathbf{q}})\right]^2 \tag{29}$$

We drop the term in t^2, which should be small if we are near the minimum value, and take the derivative of the remaining terms with respect to t. Setting the result to zero and defining $\nu_i = \delta\mathring{\mathbf{q}}^{\mathrm{T}} \mathbf{F}_i \mathring{\mathbf{q}}$, we obtain after some manipulation

$$t = -s \frac{\sum_{i=1}^{N} w_i \lambda_i \nu_i}{2 \sum_{i=1}^{N} w_i \nu_i^2} \tag{30}$$

which, combined with (28), allows us to compute s and t. This expression can be further simplified by noting that

$$\begin{aligned} \sum_{i=1}^{N} w_i \lambda_i \nu_i &= \delta\mathring{\mathbf{q}}^{\mathrm{T}} \sum_{i=1}^{N} w_i \lambda_i \mathbf{F}_i \mathring{\mathbf{q}} \\ &= \delta\mathring{\mathbf{q}}^{\mathrm{T}} \nabla_q \underline{S} \\ &= -\|\nabla_q \underline{S}\| \end{aligned} \tag{31}$$

and

$$\begin{aligned} \sum_{i=1}^{N} w_i \nu_i^2 &= \delta\mathring{\mathbf{q}}^{\mathrm{T}} \left(\sum_{i=1}^{N} w_i \mathbf{F}_i \mathring{\mathbf{q}} \mathring{\mathbf{q}}^{\mathrm{T}} \mathbf{F}_i\right) \delta\mathring{\mathbf{q}} \\ &= \delta\mathring{\mathbf{q}}^{\mathrm{T}} \mathbf{W} \delta\mathring{\mathbf{q}} \end{aligned} \tag{32}$$

where $\mathbf{W}$ is used to replace the summation term in parentheses which does not depend on $\delta\mathring{\mathbf{q}}$. (Note that $\delta\mathring{\mathbf{q}}^{\mathrm{T}} \mathbf{W} \delta\mathring{\mathbf{q}} > 0$.) Finally,

$$\frac{t}{s} = \frac{\|\nabla_q \underline{S}\|}{2\delta\mathring{\mathbf{q}}^{\mathrm{T}} \mathbf{W} \delta\mathring{\mathbf{q}}} \tag{33}$$

Only one pass over the stored data (ray pairs) is needed to compute $\nabla_q \underline{S}$ and $\mathbf{W}$. These terms are sufficient to then calculate t/s, and hence $\mathring{\mathbf{q}}'$, without further reference to the data. If (33) is substituted back into (29), still neglecting the term in t^2, we find after simplifying that

$$\sum_{i=1}^{N} w_i \lambda_i'^2 \approx s^4 \left[\left(\sum_{i=1}^{N} w_i \lambda_i^2\right) - \frac{\|\nabla_q \underline{S}\|^2}{\delta\mathring{\mathbf{q}}^{\mathrm{T}} \mathbf{W} \delta\mathring{\mathbf{q}}}\right] \tag{34}$$

Hence, as long as $\|\nabla_q \underline{S}\| \neq 0$, the total error will always decrease.

4.3 The algorithm

The simplified algorithm is summarized as follows:

Input: $\mathring{\mathbf{q}}^{(0)}$, $\mathbf{b}^{(0)}$, data
$k = 0$
change $= 1$
$S^{(0)} = \sum_{i=1}^{N} w_i \lambda_i^{(0)2}$
while (*change* $> \epsilon$) {
  $\mathring{\mathbf{q}}^{(k+1)} =$ UPDATE_Q($\mathring{\mathbf{q}}^{(k)}$, $\mathbf{b}^{(k)}$, data)
  ($\mathbf{b}^{(k+1)}$, $S^{(k+1)}$) = UPDATE_B($\mathring{\mathbf{q}}^{(k+1)}$, data)
  /* $S^{(k+1)}$ = min eigenvalue of $\mathbf{C}$ */
  change $= (S^{(k)} - S^{(k+1)})/S^{(k)}$
  $k{+}{+}$
}

It is easy to see that the weighted sum of squares, S, monotonically decreases with each iteration, since it is reduced at each step. Since S is bounded below by zero, the algorithm must converge to some stationary point.

4.4 Multiple solutions

If the data are error-free, a finite number of solutions exist to the non least-squares problem of solving $\lambda_i = 0$ for all N if there are at least five ray pairs that do not lie on a degenerate surface. Faugeras and Maybank [2] showed that in general there are 10 solutions to the five-point problem. If at least eight pairs are available, there is a unique solution since, as shown by Longuet-Higgens [7], the equations can be written in a linear form.

There is no guarantee, however, that the least-squares problem has a unique minimum, even with error-free data, for any number of ray pairs. If the field of view is small, or if the baseline is oriented along the optical axis, the problem may become unstable or develop local minima. If either of $|\mathbf{b} \times \mathbf{r}_i|$ or $|\mathbf{l}'_i \times \mathbf{r}_i|$ is small, which occurs when the rays are nearly parallel, the triple product will be small regardless of the orientation of the third vector. A common non-global

minimum of S occurs when the true baseline is parallel to the image plane and the points viewed are several baseline units in front of the cameras. In this case, $|\hat{z} \times \mathbf{r}_i|$ is small for almost all pairs ($\hat{z}$ being the direction of the optical axis), and a local minimum of S may exist for $\mathbf{b}$ strongly oriented in the $\hat{z}$ direction. The problems of instability and local minima are intrinsic to the equations and not to the algorithm. The only way to avoid them is by incorporating some higher level techniques. In the next section I discuss a case where a local minimum occurs in simulation and offer suggestions for simple ways to avoid getting stuck in it.

5 Simulations

5.1 The matching procedure

Figure 2 illustrates the results of the matching procedure on an image sequence. The viewing geometry is defined such that the $\hat{z}$-axis points in the direction of the optical axis, while the (horizontal) $\hat{x}$- and (vertical) $\hat{y}$-axes are parallel to the image plane and are aligned with the rectangular pixel grid. The base image is a picture of a poster (of Neil Armstrong) taken by a digital CCD camera, with the second image produced by computer-simulated motion. The reason for generating simulated motion is to be able to compare the results produced by the relative orientation algorithm with a known motion and known camera calibration. The motion simulation program assumes an image of a planar surface at a user-supplied depth and orientation. The focal length, principal point, and x, y pixel spacing are input to the program to compute ray directions using the pinhole camera model. For this sequence, the focal length and pixel spacing were such that the effective field of view was 110°, full-field, on both axes. The base image was modeled as a frontal plane (parallel to the image plane) at a depth of 10 (baseline) units, and the motion was a translation of 1 unit in the positive $\hat{x}$-direction with a 5° rotation about the $\hat{y}$-axis.

The matching program executes the scoring procedure in exactly the manner previously described on the binary edge maps produced by simulation of the edge detection circuit. The image size is 400×400 pixels, and the base edge map was divided into 256 patches (16 rows of 16), each of size 24×24. Searching was done over a 100×100 pixel window in the second edge map centered about each patch location.

The numbers on the edge maps correspond to the matches which passed the threshold test (10). In this case, the threshold was set at $\tau = .6$ and 52 good matches were obtained. The quality of these matches can be appreciated visually. The correspondence is excellent, and is almost as good as would have been achieved by a human operator.

5.2 Relative orientation results

The simplified relative orientation algorithm was applied to the data from the matching procedure, and the results starting from two different initial values are listed in Table 1. Convergence of the algorithm was declared once the fractional change in the sum of squared errors between iterations was less than 10^{-5}. Starting with initial values for $\mathbf{b}$ and $\hat{\mathbf{q}}$ near the correct values, the algorithm converges to a solution which is very close to the exact motion. The value of .00011 for the sum of squared errors is a direct indication of how well the matching procedure worked.

	Initial Values	
$\mathbf{b}$	(.958, 0, .287)	(0, 0, 1.0)
$\hat{\mathbf{q}}$	(1.0, 0, 0, 0)	(1.0, 0, 0, 0)
	Results	
S	.00011	.00022
$\mathbf{b}$	(.9997, -.02, -.02)	(.1049, -.07, .992)
$\hat{\mathbf{q}}$	(.9991, 0, .0419, 0)	(.9956, 0, .0936, 0)
$(\theta, \hat{\omega})$	(4.8°, -.01, .9998, .02)	(10.7°, 0, .9999, .02)

Table 1: Simulation results.

The results in the second column were obtained by starting with a different initial value for $\mathbf{b}$. The algorithm converges to a stationary point corresponding to an incorrect solution for both $\mathbf{b}$ and $\hat{\mathbf{q}}$. The value of $\mathbf{b}$ which is found, (.1049, -.071, .992), is almost perpendicular to the correct one. This is an example of the case just discussed where a local minimum results due to the fact that the ray directions from the images are nearly parallel to the optical axis, and hence $\|\mathbf{b} \times \mathbf{r}_i\|$ is small for all ray pairs. The angle subtended in the horizontal direction, which is the true direction of $\mathbf{b}$, by the matched pairs is much less than the 110° field of view of the entire image. It is therefore possible to find a rotation such that $\|\mathbf{l}'_i \times \mathbf{r}_i\|$ is also small.

There are several ways in which this situation can be recognized or avoided. The sum of squared errors, .00022, which is small, is not a good indicator that we are stuck in a local minimum. However the condition number of $\mathbf{C}$, where $\mathbf{C}$ is the symmetric matrix given in (14), is a good check on whether or not the solution is the correct one. Since the least eigenvalue of $\mathbf{C}$ must be computed in solving for $\mathbf{b}$, not much extra work is required to compute the other two eigenvalues. If the ray pairs do not contain much error, as is the case here, the condition number of $\mathbf{C}$, K_C, should be large, since the largest eigenvalue is related to the average value of $\|\mathbf{l}'_i \times \mathbf{r}_i\|$. For the correct solution, column 1, $K_C = 2450$, while for the incorrect solution, $K_C = 62$. We can compute K_C during the execution of the algorithm, and if it is small, suspect the reliability of the solution and restart with a different value of $\mathbf{b}$. Of course, other possibilities for avoiding local minima are to begin with a good initial guess and to use a wide-angle lens so that the part of the scene which is visible to both cameras subtends as wide an angle as possible.

6 Summary

In this paper, I have outlined a design for a system to compute the relative orientation between two cameras in specialized VLSI hardware. An important part

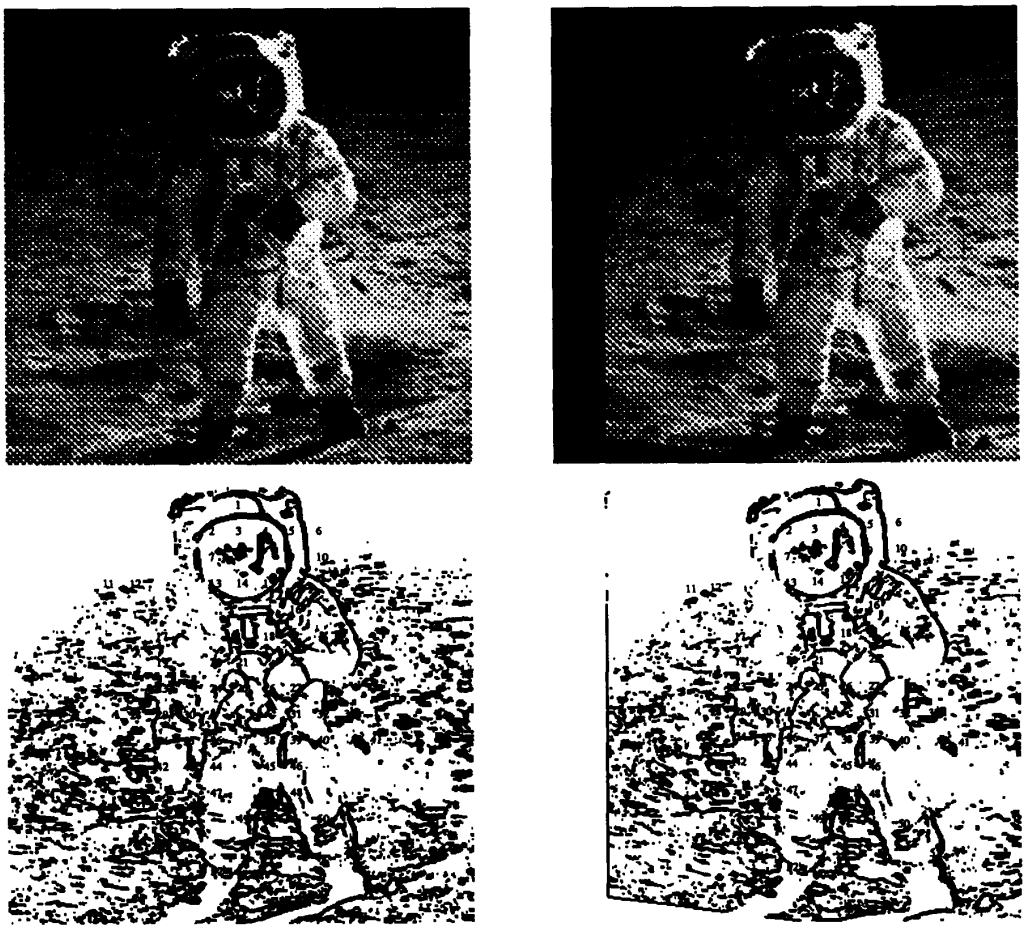

Figure 2: Top left: original image – frontal plane. Top right: simulated motion of $\hat{x}$ translation and 5° rotation about $\hat{y}$. Bottom: binary edge maps showing matched points.

of the design is developing methods which can be implemented in hardware for determining point correspondences and solving the nonlinear equations. I have described a procedure for finding point matches, which is simple to compute with analog circuits, and have derived an algorithm for solving the least-squares minimization problem which requires less computation than previous methods. Results from simulations show that the matching procedure works well and that the motion can be accurately recovered. A situation where a local minimum arises in simulation was also discussed, as were suggestions for simple ways of recognizing and avoiding it.

Acknowledgements

I wish to thank Prof. Berthold Horn for providing helpful comments on a previous draft of this paper.

I also wish to acknowledge my fellowship sponsor, AT&T Bell Laboratories, who has generously supported me for the last three and a half years. This paper describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124. Funding for the Analog VLSI for Machine Vision research project is provided by NSF Grant MIP-91-17724 and NSF and DARPA contracts MIP-88-14612.

Appendix. Quaternion Algebra

Quaternions may be thought of as 4-vectors endowed with special operations of multiplication and conju-

gation. It is often convenient to write them as the composition of a scalar and 'vector' part [6].

$$\mathring{a} = (a_0, \mathbf{a}) \tag{35}$$

where $\mathbf{a}$ is a vector in $\mathbb{R}^3$. Conjugation is defined by

$$\mathring{a}^* = (a_0, -\mathbf{a}) \tag{36}$$

and the multiplication of two quaternions is defined by

$$\mathring{a}\mathring{b} = (a_0 b_0 - \mathbf{a}\cdot\mathbf{b}, a_0\mathbf{b} + b_0\mathbf{a} + \mathbf{a}\times\mathbf{b}) \tag{37}$$

In all other respects, quaternions can be treated as ordinary vectors with the transpose, dot product, and multiplication by a matrix defined in the usual manner.

Another way to express the same operation in (37) is by a matrix-vector multiplication using equivalent quaternion matrices. For example,

$$\mathring{a}\mathring{b} = \begin{pmatrix} a_0 & -a_x & -a_y & -a_z \\ a_x & a_0 & -a_z & a_y \\ a_y & a_z & a_0 & -a_x \\ a_z & -a_y & a_z & a_0 \end{pmatrix} \mathring{b} \equiv \mathcal{A}\mathring{b} \tag{38}$$

and

$$\mathring{a}\mathring{b} = \begin{pmatrix} b_0 & -b_x & -b_y & -b_z \\ b_x & b_0 & b_z & -b_y \\ b_y & -b_z & b_0 & b_x \\ b_z & b_y & -b_x & b_0 \end{pmatrix} \mathring{a} \equiv \underline{\mathcal{B}}\mathring{a} \tag{39}$$

$\mathcal{A}$ is referred to as the *left* quaternion matrix associated with $\mathring{a}$, and $\underline{\mathcal{B}}$ is referred to as the *right* quaternion matrix associated with $\mathring{b}$. Quaternion matrices can be useful for rearranging formulas into more convenient expressions.

A vector in $\mathbb{R}^3$ can be represented by a quaternion with zero scalar part. For example, the 3-vectors $\mathbf{r}$, $\boldsymbol{l}$, and $\mathbf{b}$ can be represented as

$$\mathring{r} = (0, \mathbf{r}), \quad \mathring{l} = (0, \boldsymbol{l}), \quad \text{and} \quad \mathring{b} = (0, \mathbf{b}) \tag{40}$$

Quaternions with zero scalar part have some additional convenient properties. If $\mathring{r}$, $\mathring{l}$, and $\mathring{b}$ are as above, then

$$\mathring{r}^* = -\mathring{r} \tag{41}$$

$$\mathring{r}\cdot\mathring{l} = \mathbf{r}\cdot\boldsymbol{l} \tag{42}$$

$$\mathring{r}\mathring{l} = (-\mathbf{r}\cdot\boldsymbol{l}, \mathbf{r}\times\boldsymbol{l}) \tag{43}$$

$$\mathring{b}\cdot(\mathring{r}\mathring{l}) = \mathbf{b}\cdot(\mathbf{r}\times\boldsymbol{l}) \tag{44}$$

The last identity above is the quaternion representation of the triple product.

The reason that quaternions are so useful in the formulation of the relative orientation problem is because of the simplicity with which rotation about an arbitrary axis can be represented. A unit quaternion is one whose magnitude, defined as the square root of its dot product with itself, is unity.

$$\mathring{q}\cdot\mathring{q} = 1 \tag{45}$$

Every unit quaternion represents a rotation in $\mathbb{R}^3$ of an angle θ about an axis $\hat{\omega}$ in the sense that

$$\mathring{q} = \left(\cos\frac{\theta}{2}, \hat{\omega}\sin\frac{\theta}{2}\right) \tag{46}$$

The rotational transformation of a vector $\boldsymbol{l}$ is obtained from

$$\mathring{l}' = \mathring{q}\mathring{l}\mathring{q}^* \tag{47}$$

Additional results on quaternions and their properties can be found in [4, 6].

References

[1] Lisa Dron. *The Multi-Scale Veto Model: A Two-stage Analog Network for Edge Detection and Image Reconstruction.* A.I. Memo 1320, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, March 1992. Submitted to International Journal of Computer Vision.

[2] Olivier D. Faugeras and Steve Maybank. Motion from point matches: multiplicity of solutions. *International Journal of Computer Vision*, 4:225–246, 1990.

[3] W. Eric L. Grimson. Computational experiments with a feature based stereo algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(1):17–34, 1985.

[4] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4:629–642, April 1987.

[5] Berthold K. P. Horn. Relative orientation. *International Journal of Computer Vision*, 4(1):59–78, 1990.

[6] Berthold K. P. Horn. Relative orientation revisited. *Journal of the Optical Society of America*, 8:1630–1638, October 1991.

[7] H. C. Longuet-Higgens. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[8] Juyang Weng, Narendra Ahuja, and Thomas S. Huang. Matching two perspective views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(8):806–825, August 1992.