From Image Sequences to Recognized Moving Polyhedral Objects

DAVID W. MURRAY, DAVID A. CASTELOW AND BERNARD F. BUXTON
Department of Engineering Science, University of Oxford, Parks Road, Oxford, OX1 3PJ, UK

Abstract
This paper describes the combination of several novel algorithms into a system that obtains visual motion from a sequence of images and uses it to recover a three-dimensional description of the motion and geometry of the scene in terms of moving extended straight edges. The system goes on to recognize the recovered geometry as an object from a database of wireframe models, a stage that also resolves the depth/speed scaling ambiguity inherent in visual motion processing, resulting in absolute depth and motion recovery. The processing sequence is demonstrated on imagery from a well-carpentered CSG model and on natural imagery of simple polyhedral objects.

1 Introduction
The primary aim of research into computational vision, and indeed of that into many other automated sensing techniques, is to give machines the power to perceive the three-dimensional nature of the environment in which they are required to take intelligent action. More often than not, action involves movement, and so the recovery of three-dimensional motion at a low level of the sensory processing is of great importance in robotics.

The two-dimensional visual motion derived from a sequence of time-varying images is one valuable source of information about the 3D scene and its motion relative to the sensor. At the most basic level, visual motion can be used simply to flag scene motion, but it has long been appreciated, certainly since the work of von Helmholtz [1], that encoded within it is much more detail about the 3D geometric structure and 3D motion of the scene. The capability to exploit this in the human visual system has been demonstrated in a variety of psychophysical experiments over many years [2,3,4] but it is only recently that computing resources have been sufficient to spur the derivation of detailed computational schemes to do the same—that is, solve the structure-from-motion problem.

In reality, the notion of one structure-from-motion problem with one solution is quite erroneous. The growing literature on motion processing explores a near plethora of problems and solutions and all show that, with the extra degrees of freedom introduced by unconstrained relative motion between camera and viewed scene, obtaining reliable structure from image motion is difficult. There are several contributory factors. In the first place, obtaining visual motion itself from a sequence of images is prone to error. Secondly, structure-from-motion algorithms are notoriously ill-conditioned with respect to such errors in the visual motion, and thus demand both high quality visual motion and a high quality segmentation of the visual motion. Yet another pitfall is that the visual motion field computed from the imagery may not relate simply to the geometric motion field (i.e., the projected scene motion) because of lighting and occlusion effects [5,6].

Broadly speaking, these difficulties are emphasized when gradient-based methods are used to compute visual motion (e.g. [7,8,9]) and 3D surface structure is recovered, whether that of planar facets [10,11,12] or curved surfaces [13,14,15]. By contrast, the difficulties appear alleviated when token matching schemes are used to compute visual motion and 3D point structure is recovered. Two recent highly successful schemes have used image corners as the matching tokens [16,17]. Unfortunately, the use of distinctive tokens can often result in rather sparse scene descriptions.

In this paper, we describe the combination of algorithms into a system (ISOR) which obtains visual motion from an image sequence and uses it to recover a three-dimensional description of the motion and geometry of the scene in terms of moving extended edges. The system goes on where possible to recognize the recovered geometry as an object from a database of 3D wireframe models. As we will discuss again in our conclusions, the edge-based scheme appears to lie, in terms of both simplicity and richness of scene description, somewhere between schemes based on corners and those based on surfaces. In this work we choose to restrict processing to straight extended edges,
and thus to strictly polyhedral objects. However, the algorithm to compute visual motion operates irrespective of edge contour shape and, in retrospect, we see no reason why the remaining algorithms should not be extended to embrace simple curved primitives.

An overview of the system is given in figure 1, where the four leftmost boxes embrace the principal conceptual tasks of (a) early processing and obtaining visual motion, (b) segmentation, (c) obtaining structure from motion, and (d) model matching. The rightmost box (e) is concerned with model compilation which is used off-line once only for each model in the database.

![Flowchart](image)

Fig. 1. An overview of the system. The four leftmost large boxes embrace the major tasks of (a) low-level processing and obtaining visual motion, (b) image and visual motion segmentation, (c) structure-from-motion and (d) model matching; box (e) describes model compilation. The smaller boxes describe intermediate visual data and representations and the ovals are individual processes (their names should not concern the reader).

For the first task (a) we present a new method developed from the technique of Scott [18]; for the second (b) we develop a pot-pourri of image-based techniques from the literature; for the structure-from-motion stage (c) we describe a new non-linear least-squares method and, finally, for matching (d) we have used the matching paradigm of Grimson and Lozano-Pérez [19,20] in which extensive geometrical modifications have been made to overcome the depth/speed scaling ambiguity inherent in visual motion processing and to allow matching to 3D edge segments [21]. In the following sections 2–5 we describe these tasks in turn. In each section we first discuss the motivation for the task and give a broad outline of the algorithm before giving a detailed description. Each section concludes with the results of applying the algorithm to synthetic imagery. We discuss these results fully in section 6, before showing results from natural imagery of real objects in section 7. In section 8 we arrive at broader conclusions.

The first and last algorithms, those to recover 2D visual motion [22] and for matching 3D edges [21], are explored more fully elsewhere, and our emphasis here is on the functionality of the whole sequence of processing, examining whether methods that appear successful in isolation are sufficiently robust to accept the corrupted data processed at an earlier stage of a bottom-up machine vision hierarchy.

The ISOR system currently operates in 'snapshot' mode—that is, derives all its information from a very short sequence of images—and is envisaged as the stage which will bootstrap analysis over an extended period of time. Recognizing the recovered geometry after so few frames is therefore perhaps premature, but it proves the quality of the data and enables resolution of the depth/speed scaling ambiguity, allowing straightforward comparison of recovered depths and motion with their veridical values.

2 Task (a)—Obtaining Visual Motion

We have already suggested that it is possible to distinguish two classes of scheme for obtaining visual motion from a sequence of images. At the lower level are the intensity-based methods where local visual motion estimates are derived from local changes in the image irradiance. While these methods typically yield visual motion densely throughout the image, they supply only partial information: use of the motion constraint equation [7] necessarily means that information is only obtained about the component of visual motion along the irradiance gradient (the vernier motion), leading to the well-known aperture problem [23]. At a higher level in the visual processing hierarchy are the token matching or correspondence techniques in which features such as corners are tracked and matched over time. Here, if the matching tokens are highly distinctive,
a complete visual motion vector is obtained, but only at a few points in the image. If the matching tokens are less distinctive, greater ambiguity in matching arises inevitably and, at the extreme, a pure gradient scheme can be thought of as matching on featureless tokens where the most that can be recovered using only local information is the vernier motion [24].

Scott [18] has recently introduced a new method of deriving visual motion which successfully combines aspects of the two schemes outlined above. He uses a patch of pixel intensities in one frame as the matching token and searches uniformly in a region around that patch in the next frame, computing a matching strength at each position to build up a spatial matching-strength distribution. By determining the principal axes and moments of the matching strength distribution (a principal axis decomposition—see, for example, appendix A1.9.2 in [25]) the algorithm determines two orthogonal components of the visual motion, together with their associated confidences. If the image region is distinctive, say a blob or corner, the strong matches will be tightly grouped and both components of the visual motion are recovered with high confidence. On the other hand, if the intensity patch forms part of, say, an extended edge, the strong matches will have a similarly extended distribution, and only the component of visual motion perpendicular to the edge is returned with high confidence. The crucial and elegant point is that the principal axis decomposition determines whether the patch of pixels used for matching is a distinctive token or not and therefore to what extent the aperture problem applies.

We use Scott's principal axis decomposition method here but, rather than using pixel intensity patches, we match intensity edgels (elements of intensity edge one pixel long). The edgels themselves are derived using the Canny operator [26] and refined using thresholding with hysteresis. The reasons for using edgels are twofold. First, in our chosen scene domain edgels are sparser than pixels, leading to economies in computation time. Secondly, and more importantly, the edgels are more distinctive tokens than pixels, having both strength and direction, but they still only encode local direction, not curvature. In this way it is still left to the principal axis decomposition to determine whether a group of edgels comprises a distinctive shape token from which both components of visual motion can be found with confidence, or an indistinct token from which only one component can be found with high confidence, as shown in figure 2.

![Figure 2](image)
A further modification is that we symmetrize the matching by using three frames, averaging the spatial matching-strength distribution around an edgel from a central image to the image forward in time with the equivalent, but time-reversed, distribution from the central image to the image backward in time. This temporal smoothing process not only produces more reliable results by suppressing gross mistakes, but also has the significant advantage that the computed visual motion is associated with one particular image and set of edgels. A possible philosophical disadvantage is that, being a temporal smoothing operation, the visual motion lags behind image capture by the frame interval. The algorithm uses a fuzzy relaxation method to compute matching strengths, requiring the specification of initial matching strengths together with a neighborhood compatibility function. After relaxation, the resulting matching strength distribution is analyzed using a principal axis decomposition. We now derive the algorithm in more detail.

2.1. The Visual Motion Algorithm in Detail

By applying Canny’s edge detection operator [26] to each image we derive edgels $i$ with orientation $a_i$ and strength $s_i$ at image positions $r_i$ given to subpixel precision. The first step of the visual motion algorithm is to specify matching strengths from edgels in one frame to edgels in another. In the center frame of the three, the position of edgel $i$ is notionally surrounded by a small neighborhood $N_i$ (which excludes $i$) and around the same position in the forward and backward frames are constructed two larger locales $F_i$ and $B_i$, as shown in figure 3 for the forward frame.

The initial pairwise matching strength from edgel $i$ to an edgel $j$ in the forward frame need only be a reasonable guess, and is defined heuristically by

$$p_{ij}^{(0)}(j \in F_i) = \exp\left(-\frac{1}{2} \left[ \frac{\Delta a^2}{\sigma_a^2} + \frac{\Delta s^2}{\sigma_s^2} \right]\right)$$

where $\Delta a_{ij}$ and $\Delta s_{ij}$ are the differences in orientation (modulo $2\pi$) and strength of the edgels, respectively, and $\sigma_a$ and $\sigma_s$ control the sharpness of the distribution. This heuristic has the desirable characteristic of giving a high matching strength between those edgels which have similar orientation and strength. (We note though that this implies that the rotation per frame is small. However, if there were an estimate of the expected rotation, perhaps from processing an extended sequence, this could be incorporated as an offset to the mode of the distribution.)

The initial matching strengths are then modified iteratively by invoking support from the neighbors, $h \in N_i$, of $i$ using a relaxation formula [27]:

$$p_{ij}^{(t)} = \frac{1}{n(N_i) + 1} \left[p_{ij}^{(t-1)} + \sum_{h \in N_i} \max_{k \in F_h} \{c(i,j,h,k)p_{kh}^{(t-1)}\} \right]$$

Fig. 3. Edgels used in the visual motion matching strategy to the forward frame. Edgel $i$ is in the current central frame, and edgel $j$ lies in $i$’s forward locale, $F_i$. Edgel $k$ lies in the neighborhood, $N_i$ of $i$ in the central frame and edgel $k$ lies in the neighborhood of $j$ in the forward frame.
where $N_j$ is the neighborhood of $j$ in the forward frame and $n(N_j)$ is the number of edges in $N_j$.

The function $c(i,j,h,k)$ describes the compatibility of matching $i$ with $j$ and $h$ with $k$. Now, large fluctuations in the depth or motion of objects are not expected within small displacements in the image, an expectation that can be expressed in terms equivalent to the disparity gradient constraint exploited in human and computer stereovision [28,29]. This invokes a limit $L$ in the matching disparity gradient $\Gamma_D$ defined by

$$\Gamma_D = \frac{1}{2} \left| r_{hk} - r_{ij} \right|$$

where

$$r_{ij} = r_j - r_i$$

An appropriate form of the compatibility function is [30]

$$c(i,j,h,k) = \frac{q(i,j,h,k)}{1 + d(i,j,h,k)}$$

where the numerator is

$$q(i,j,h,k) = \begin{cases} \frac{1}{1 - \Gamma_D^2} & \text{if } \Gamma_D < L \\ 0 & \text{otherwise} \end{cases}$$

and the inclusion in the denominator of the square of the cycloplane separation

$$d(i,j,h,k) = \left| r_{ih} - r_{jk} \right|^2$$

ensures that the compatibility falls off as $i$ and $h$ (or $j$ and $k$) move further apart. All the experiments described here use a limit $L$ of unity.

The derivation of the matching strengths from the central to the backward frame is similar with $F_i$ replaced by $B_j$. However, when computing the moments in the principal axis decomposition, the matching strengths for the forward and backward fields are combined by associating displacements $+r_{ij}$ with $p_{ij}^{(f)}$ for forward matches $i$ to $j$ but displacements $-r_{ij}$ with $p_{ij}^{(b)}$ for backward matches $i$ to $j$.

We now turn to the principal axis decomposition (see [25]) carried out on the matching-strength distribution computed within each locale. After the $n$th iteration of equation (2), one computes the mean displacement of edge $i$ as

$$<\delta r_i> = \frac{\sum_{j \in F_i} p_{ij}^{(f)} r_{ij} - \sum_{j \in B_i} p_{ij}^{(b)} r_{ij}}{\sum_{j \in F_i} p_{ij}^{(f)} + \sum_{j \in B_i} p_{ij}^{(b)}}$$

The deviations, $\Delta r_{ij}$, from the mean are given by

$$\Delta r_{ij} = r_{ij} - <\delta r_i>$$

and are used to construct the $2 \times 2$ scatter matrix

$$[M] = \sum_{j \in F_i} p_{ij}^{(f)} \Delta r_{ij} \Delta r_{ij}^T + \sum_{j \in B_i} p_{ij}^{(b)}(-\Delta r_{ij})(-\Delta r_{ij})^T$$

If $\hat{v}$ represents a unit direction vector in the image plane, then the least-squares error term

$$E^2 = \hat{v}^T [M] \hat{v}$$

is minimized when $\hat{v}$ is the eigenvector, $\hat{v}_1$, corresponding to the smaller eigenvalue of $[M]$. The direction $\hat{v}_1$ is thus the direction along which we have the higher confidence in our estimate of the visual motion and its magnitude in that direction is $<\delta r_i> \cdot \hat{v}_1$, with a confidence given by the inverse of the eigenvalue. The other, lower-confidence, component of visual motion is found using the other eigenvector and eigenvalue. That is, the two orthogonal visual motion components, dubbed major and minor, are

major (higher confidence): $(<\delta r_i> \cdot \hat{v}_1)\hat{v}_1$

minor (lower confidence): $(<\delta r_i> \cdot \hat{v}_2)\hat{v}_2$

2.2. Examples on Synthetic Image Sequences

Before showing results from sequences where the scene has depth variations, we illustrate some of the points made above on a moving Mondrian (figure 4a). At points of high edge curvature we expect to find two high confidence components of the visual motion, whereas along extended edges we expect to recover only the edge normal motion with high confidence. This is exemplified in figure 4b, where we show the components of motion that are recovered with certainty when the Mondrian moves to the top left at $(-1, -1)$ pixels per frame. At most of the corners the full motion has been recovered with certainty, whereas along the extended edges, only the component perpendicular to edge direction is known with confidence.

In figure 5a we show the central image from a sequence of three images of a 'house' translating sideways relative to the camera in the direction of the
Fig. 4. The central image (a) from the sequence of a Mondrian moving toward the top left of the image, and the components of visual motion (b) that could be derived with certainty.

Fig. 5. The central image (a) from the house sequence, the detected intensity edgels before (b) and after (c) thresholding with hysteresis, and the visual motion (d). (There were, to our surprise, 'pinholes' in the image, which gave rise to image motion. This noise was effectively removed by the segmentation stage.)
—x-axis (refer to figure 8 for a definition of the camera geometry). This sequence was synthesized using a CSG body modeler so that we could obtain accurate scene coordinates to compare with our computed results. Each image subtends a total angle to the camera of about 36° and the depth variation from front to back of the house is some ±20% of its mean depth. In figures 5b and 5c we show the edgels detected by the Canny operator before and after thresholding, again for the central frame. Note that the contrast between the two forward-facing roof sections and the roof and front wall of the side porch was so low that these edges did not survive thresholding. In figure 5d are shown the major (i.e., higher confidence) components of the visual motion computed at the edgel positions. Again, as expected for extended edges, the major components are all nearly perpendicular to their associated edge directions. The minor components along the edge direction are small and of such low confidence that their inclusion in the structure-from-motion computation is of little value. In these experiments only one iteration of equation (2) was used: further iteration gave only marginal improvement for the types of imagery used here.

3 Task (b) — Segmenting the Visual Motion

In the human visual system, visual motion is a very strong cue for segmentation, as illustrated by the ease with which we detect camouflaged creatures as soon as they move and by simple but compelling experiments with interpenetrating moving random dot patterns. In computational vision, however, segmentation using properties of the visual motion alone has proved far from straightforward to implement. Most work is based on image properties of the visual motion with little regard being given to the implications in terms of scene motion [31,32,33] (although recently early results of a more systematic study of scene assumptions and visual motion segmentation have been reported [34]). The few methods in which visual motion segmentation is combined with a realistic and global scene reconstruction appear computationally very expensive, involving in one case the use of multiple Hough transforms [35] and in another global optimization using simulated annealing [36].

In the light of the difficulties in segmenting visual motion using its own properties, it is helpful to look for other simpler image properties that might be used to present the visual motion into subsets connected with a feature of scene geometry. Within our present scheme, this is relatively straightforward. Because we have computed visual motion at edgels which we expect to be arranged as extended straight edges, it suffices to segment the edgel positions. Later, in the structure-from-motion algorithm (section 4), we map these straight edges in the image onto straight edges in the scene and hence have a way of relating a particular visual motion vector to a particular piece of scene geometry.

To segment the edgel positions we build a vertex-edge graph in the image using a recipe which borrows much from techniques described in standard texts such as Ballard and Brown [25]. Although the recipe contains mostly reasoned principles, it also uses helpful expedients such as ‘delete short edges.’ Such unjustified heuristic approaches in computer vision have been rightly deplored [37], and we stress that the means used here are only justified by the end result. There are other more principled methods of extracting lines from images (e.g., [38]), locating kinks (e.g., [39]) and so on, which one could use in preference.

3.1. The Edgel Segmentation in Detail

The starting point for the construction of the vertex-edge graph is the set of edgels in the center frame that were used for visual motion matching. In terms of figure 1, this is the upper input to box (b) from box (a).

The edgels are first linked into extended strings as follows:
1. Starting at the strongest unvisited edgel i, links are made to edgels on the left until a break occurs. The leftmost edgel becomes the left end of a new string. (Left and right are defined by looking from the dark to the light side of an edgel.)
2. Links are made to the right from i until a break occurs. The rightmost edgel becomes the right end of the string.
3. If there are no unvisited edgels, continue from (1).

The linking process, similar to [40], is illustrated in figure 6a. Suppose for example that linking is traveling leftwards from an edgel i. The orientation $a_i$ of the edgel is used to determine which of the eight pixel neighbors is directly to the left of it, and this neighbor is labeled C. The two pixels on the dark and light sides of C are labelled D and L. A search is first made for an edgel $j_C$ in C; and, if successful, i and $j_C$ are linked, provided the orientation difference between i and $j_C$ is less than some threshold angle. If the threshold angle is exceeded the string terminates at i.
Fig. 6. Details of the segmentation process. In (a) we show two examples of the choice of neighbors $C$, $D$, and $L$. In the first, edgel $j_c$ exists and a link will be made provided $\Delta a_{ijc}$ is less than the orientation change threshold. In the second example, no edgel exists in $C$, so a search is made in $L$ and $D$. In (b) we show stages in locating substantial kinks in predominantly straight edges at which to insert vertices. In (c) we show linking of a 'loose end' to an existing vertex. The contour is one of constant linking likelihood, indicating that forward growth is favored.

edgel in $C$, a search is made in both $D$ and $L$ and potential links are established if edgels $j_D$ and/or $j_L$ exist and the orientation difference threshold is not exceeded. If only one potential link is found, this is confirmed immediately; but if two exist then the one with the smaller orientation difference is chosen. Naturally, if no potential link exists, the string is terminated.

The next step (figure 6b) is to find points of high curvature along the strings. A signal $S$ is derived by convolving the edge orientation $a$ (now including winding number) as a function of length $l$ along the string with the derivative of a Gaussian, $G$, whose width is a few edgels [41]. That is, for each string we derive
\[ S(l) = \frac{\partial}{\partial l} G \ast a(l) \]  

Extrema in \( S \) whose magnitudes exceed some threshold are marked as kinks. No attempt is made to classify the kink shape: the string is simply broken and a vertex inserted and notional links made on both sides to the string. Each section of string is now called an extended edge.

After removing very short edges, the shape of each remaining edge is determined and, at present, only those found to be straight are considered further. ('Straight' is defined as below some threshold deviation from linearity.) Some edges, for example those at the end of strings, will not be bounded at both ends by vertexes. A search is made around each 'loose end' to find a vertex, or other loose end, to which to link, as illustrated in figure 6c.

The final stages in constructing the vertex-edge graph are to coalesce collinear linked edges and refine vertex positions. The graph, which extends across the entire image, is then split into subgraphs of mutually connected edges. With the edge segmentation complete, it is a trivial matter to import the visual motion at the edgels and attach it to the graph data structure, as indicated by the lower input to box (b) of figure 1.

### 3.2. Example on the CSG House Image

The results of this sequence of operations on the central image of the CSG house sequence are shown and described in figure 7. Figure 7a shows the creation of strings. The loose ends are denoted by diamonds. Notice, for example, that the front edge of the roof has been found as a continuous string. By figure 7b this has been split into two straight edges, labeled 34 and 35, with the insertion of vertex 17. At figure 7c, edge 56, for example, has been grown forward and linked to vertex 17.

It is worth noting that the vertex-edge subgraph of figure 7 does not comprise a complete line drawing of

![Fig. 7. Stages in creating the image vertex-edge graph for the house sequence. In (a) strings have been created and their loose ends marked by diamonds. In (b) vertexes are inserted at points of high curvature and nonstraight lines eliminated. By (c) loose ends have been linked into neighboring vertexes or ends, linked collinear lines coalesced and the position of vertexes recomputed.](image-url)
the object, simply because two edges were extremely weak in the original image and were removed by the thresholding stage of the original edgel map. Any loss of information of this sort must of course worsen the structure-from-motion computation described in the next section, but its success does not rely critically upon completeness of the line drawing from the 2D segmentation, only that there is a reasonable amount of structural linkage.

4 Task (c) — Obtaining 3D Structure from the Visual Motion

At the end of task (b), the representation comprises grouped visual motion (figure 1). The visual motion has been computed at edgel subgraphs of extended edges, that are in turn connected by vertexes in the image. The data is still two-dimensional and in this section we consider the task of computing a 3D representation from it.

We first make two assumptions: (i) that each subgraph is the projection of part of a rigid body and (ii) that straight edges and vertexes in the subgraph map onto straight edges and vertexes in the 3D scene. The parameters describing the scene for each subgraph are thus reduced to $n + 5$—the depths $Z_i$ of the $n$ ends of the edges and the five motion parameters $V$ and $\Omega$, where $V$ is the body's translational velocity and $\Omega$ its instantaneous angular velocity relative to the camera. One cannot compute more than the direction $\hat{V}$ of the rectilinear velocity because of the depth/speed scaling ambiguity inherent in visual motion perception: without higher-level knowledge it is impossible to say whether an object is large, far off, and traveling quickly or small, near to, and moving slowly. As we shall see, this ambiguity is only removed after matching to a model of known size.

Now if we propose values for the $n + 5$ scene parameters, we show in section 4.1 that it is a straightforward matter to project the scene into the image to predict the visual motion we should observe along image edges. Hence we can construct an optimization procedure to find the parameters that best reproduce the observed data, subject to whatever constraints or regularization might be appropriate. For example, one might favor solutions where the angular velocity is small [42] or solutions where scene vertexes lying around closed loops on the vertex-edge graph were coplanar (although this of course expects complete line drawings). In this work we use a nonlinear least-squares algorithm and consider solutions where (i) there are no constraints, (ii) the angular velocity is constrained to a known value.

4.1. The Structure-from-Motion Reconstruction in Detail

Let the end point $i$ of an edge in the scene have 3D coordinates

$$\mathbf{R}_i = (X_i, Y_i, Z_i)$$

Its image position in the camera of figure 8, which has a pinhole lens at the origin, optic axis along the positive $\hat{z}$ axis, and image plane $z = -l$, will be

$$\mathbf{r}_i = (x_i, y_i, -l) = \frac{-l \mathbf{R}_i}{Z_i}$$

a point which will be the end of an edge (i.e., a vertex or 'loose end') in the vertex-edge graph.

Fig. 8. The scene, camera, and image geometries.

The motion of the scene point $\mathbf{R}_i$ with respect to the camera can always be written in terms of the relative translational and instantaneous angular velocities as

$$\mathbf{R}_i = V + \Omega \times \mathbf{R}_i$$

and thus by differentiating equation (15) and using (16) the predicted full visual motion $\mathbf{r}_i = (x_i, y_i)$ at the corresponding image point is
\begin{equation}
\dot{x}_i = -\frac{V_x}{Z_i} - x_i \frac{V_z}{Z_i} - \Omega_z l - \Omega_x y_i \tag{17}
\end{equation}

\begin{equation}
\dot{y}_i = -\frac{V_y}{Z_i} - y_i \frac{V_z}{Z_i} + \Omega_x x_i + \Omega_y l
\end{equation}

\begin{equation}
\dot{z}_i = \left(\Omega_y x_i - \Omega_x y_i\right) \frac{V_z}{l} \tag{17}
\end{equation}

Now at an image point \( r \) lying on the straight edge between the two end-points \( i \) and \( j \), described parametrically by

\begin{equation}
r(\lambda) = \lambda r_j + (1 - \lambda) r_i, \quad (0 \leq \lambda \leq 1) \tag{18}
\end{equation}

the predicted full visual motion will be just

\begin{equation}
r_{\text{pred}}(\lambda) = \lambda r_j + (1 - \lambda) r_i \tag{19}
\end{equation}

But this is just the data we measure, barring the following caveats.

1. The motion algorithm of section 2 computes visual motion at edge positions, \( r_e \), which will not in general lie precisely on the straight line between \( i \) and \( j \), but will be scattered either side of it. To take account of this we approximate the parameter \( \lambda \) by that appropriate to the nearest point on the line: thus,

\begin{equation}
\lambda = \frac{(r_j - r_i) \cdot (r_{\text{pred}} - r_i)}{(r_j - r_i) \cdot (r_j - r_i)} \tag{20}
\end{equation}

2. The visual motion algorithm computes vector components, \( v_{\text{meas}} \), of the full motion at the edge at position \( r_e \). This technicality is overcome by projecting the predicted full motion onto the direction of \( v_{\text{meas}} \):

\begin{equation}
v_{\text{pred}} = r_{\text{pred}}(\lambda) \cdot \hat{v}_{\text{meas}} \tag{21}
\end{equation}

where \( \hat{v}_{\text{meas}} \) is the unit vector in the direction of \( v_{\text{meas}} \).

Hence, after some routine working, an expression for \( v_{\text{pred}} \) in terms of a slightly different set of \( n + 5 \) scene parameters \((\rho, \phi, \Omega, \xi_1, \ldots, \xi_n)\) is

\begin{equation}
\begin{aligned}
v_{\text{pred}} &= \sin \rho \cos \phi [\xi_1(\lambda - 1) - \xi_2 \lambda] / \cos \theta \\
&+ \sin \rho \sin \phi [\xi_3(\lambda - 1) - \xi_4 \lambda] / \sin \theta \\
&+ \cos \rho [\xi_5(\lambda - 1)f_i - \xi_6 \lambda f_j] \\
&+ \Omega_z[(1 - \lambda)f_i + \lambda f_j] + f_i \sin \theta / l \\
&+ \Omega_y[(1 - \lambda)f_i x_i - \lambda f_j y_i - f_i \cos \theta] / l \\
&+ \Omega_z[(1 - \lambda)g_i - \lambda g_j] \tag{22}
\end{aligned}
\end{equation}

where

\begin{equation}
\cos \theta = \hat{x} \cdot \hat{v}_{\text{meas}}, \quad \sin \theta = \hat{y} \cdot \hat{v}_{\text{meas}} \tag{23}
\end{equation}

and

\begin{equation}
f_i = x_i \cos \theta + y_i \sin \theta, \tag{24}
\end{equation}

\begin{equation}
g_i = y_i \cos \theta - x_i \sin \theta
\end{equation}

and similarly for \( j \). The parameters \( \rho \) and \( \phi \) are polar and azimuthal angles describing the translational velocity direction

\begin{equation}
\hat{V} \cdot \hat{x} = \sin \rho \cos \phi, \quad \hat{V} \cdot \hat{y} = \sin \rho \sin \phi, \quad \hat{V} \cdot \hat{z} = \cos \rho
\end{equation}

and the parameters \( \xi_i \) are

\begin{equation}
\xi_i = V/Z_i \tag{26}
\end{equation}

A comparison of the measured and predicted components forms the basis of a least-squares estimation of the scene parameters. However, it is evident from equation (22) that the problem is nonlinear and to solve it we use a procedure due to Marquardt [43], a least-squares method which evolves gradually from a decoupled gradient search to a linearization of the fitting function as it approaches convergence. It demands computation of the \( n + 5 \) first derivatives, \( \partial v_{\text{pred}} / \partial \rho \) etc., of each \( v_{\text{pred}} \) but because only two depths are involved in the expression for any \( v_{\text{pred}} \), \( n - 2 \) of these are identically zero.

4.2. Example of the Method on the House Sequence

The algorithm was applied to the segmented visual motion from the house sequence (figure 6d) and several views around the resulting 3D partial wireframe are shown in figure 9. The reader's own ability to interpret images will indicate that the recovered structure is sound and we defer a full discussion of the values of the computed velocities and depths until after resolution of the depth/speed scaling ambiguity during model matching in the next section.
5 Task (d)—Geometric Model Matching

The data input to this task from the structure-from-motion stage comprise the computed 3D positions of the endpoints of the edges, as shown in figure 1. It is important to note, however, that these 3D positions are still scaled by the speed—that is, we use \(1/\xi\) values as "depths" under the temporary assumption that the speed is unity.

The 3D geometry of these straight edges (e.g., figure 9) is well-suited to matching to a CAD 3D wireframe model. Murray and Cook [21] have described the geometry required to use the generate-and-test algorithm of Grimson and Lozano-Pérez [19,20] to match a partial data wireframe to a complete model wireframe and we give only a brief resume here.

The method of Grimson and Lozano-Pérez involves a quasi-exhaustive search of the interpretation space, using simple consistency checks between the geometry of pairs of data and the geometry of the pair of potential matches on the model to establish facts like "if edge datum \(a\) is matched to model edge \(i\) then (edge datum \(b\) can/cannot be matched to model edge \(j\))." The "cannot" answers, of which there should be many, are remarkably effective in quenching the otherwise unbridled combinatorial explosion of the matching search space. This process usually generates a small number of feasible interpretations of the data, which must then be tested for global geometric consistency by determining the rotation, translation, and also, because of the depth/speed ambiguity in our case, scaling of the model to fit the data.

Similar graph-matching approaches have been taken by several other researchers, and recently Pollard et al. [44] combined the best features of several of these—the pair constraints of Grimson and Lozano-Pérez [19,20], the maximal clique search of Bolles et al. [45], and the early geometric transformation of Faugeras and Hébert [46]—to improve search speed. They also consider geometry for 3D edges, but unfortunately their work employs absolute depths and size (obtained from stereo processing), a constraint that is unavailable in our case.
5.1. Generating Feasible Interpretations

Murray and Cook [21] use an angle constraint and three direction constraints to impose consistency between the model edges and data edges, based on the body-centered data metrics

\[ \hat{e}_a \cdot \hat{e}_b \quad \hat{e}_a \cdot \hat{p}_{ab} \quad \hat{e}_b \cdot \hat{p}_{ab} \quad \hat{e}_{ab} \cdot \hat{p}_{ab} \quad [27a] \]

and their equivalents on the model

\[ \hat{m}_i \cdot \hat{m}_j \quad \hat{m}_i \cdot \hat{q}_{ij} \quad \hat{m}_j \cdot \hat{q}_{ij} \quad \hat{m}_i \cdot \hat{q}_{ij} \quad [27b] \]

Here \( \hat{e}_a \) and \( \hat{e}_b \) are data edge directions, \( \hat{p}_{ab} \) is a unit vector pointing between any two positions on edges \( a \) and \( b \), and \( \hat{e}_{ab} \) is a unit vector in the direction of \( \hat{e}_a \times \hat{e}_b \), and similarly for the model vectors as sketched in figure 10. In addition, the constraints determine consistency between the arbitrary but fixed signs of edges on the model and the arbitrary signs given to the data edges before matching. That is, where possible the matcher determines which end of an edge fragment is nearer which end of a model edge. This resolution of the edge fragments' director/vector ambiguities increases the power of the constraints as the search ventures deeper into matching space.

As an example of the use of constraints [21], we outline here the simplest, the angle constraint. It demands that if edge fragments \( a \) and \( b \) are assigned putatively to model edges \( i \) and \( j \) then the range of possible angles between sensed fragments must embrace the angle between the edges on the model.

On the model all edges have an arbitrarily chosen, but unambiguous sign. If \( \hat{m}_i \) and \( \hat{m}_j \) are unit vectors in the direction of the model edges, the angle between them is

\[ A_{ij} = \cos^{-1}(\hat{m}_i \cdot \hat{m}_j) \quad [28] \]

These model values are compiled once only off-line and stored in look-up tables, as indeed are the model data for the remaining constraints.

The signs of the data fragments however are initially ambiguous. If we arbitrarily assign a vector \( \hat{e}_a \) to the direction of fragment \( a \), its actual direction (i.e., that consistent with the model) could be \( \pm \hat{e}_a \). To take account of this we maintain a table of the signs of edges, putting them into five categories. The sign of \( a \) can be

\[ (U) \quad (+) \quad (-) \quad (+n) \quad (-n) \]

for uncertain, definitely \( +\hat{e}_a \), definitely \( -\hat{e}_a \), and of the same sign as, and different sign from, edge fragment \( n \). With no knowledge of the signs, there are then two possibilities for the sensed angle between fragments \( a \) and \( b \). If the fragments have the same sign the angle is

\[ \gamma_{ab} = \cos^{-1}(\hat{e}_a \cdot \hat{e}_b) \quad [29] \]

and if different

\[ \gamma_{*ab} = \cos^{-1}(-\hat{e}_a \cdot \hat{e}_b) = \pi - \gamma_{ab} \quad [30] \]

Including sensing error angles \( \alpha_a \) and \( \alpha_b \), for a valid pairing at least one of two logical expressions must be satisfied:

\[ l_s = \max \left[ (\gamma_{ab} - \alpha_a - \alpha_b), 0 \right] \leq A_{ij} \]

\[ \leq \min \left[ (\gamma_{ab} + \alpha_a + \alpha_b), \pi \right] \]

OR

\[ l_d = \max \left[ (\gamma_{*ab} - \alpha_a - \alpha_b), 0 \right] \leq A_{ij} \]

\[ \leq \min \left[ (\gamma_{*ab} + \alpha_a + \alpha_b), \pi \right] \]

[31]
The subscripts \( s \) and \( d \) indicate that these are the satisfaction conditions for when fragments \( a \) and \( b \) have the same or different signs. However, it is important to note that because of measurement uncertainties the logical OR above is not exclusive.

There are several output conditions depending on the fragments’ signs input to the constraint [21]. Suppose that:

- **on entry the signs of** \( a \) **and** \( b \) **are both uncertain** \((U)\). Then, if \( l_s \) is true and \( l_d \) false, the pairing is valid and the two edge fragments \( a \) and \( b \) must have the same sign and can be relabelled \((+b)\) and \((+a)\), respectively. Conversely, if \( l_s \) is false and \( l_d \) true, then the pairing is valid and the two edge fragments \( a \) and \( b \) must have different signs and can be relabelled \((-b)\) and \((-a)\), respectively. If both \( l_s \) and \( l_d \) are true, nothing is learned about the signs; and if both \( l_s \) and \( l_d \) are false, the pairing is invalid and the search backtracks.

- **the sign of one edge fragment is known on entry**, say \( a \) is signed \((-)\). Then when \( l_s \) is true and \( l_d \) false, fragment \( b \) must also have sign \((-)\). Conversely, if \( l_s \) is false and \( l_d \) true, fragment \( b \) must be signed \((+)\). If both tests succeed, the sign of \( b \) remains \((U)\); and if both fail, the pairing is invalid.

- **fragments** \( a \) **and** \( b \) **are signed as** \((-n)\) **and** \((U)\) **on entry**. Then, if \( l_s \) is true and \( l_d \) false, \( b \) must be signed \((-n)\), and so on. If \( a \) and \( b \) are signed \((-n)\) and \((+m)\) on entry, then if \( l_s \) is true and \( l_d \) false we know that \((+m) = (-n)\), and so on.

- **both signs are known absolutely on entry**. If the signs are identical, then for a valid pairing \( l_s \) must be true. If different, \( l_d \) must be true.

Whenever a sign is changed a check is made for any other signs that depend on it and they are updated recursively.

We note that, given two edge fragments with uncertain signs, the angle constraint can never determine the absolute sign for each fragment because it involves a product. However, the second and third constraints of expression (27) involve only one edge vector and so can determine the absolute sign of that edge. Thus these latter direction constraints determine signs and the angle constraint and third direction constraint propagate them.

### 5.2. Matching to the House Partial Wireframe

The results of matching the house geometry to a small database of wireframe models are shown in figure 11. Only one feasible interpretation was found during this generation stage and this is shown in detail in figure 11a. By inspection of the edge labels and directions, the interpretation is correct, but this is formally tested below. Figure 11b shows the remaining models from the database for which no feasible matches were found.

### 5.3. Testing the Feasible Interpretations

The pairwise constraints do not guarantee that a feasible interpretation is consistent with a single global transformation between model and sensor spaces. Therefore each interpretation is tested by finding the rotation, scaling, and translation \((\mathbf{R}, S, t)\) which link model \((\mu)\) and sensor \((\sigma)\) spaces as

\[
\sigma = \mathbf{S}[\mathbf{R}]\mu + t
\]  

The transformation is computed as an average over all the matched edges, and then we demand that each edge after transformation lies within some threshold distance of its matched edge on the model. In the experiments here the threshold was chosen as 10% of the largest vertex-vertex distance on the model. Where there are multiple valid interpretations, comparison of the goodness of fit to the model gives an indication of the best overall interpretation. We first determine the rotation matrix using the quaternion method described by Faugeras and Hebert [46], and then find the scaling and translation simultaneously in a least squares procedure [21].

### 5.4. Resolving the Depth/Speed Scaling Ambiguity to Locate the Object

If all the objects in the model database have different shapes, the computed scale \( S \) resolves the depth/speed scaling ambiguity and hence allows absolute location of the object and determination of its speed. Recall that in the matching stage we used as ‘depths’ the \( 1/s \) values, assuming for convenience that the speed was unity. However, using the derived scale factor the absolute values of depth and speed \((Z_{abs} \text{ and } V_{abs})\) must be

\[
Z_{abs} = \frac{1}{S} \quad V_{abs} = \frac{1}{S}
\]  

These allow immediate comparison with the real-world values for the depth and motion of the CSG model, and this is shown in table 1.
Fig. 11. (a) Shows the single valid match found for the 3D edge data of the house (top) to the model (below). The table gives the derived mapping from data to model, together with the relative vector signs. (On the model and data, the larger numbers label edges to their left. On the model, the smaller numbers label vertices and are used again in tables 1 and 2. (b) Shows a library of models—a plug, vdu, chipped block, slider, and bead—to which there was no valid match.
Table 1. The vertical motion (Ω in radians per frame and V in depth units per frame) and scene vertex coordinates (R) compared with those computed by the system for the house sequence when the angular velocity is allowed to vary. The depths (R × z) have computed errors of ±1.5% and all lie within a single standard deviation of the veridical values. In braces are the corresponding vertex indices on the model.

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Veridical</th>
<th>Computed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ω</td>
<td>1.25</td>
<td>0</td>
</tr>
<tr>
<td>V</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>R(10)</td>
<td>12.9</td>
<td>-7.8</td>
</tr>
<tr>
<td>R(15)</td>
<td>20.0</td>
<td>0.0</td>
</tr>
<tr>
<td>R(12)</td>
<td>-11.8</td>
<td>-5.3</td>
</tr>
<tr>
<td>R(17)</td>
<td>-18.9</td>
<td>2.5</td>
</tr>
<tr>
<td>R(18)</td>
<td>-18.9</td>
<td>11.7</td>
</tr>
<tr>
<td>R(21)</td>
<td>-15.4</td>
<td>7.8</td>
</tr>
<tr>
<td>R(9)</td>
<td>27.0</td>
<td>2.2</td>
</tr>
<tr>
<td>R(11)</td>
<td>1.2</td>
<td>2.3</td>
</tr>
<tr>
<td>R(2)</td>
<td>12.9</td>
<td>19.1</td>
</tr>
<tr>
<td>R(13)</td>
<td>-26.0</td>
<td>4.7</td>
</tr>
<tr>
<td>R(4)</td>
<td>-11.8</td>
<td>16.6</td>
</tr>
<tr>
<td>R(6)</td>
<td>-18.9</td>
<td>16.6</td>
</tr>
<tr>
<td>R(19)</td>
<td>-29.5</td>
<td>4.2</td>
</tr>
<tr>
<td>R(5)</td>
<td>-15.4</td>
<td>14.1</td>
</tr>
<tr>
<td>R(16)</td>
<td>1.2</td>
<td>15.0</td>
</tr>
<tr>
<td>R(20)</td>
<td>-26.0</td>
<td>-0.3</td>
</tr>
<tr>
<td>R(1)</td>
<td>27.0</td>
<td>-9.1</td>
</tr>
<tr>
<td>R(3)</td>
<td>1.2</td>
<td>-9.1</td>
</tr>
<tr>
<td>R(7)</td>
<td>-29.5</td>
<td>9.1</td>
</tr>
<tr>
<td>R(14)</td>
<td>-1.2</td>
<td>22.2</td>
</tr>
</tbody>
</table>

6 Discussion

Before showing results from natural imagery, we look here more broadly at the results we have presented so far.

Although the motion and depth values presented in table 1 show good agreement with the real-world values, there is a systematic discrepancy. The source of this is the small, but significant, value of the angular velocity component, Ω. Snapshot processing of a short sequence of images is particularly soft to coupling between translational velocity and appropriate components of the angular velocity. Scrutiny of equations (17) shows that it is possible to convert purely translational and purely rotational motion fields that differ only in second order (i.e., in $x^2$, $xy$, or $y^2$), making it difficult to decide whether the scene motion is translational or rotational, especially when the angle of view is small.

In many practical circumstances, however, it is possible to measure the angular velocity of the camera, say using gyroscopic sensing, and so in a second analysis of the same sequence we have constrained the angular velocity to its known value of zero in the structure from motion calculation. The recovered partial wireframe is shown in figure 12 and the motion and depth values are given in detail in table 2: they show better overall agreement. If it is not possible to determine the angular velocity by other sensory means, two courses of action are open to disambiguate the types of motion. Most obviously, one could collect further frames, effectively increasing the viewing angle, and volume and quality of data. However, a second approach, once the data has been matched, would be to refine the scene motion so as to reduce the systematic deviations of the recovered partial wireframe from the known shape of the matched model.

Table 2. Veridical motion (V in depth units per frame) and scene vertex coordinates (R) compared with those computed by the system for the house sequence when the angular velocity is fixed. The depths (R × z) have computed errors of ±1.5%. In braces are the corresponding vertex indices on the model.

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Veridical</th>
<th>Computed</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>1.25</td>
<td>0</td>
</tr>
<tr>
<td>R(10)</td>
<td>12.9</td>
<td>-7.8</td>
</tr>
<tr>
<td>R(15)</td>
<td>20.0</td>
<td>0.0</td>
</tr>
<tr>
<td>R(12)</td>
<td>-11.8</td>
<td>-5.3</td>
</tr>
<tr>
<td>R(17)</td>
<td>-18.9</td>
<td>2.5</td>
</tr>
<tr>
<td>R(18)</td>
<td>-18.9</td>
<td>11.7</td>
</tr>
<tr>
<td>R(21)</td>
<td>-15.4</td>
<td>7.8</td>
</tr>
<tr>
<td>R(9)</td>
<td>27.0</td>
<td>2.2</td>
</tr>
<tr>
<td>R(11)</td>
<td>1.2</td>
<td>2.3</td>
</tr>
<tr>
<td>R(2)</td>
<td>12.9</td>
<td>19.1</td>
</tr>
<tr>
<td>R(13)</td>
<td>-26.0</td>
<td>4.7</td>
</tr>
<tr>
<td>R(4)</td>
<td>-11.8</td>
<td>16.6</td>
</tr>
<tr>
<td>R(6)</td>
<td>-18.9</td>
<td>16.6</td>
</tr>
<tr>
<td>R(19)</td>
<td>-29.5</td>
<td>4.2</td>
</tr>
<tr>
<td>R(5)</td>
<td>-15.4</td>
<td>14.1</td>
</tr>
<tr>
<td>R(16)</td>
<td>1.2</td>
<td>15.0</td>
</tr>
<tr>
<td>R(20)</td>
<td>-26.0</td>
<td>-0.3</td>
</tr>
<tr>
<td>R(1)</td>
<td>27.0</td>
<td>-9.1</td>
</tr>
<tr>
<td>R(3)</td>
<td>1.2</td>
<td>-9.1</td>
</tr>
<tr>
<td>R(7)</td>
<td>-29.5</td>
<td>9.1</td>
</tr>
<tr>
<td>R(14)</td>
<td>-1.2</td>
<td>22.2</td>
</tr>
</tbody>
</table>

Although speed was not one of the issues being addressed in our work and little of the code has been optimized, we show for interest, in table 3, the cpu time taken for the entire processing on a Sun 3/160 with floating point accelerator. We note that the visual motion computation and the segmentation can run concurrently, and that the Canny edge detector and thresholding
are being implemented in video-rate hardware by Ruff [47]. The figures for model matching may be somewhat misleading, because we have not considered a method for invoking likely models to which to match from a larger database. In general, this seems to be an underexplored area in vision and one which already threatens as a computational bottleneck. However, the algorithm with most immediate heavy computational cost is that for determining visual motion. For dense uniform texture the algorithm can scale as $r_L r_N^2$, where $r_L, r_N$ are the radii of the locale and neighborhood, respectively, though for sparse edgemaps dominated by linear features of the sort used here, the scaling is far better, of order $r_L r_N^3$. We see two potential palliatives: first the method is parallel and, secondly, it should be possible to direct the search for matches, either by using prior motion values from processing an extended sequence, or by adopting a hierarchical multiscale approach.

### Table 3. Processing times on a Sun 3/160 with floating-point accelerator (circa 3 Mips).

<table>
<thead>
<tr>
<th>Function</th>
<th>cpu seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge extraction</td>
<td>3 x 98</td>
</tr>
<tr>
<td>yields 3,700 edgedgels</td>
<td></td>
</tr>
<tr>
<td>Edge thresholding</td>
<td>3 x 1.5</td>
</tr>
<tr>
<td>yields 1,200 edgedgels</td>
<td></td>
</tr>
<tr>
<td>Visual motion</td>
<td>215</td>
</tr>
<tr>
<td>using 1200 edgedgels &amp; 1 iteration</td>
<td></td>
</tr>
<tr>
<td>Segmentation</td>
<td>24</td>
</tr>
<tr>
<td>using 1200 edgedgels</td>
<td></td>
</tr>
<tr>
<td>Structure from motion</td>
<td>113</td>
</tr>
<tr>
<td>14 iterations</td>
<td></td>
</tr>
<tr>
<td>Matching to house model</td>
<td>126</td>
</tr>
<tr>
<td>28 data edges; 33 model edges</td>
<td></td>
</tr>
<tr>
<td>Null matching to other models</td>
<td>113</td>
</tr>
<tr>
<td>plug</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td></td>
</tr>
<tr>
<td>vdu</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>block</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
</tr>
<tr>
<td>slider</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td></td>
</tr>
<tr>
<td>bead</td>
<td></td>
</tr>
</tbody>
</table>

### 7 Real Objects and Natural Imagery

It was noted in the introductory section that structure-from-motion algorithms are typically ill conditioned—that is, the scene reconstruction is very sensitive to noise in the input data. Noise arises not only because of electrical and optical distortions in the camera but also because of deficiencies in the scene and image descriptions. To examine the performance of the sequence of algorithms in the presence of significant noise we describe here experiments on natural images of a (i) wooden chipped block and (ii) a toy truck.
7.1. The Chipped Block

The chipped block was carpentered from wood with overall size approximately 164×75×58 mm³ and painted with a light matte paint. The images were captured from a CCD camera with 512×512 rectangular pixels which were resampled in software to produce 512×384 images with square pixels, from which the central 384×384 sections were used for experiment. No special calibration of the camera was performed. It was assumed that the optic axis passed through the center of the captured image, that the pixel width was given by the nominal width of the CCD chip of 8.8 mm divided into 512 pixels and that the focal length of the lens was the quoted 24 mm and that the optics gave perfect perspective images. In all the experiments the full cone angle subtended by the object to the camera did not exceed 16°.

In the first experiment the object was positioned at a mean distance of about 415 mm in front of the camera and translated vertically along +y at 3 mm-frame⁻¹. The block was illuminated by natural light. The central frame from the three-image sequence along with its intensity edges are shown in figures 13a–c. It can be seen that the somewhat rounded (physical) edges of the object have combined with lighting and reflectance effects lead to rather meandering intensity edges, an example of noise introduced by incomplete modeling of the scene and image. The computed visual motion and stages in the segmentation are shown in figures 13d–f. In figure 14 we show views around the 3D partial wireframe computed by the structure-from-motion algorithm which here included variation in the angular velocity. The matching of data to model is shown in figure 15, and from the computed scale we derive the motion and depths shown on the left-hand side of table 4.

Here again we see the effects of angular velocity coupling, (this time the depths with Ωχ) further exacerbated by the small camera angle. The direction of the translation velocity is found correctly, but its magnitude is overestimated as 7mm-frame⁻¹, the discrepancy being largely made up of Ωz which is ~3 mm-frame⁻¹. In effect, the system believes the camera is panning upward, tracking the block and so, to recreate the observed visual motion, the block must be translating faster than it really is.

When the angular velocity is constrained the processing through to matching proceeds similarly, but the motion and depth values obtained are shown in the right-hand side of table 4. The speed is now better estimated as 2.9 mm-frame⁻¹ and the midpoint depth

Fig. 13. The central image (a) from the sequence of the block moving along +y. The edge maps and computed visual motion are shown in (b), (c), and (d). (e) and (f) show the initial and final stages of segmentation.
Table 4. The motion (Ω in radians per frame and V in mm per frame) and scene vertex coordinates (R in mm) computed by the system for the block translating along +y. On the left the angular velocity has been allowed to vary. On the right it has been fixed, and the recovered values are close to reality. The depths (R · d) and speed have computed errors of ±12% (left) and ±3% (right). In braces are the corresponding vertex indexes on the model.

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Variable rotation</th>
<th>Fixed rotation (near veridical)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>Ω</td>
<td>5e-03</td>
<td>-6e-04</td>
</tr>
<tr>
<td>V</td>
<td>0.45</td>
<td>7.20</td>
</tr>
<tr>
<td>R(5)</td>
<td>19</td>
<td>-6</td>
</tr>
<tr>
<td>R(10)</td>
<td>21</td>
<td>-61</td>
</tr>
<tr>
<td>R(4)</td>
<td>-31</td>
<td>27</td>
</tr>
<tr>
<td>R(3)</td>
<td>-73</td>
<td>29</td>
</tr>
<tr>
<td>R(2)</td>
<td>-69</td>
<td>-55</td>
</tr>
<tr>
<td>R(6)</td>
<td>-99</td>
<td>45</td>
</tr>
<tr>
<td>R(7)</td>
<td>83</td>
<td>-13</td>
</tr>
<tr>
<td>R(10)</td>
<td>83</td>
<td>63</td>
</tr>
</tbody>
</table>

is found as 420 mm. Views of the reconstructed wireframe are given in figure 16.

In a further experiment the block was placed in a similar position in front of the camera but moved directly away from the camera at 10 mm-frame⁻¹. The central image of the sequence, the intensity edge maps, the computed visual motion, and segmentation are shown as figure 17.

Examination of equations (17) shows that it is not possible to simulate a purely expanding or contracting visual motion field with rotational terms to any order, and so we expect here that the structure-from-motion algorithm should be more tolerant of variable angular velocity in this experiment. This is indeed the case: in figure 18 we show the reconstructed 3D wireframe, and after matching the twelve visible edges (figure 19) we obtain the absolute depth and motion values of table 5. The translational velocity is now within two standard deviations of its veridical value and the midpoint depth is found as 430 mm.

7.2. The Toy Truck

We now demonstrate the results of running ISOR on a sequence of a more complicated object, a toy truck. These experiments were performed with another camera which had similar resolution to that above, but required calibration to locate the optic center of the image array. Figure 20a shows the central image from
Fig. 16. The recovered partial wireframe for the block when the angular velocity was fixed to its known value during the structure-from-motion stage.

Fig. 17. A set of processing stages similar to figure 13, but with the block moving directly away from the camera along $+z$. 
Fig. 18. Views of the recovered partial 3D wireframe for the block when moving away from the camera. The angular velocity was allowed to vary.

Fig. 19. The only match for the block data of figure 18 to the model library of figure 11b.
Table 5. The motion (Ω in radians per frame and V in mm per frame) and scene vertex coordinates (R in mm) computed by the system for the block translating away from the camera along +z. The angular velocity has been allowed to vary; but because it is more difficult for the angular velocity to couple, the recovered motion and depths are close to their veridical values. The depths and speed have a computed error of ±12%.

<table>
<thead>
<tr>
<th>Quantity</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ω</td>
<td>4e-04</td>
<td>1e-03</td>
<td>7e-04</td>
</tr>
<tr>
<td>V(4)</td>
<td>29</td>
<td>7</td>
<td>353</td>
</tr>
<tr>
<td>R(4)</td>
<td>-11</td>
<td>33</td>
<td>348</td>
</tr>
<tr>
<td>R(5)</td>
<td>29</td>
<td>-38</td>
<td>359</td>
</tr>
<tr>
<td>R(3)</td>
<td>-48</td>
<td>35</td>
<td>372</td>
</tr>
<tr>
<td>R(2)</td>
<td>-47</td>
<td>-34</td>
<td>382</td>
</tr>
<tr>
<td>R(6)</td>
<td>43</td>
<td>44</td>
<td>405</td>
</tr>
<tr>
<td>R(10)</td>
<td>66</td>
<td>61</td>
<td>507</td>
</tr>
<tr>
<td>R(7)</td>
<td>65</td>
<td>-4</td>
<td>512</td>
</tr>
<tr>
<td>R(9)</td>
<td>-3</td>
<td>61</td>
<td>516</td>
</tr>
</tbody>
</table>

an image sequence where the toy truck translates past the camera along the negative x-axis at 5 mm-frame⁻¹. The computed visual motion and the segmentation are shown in figures 20b and 20c. Comparing these figures shows the removal of curved edges by the segmentation process. The company logo on the side of the truck forms a separate subgraph in the segmentation as it is not connected to the truck’s main outline. The structure recovered with constrained angular velocity is shown in the views of figure 20d, and after matching we obtained absolute depths correct to ±10%. The translational velocity was found to lie within 0.6° of the x-axis and the speed was recovered as 5.3 mm-frame⁻¹.

Figure 21 illustrates the visual motion and 3D partial wireframe recovered when the truck moved directly toward the camera. Compared with figure 20 there is considerably increased distortion in the wireframe, and we note that a similar comparison can be made between the two different motion examples for the block (cf. figures 18 and 16). This distortion arises because the structure-from-motion algorithm is sensitive to the increased noise in the visual motion as the motion becomes very small around the focus of expansion.

8 Conclusions and Discussion

Our conclusions can be stated simply. We have a first attempt at a system that can travel up the computational hierarchy of representations from the pixel grey levels in a sequence of 2D images, through 2D edge representations and visual motion to 3D edge representations and scene motion and, finally, to symbolic descriptions of those moving 3D edges.

Although each algorithm is successful in itself, the linch-pin of the system’s ability to obtain accurate depth and motion values is the success of the segmentation and grouping of the visual motion into structures related to the scene. The problem is of course greatly eased because of our restricted world domain but there are, nevertheless, difficulties remaining.

The principal one is captured by the line drawing of figure 22 which shows one cuboid (B) in the background occluded by another (F) in the foreground. Is cuboid B rigidly attached to F or not, and are there depth discontinuities at the points X or not? Although it is possible to devise methods of detecting occlusions in the line drawings using relatively high-level knowledge, a much more attractive scheme is to detect occlusions by searching for anomalies in the visual motion at the virtual vertex at the image junction of the occluding and occluded lines. This information is available at a low level in the matching probability surfaces and work is continuing to elicit it.

It is worth returning to the more general importance of segmentation in visual motion processing. If our laboratory is typical, there must be many elegant ways of determining visual motion which languish unused simply because it is difficult to find ways of usefully relating the visual motion in the image to its originating structure in the scene. Similarly, many structure-from-motion algorithms must lie redundant merely because there is no way of segmenting the geometry-related visual motion they require from the image. If detailed 3D structure is required from visual motion it appears that one must be able to link image representations associated with the visual motion to the 3D geometric representations associated with the scene. In this work, we have used a consistent representation of 2D and 3D edges and vertices throughout the processing. This provides a strong set of geometrical constraints which allows us to identify visual motion with the geometrical visual motion field from physical structures moving in the scene and provides stability to the structure-from-motion computations. In these circumstances, we can indeed recover accurate and recognizable 3D geometry and motion.

Inherently more difficult to link up in this way are representations based on surfaces. For surfaces one must detect 3D motion and orientation boundaries in
Fig. 20. The central image (a), the visual motion (b), the segmentation (c), and views of the wireframe (d), from a sequence where a toy truck moves past the camera along the \(-z\)-axis.
Fig. 21. The visual motion (a) and views of the wireframe (b) computed when the truck moved directly toward the camera. The increased distortion in this figure compared with figure 20d is similar to that between figures 18 and 16.

Brady [48] has made a similar observation for several other areas of vision, noting the decline in information content as one progresses from corner features to edge and line features and finally to surface features, and dubbing corner features 'seeds of perception.'

There is clearly a limit to the quality of information one can obtain using 'snapshot' processing. To hope to recover complete and accurate structure and motion of a scene on the basis of a handful of frames, which

the visual motion which, in the absence of other clues about the 3D, implies a simultaneous solution of the segmentation and structure-from-motion problems. At the extremes of difficulty are the far less structured environments, for example, natural scenes of trees, where it is difficult to define what we mean by segmentation let alone perform it. Perhaps in these circumstances all we should hope for are qualitative notions of scene depth and motion.
Fig. 22. The problem of occlusions.

at video rates would span say 60 m, is rather optimistic and it is indeed gratifying that the algorithms presented here perform as well as they do. The experiments show clearly the impossibility of disambiguating certain components of translational and rotational motion when the sequence embraces only a small viewing angle. However, the high quality of the 3D data recovered here in bootstrap mode augers well for the extension of the system into the continuous time domain.

Acknowledgements

This work was part funded by the UK Alvey Programme as IKBS Project 013 Spatio-temporal processing and optical flow for computer vision. The authors thank Al Vision Research Unit at Sheffield University for the use of the Canny code and IBM UK for the CSG body modeller. Part of the experimental work was carried when one of us (DWM) was GEC Visiting Fellow at University College, Oxford and he wishes to thank the College, the Department of Engineering Science, and Professor Mike Brady for providing facilities.

References


