Fitting multiple models to multiple images with minimal user interaction

No Author Given

No Institute Given

Abstract. Utilising the broad range of information which human observers bring to bear when interpreting their visual environment is currently infeasible for artificial vision systems. We propose instead a method for modelling compound structures which intelligently divides this prior information into that which may be applied by the system and that which may not. Models are fitted to the input data on the basis of 2D and 3D image-based measures, but also as directed by a prior whose specification is split between the human and the system. Importantly this split is carried out in a manner which minimises the human input required.

1 Introduction

Visual inspection of a scene provides a person with a wealth of information about the objects it contains. This might include the objects' shape, their identity, whether they are likely to be attached to each other, and how they might fit together, for instance. This information is fundamentally more useful for guiding interaction than the reconstruction obtained by an artificial vision system, which is typically a point cloud or texture mapped mesh. A reconstruction that contains some semantic information opens up new possibilities such as object removal in video, or the interaction of real and computer-generated elements in film post-processing. This level of scene description might also be used to guide lower level image-based processes such as dense matching, segmentation, or image-based rendering in the same way that higher level information is used in the human visual system to guide low level processing.

A description of a scene in terms of the objects it contains and the relationships between them is thus a more useful result of image analysis than a 3D point cloud, but it is also more difficult to generate. Some of the information required to generate such a model of the scene may be extracted purely from images of the scene, but some cannot be derived without significant prior experience of the types of objects depicted. Humans are extremely adept at acquiring and applying this prior experience to visual interpretation, but artificial vision systems lag far behind. One solution to the problem has been to restrict the application of algorithms to domains such line drawings of blocks [1]. By restricting the domain of application it becomes possible to encode all of the information necessary to interpret the scene, but at the cost of decreasing the utility of the method.

Rather than attempt to encode all prior information necessary to interpret a scene on the basis of a set of images, we aim to divide this information into two parts—that which may be readily represented and applied automatically, and that which may not. The prior information which is unsuitable for automatic application is provided by the human operator, and guides the application of the machine-applicable portion. This allows the recovery of a semantic model-based representation of the scene, but requires some user input. The key to minimising the required human input is to apply as much information as possible automatically. Toward this goal the method we present here is capable of utilising both 2D and 3D information in order to fit parameterised models to a scene on the basis of a set of images. This information is combined with user interaction in order to determine the most appropriate scene interpretation.

Importantly, the method is also capable of propagating information from one object to another on the basis of the relationship between them, which is crucial to scene (rather than object) understanding. The possible relationships between objects are represented by the automatically applied prior, and used to automatically direct attention to locations more likely contain items of interest. If an object normally sits on a surface, for example (instead of hanging in space), it is likely that there will be a planar surface in the vicinity of the object. The prior on the first object thus guides the search for the second.

The process of user-driven model-based scene analysis is not new. The Facade system [2], for example, reconstructs architectural scenes as a collection of polyhedra, but requires the user to outline each block in each image, and manually label corresponding features in each image—a time consuming process. PhotoModeler [3], a commercially available package for generating 3D models from photographs, has similar requirements. By contrast our system can identify a block with a single mouse click in one image. Photobuilder [4] requires the user to highlight enough lines in each image to identify vanishing points in 3 orthogonal directions. Again, this is quite tedious and relies on there being visible sets of parallel lines in 3 orthogonal directions. [5] operates on a similar principle, given a single image and significant user markup as input, while [6] relies on manually marking lines with known directions in each image. The work of Criminisi [7] on single view metrology creates partial reconstructions from scene constraints provided by the user. However it is not able to infer properties about parts of the scene not labeled by the user. A more general approach to interactive modelling is given in [8], based on parallelepipeds as scene primitives. However this still requires the corners of the primitives to be marked manually in each image. Other work which has made extensive use of prior information but is not interactive is necessarily limited in the range of scenes it can reconstruct [9].

There has been great progress in visual tracking by matching the projection of a 3D model with image information [10, 11]. In [12], on the other hand, a parameterised set of interrelated object models is fitted to image sets on the basis of photoconsistency, without the explicit use of priors or 3D or 2D features. Model fitting to range data is used to match either a pair of 3D point clouds [13] (and thereby estimate their relative pose and orientation) or a 3D point cloud and a pre-existing 3D model [14].

The method presented here is novel in number of ways. Firstly, it models objects in 2-dimensions and in 3-dimensions and uses both to inform the fitting process. Secondly, it allows the user to provide high level scene information, but requires the minimum interaction possible to achieve the desired accuracy of fit. Thirdly, it uses a combination of optimisation and sampling techniques to provide the user with the best model estimates available at any moment without compromising control. Finally, information that is common to multiple objects in a scene is shared between them in order to maximise the efficiency of the search process.

Although the techniques presented here are applicable to a wide variety of objects, and the relationships between them, the current implementation models only cubes and planes, and the fact that cubes generally sit with one face in contact with a plane. This is a limitation of the current implementation rather than the method.

The rest of this paper is organised as follows: Section 2 describes the structure from motion process used to recover the point cloud from the image set. Section 3 explains the form of the models use while Sections 4 and 5 describe the cube and plane model examples respectively. The relationships between model instances and the mechanisms used to enforce them are described in Section 6. The model fitting process is explained in Sections 7 and 8, and the results of testing on real images are given in Section 9.

2 Structure-From-Motion Algorithm

The first step of the model fitting process is to automatically calibrate the camera used to record the input images and to recover a point-based reconstruction of the objects in the scene. Using homogeneous coordinates, a 3D object point is represented as $\mathbf{P} = (P_1, P_2, P_3, 1)^\top$ and a 2D image feature point as $\mathbf{p} = (p_1, p_2, 1)^\top$. The feature point $\mathbf{p}_{i,k}$ is the projection of a 3D object point $\mathbf{P}_i$ in the k-th camera image, with $\mathbf{p}_{i,k} \sim \boldsymbol{A}_k \mathbf{P}_i$, where $\boldsymbol{A}_k$ is the 3×4 camera matrix of the k-th camera image and $\mathbf{a} \sim \mathbf{b}$ indicates that the vectors $\mathbf{a}$ and $\mathbf{b}$ are equal up to a scale factor. For each image of the sequence, the camera matrix $\boldsymbol{A}_k$ contains the camera's intrinsic parameters such as focal length and principal point, as well as its extrinsic position and orientation parameters.

The process used for estimation of $\boldsymbol{A}_k$ and $\mathbf{P}_i$ is consistent with modern structure from motion techniques. The details are available in [15]. The process produces estimates of the pose and internal parameters of each camera, and of the 3D position of a set of feature points in the scene.

3 Model Specification

Before describing the modelling algorithm itself, we define what a model is and then illustrate our definition with an example. Each model is defined by a identifying label (for example, it might be a cube or a sphere). By a slight abuse of notation we identify with each instance of a model M a vector of parameters sharing the same label. The form of this parameter vector depends on the model type. In the case of a cube, for example, the parameters define the cube's position, scale, and orientation in 3D space. We aim to find parameter values that are most probable given the data D (images and 3D points) and any prior information I (on which more later). Thus we aim to maximise

$$\Pr(M|DI) \, \alpha \Pr(D|MI) \Pr(M|I) \ . \tag{1}$$

We can partition the data into 2D and 3D feature sets D_2 and D_3, and despite the fact that they have the same source (the original image set) the relationship between the two is distant enough to justify assuming that $\Pr(D_3|MI)$ and $\Pr(D_2|MI)$ are independent. We thus see that

$$\Pr(D|MI) = \Pr(D_3|MI) \Pr(D_2|MI) \tag{2}$$

As part of the model definition we define both of these likelihood functions, and the prior distribution $\Pr(I)$. The 3D likelihood function defines the probability of a set of model parameters given a set of 3D points, and typically favours parameters that result in many 3D points lying close to or on the model surface.

The 2D likelihood function defines the probability of model parameters given the images—this typically favours image edges near the projections of model edges, and incorporates any appearance information that is known about the model. We give examples of such functions in the following sections.

Because the definition of a model is quite general, the method is naturally capable of fitting a range of models, and also of fitting families of models. A simple model, for example, might be a plane or sphere. More complex models might involve objects with non-parametric descriptors, or families of objects and the relationships between them.

4 The Cube Model

In order to explain the modelling and localisation process we first consider a simple cube model. The cube model has a vector of 7 parameters $\mathbf{C}$ describing its 3 dimensional shape: the position of its centroid $\mathbf{T} = [T_x, T_y, T_z]$, its orientation $\boldsymbol{R}$, corresponding to Euler angles $[R_x, R_y, R_z]$, and its side length S.

4.1 The 3-Dimensional Likelihood

We first consider a definition for $\Pr(D_3|MI)$, the likelihood of a 3D point cloud given a particular cube model. The likelihood of each point given a cube model is closely related to the distance from that point to the cube. For a point in the interior of the cube, the distance to the cube is simply the perpendicular distance to the closest face of the cube. For a point outside the cube, the distance may be the diagonal distance to a corner of the cube. However, it is also well approximated as the perpendicular distance to the plane containing the nearest face of the cube. The problem therefore becomes one of determining the closest face to a point.

To solve this problem, we transform the point into a coordinate system aligned with the cube: $\mathbf{P}_C = \boldsymbol{R}\mathbf{P} + \mathbf{T}$ where $\boldsymbol{R}$ and $\mathbf{T}$ are the cube orientation matrix and centroid position respectively. Now if we divide this space according to which face of the cube is closest, we end up with 6 regions that are bounded by rays emanating from the centre of the cube and passing through each vertex of the cube. Testing which of these regions contains the point is very simple: if $\mathbf{P}_C = [X_C, Y_C, Z_C]$, one region is exactly the set of points where X_C is greater than Y_C, Z_C, $-X_C$, $-Y_C$ and $-Z_C$. Similarly the other regions are composed of points where Y_C is the maximum value, where Z_C is the maximum, and where $-X_C$, $-Y_C$ and $-Z_C$ are the maximum. Once $D_C = \max(X_C, Y_C, Z_C, -X_C, -Y_C, -Z_C)$ has been found, the distance from the point $\mathbf{P}_i$ to the cube is simply $d_c(\mathbf{P}_i, \mathbf{C}) = D_C - S/2$, S being the cube side length. This distance can be squared so that points inside and outside the cube are treated symmetrically.

Of course not all 3D points belong to the model that is being fitted. To achieve robustness to points that are not part of the model, a Huber function [16] is applied to the distance measure: $d(\mathbf{P}_i, \mathbf{C}) = \Psi(d_c(\mathbf{P}_i, \mathbf{C}))$. This also has the effect of segmenting the point cloud into those points belonging, and not belonging, to the cube according to their position.

Having constructed a distance measure between a point and a cube, we are now in a position to define a function for the likelihood of a set of 3D points given a cube model. Assuming that all 3D points are mutually independent, the likelihood can be written as $\Pr(D_3|MI) = \prod_i \Pr(\mathbf{P}_i|MI)$ where $D_3 = \{\mathbf{P}_i\}$ is the set of reconstructed 3D points. Assuming a Gaussian error model, the negative log likelihood for a set of points $\{\mathbf{P}_i\}$ where $i = 1 \ldots n$ given a cube $\mathbf{C}$ is

$$\mathcal{J}_3(\{\mathbf{P}_i\}, \mathbf{C}) = f_3 \sum_i d(\mathbf{P}_i, \mathbf{C})^2 \tag{3}$$

where f_3 is a constant scale factor.

4.2 The 2-Dimensional Likelihood

The 2-dimensional aspect of the model is based on the assumption that edges in the model will give rise to intensity gradients in the image. Edges have a number of advantages over corners or other features that might be used to guide model fitting. These advantages have been well discussed in the tracking literature (see [17, 18] for example) but include rapid detection and relative robustness to changes in lighting.

In order to calculate the degree to which a hypothesised cube model is supported by the image intensities the visible edges are projected back into the image set and a measure is taken of the corresponding intensity gradients. The measure is similar to that used in [17] amongst others. A series of normals to the edge are taken at points $\{E_\mu\}$ spaced equally along the original model edge. Points at the ends of the edge are discarded. The points $\{E_\mu\}$ are pre-computed and scaled according to the size of the cube model and then projected into the image to give $\{e_\mu\}$, $\mu = 1 \ldots m$. This process avoids the bias towards edges close to the camera that might occur if calculations were made in image space. At each point e_μ the normal to the reprojected model edge is taken and image gradients calculated along these lines. The points at which the gradients are calculated for a particular e_μ are labeled $n_{\mu\nu}$ where $\nu = -g \ldots g$ and $e_\mu = n_{\mu 0}$. The $2g+1$ gradients $\{n_{\mu\nu}\}$ are evenly spaced along the normal with the distance between them scaled by the distance between e_μ and its closest neighbour. Again, the the aim is to avoid bias in the calculations. The gradients $\{I'(n_{\mu\nu})\}$, $\nu = -g+1 \ldots g-1$ are calculated using a central difference formula, on the basis of the image intensities $\{I(n_{\mu\nu})\}$. The gradients are then weighted by $\exp(-\nu^2/s)$ where s is selected according to the scale of the edge expected. This scaling weights closer edges higher than those further away. The maximum of the weighted $I'(n_{\mu\nu})$ is then determined and the corresponding $|\nu|$ taken as the distance between the model and image edges for e_μ. The sum of these distances is used as the measure of fit. Assuming a Gaussian error model, the negative log likelihood function is thus

$$\mathcal{J}_2(\{I_k\}, \mathbf{C}) = f_2 \sum_k \sum_\mu \arg\max_\nu \exp(-\frac{\nu^2}{s}) I'_k(n_{\mu\nu})^2 \tag{4}$$

where f_2 is a constant scale factor.

5 The Plane Model

To demonstrate the generality of our model definition process, we now consider the model for a plane. This model has 4 parameters describing its normal and position: $\mathbf{N} = [N_1, N_2, N_3, N_4]$. These parameters are defined such that for any point $\mathbf{P}_i$ lying on the plane, $d_p(\mathbf{P}_i, \mathbf{N}) = \mathbf{N} \cdot [\mathbf{P}_i, 1] = 0$. For points not lying on the plane, d_p is the perpendicular distance of the point to the plane. Once more assuming the independence of 3D data points, and a Gaussian model relating distance to likelihood, the negative log likelihood of the 3D data can be written as

$$\mathcal{J}_3(\{\mathbf{P}_i\}, \mathbf{N}) = f_{3p} \sum_i d_p(\mathbf{P}_i, \mathbf{N})^2 \tag{5}$$

The 2-dimensional likelihood of a plane is based on the assumption that the surface of the plane is largely unoccluded by objects not modelled and that it is a Lambertian surface and will therefore have the same appearance in each image. The projections of a point on the surface into each image are related by homographies, which can be calculated analytically from the camera projection matrices and the plane parameters (for example, see [19]). The likelihood of each point on the surface of a hypothesised plane model is therefore related to the difference in pixel values at the projection of that point into each image in which it is visible. Initially, we take as the cost function the variance of the pixel values. For most points on the plane, the variance is expected to be very similar and close to 0. We therefore model the distribution of pixel variances over the plane as a Laplacian. In this case the negative log likelihood is written as

$$\mathcal{J}_2(\{I_k\}, \mathbf{N}) = f_{2p} \sum_{\mathbf{Q}} |\mathrm{var}\{I_k(\boldsymbol{A}_k \mathbf{Q})\}| \tag{6}$$

where $\mathbf{Q}$ iterates over points on the plane, and k iterates over cameras.

6 Relationships Between Models

Relationships between models are defined in terms of their parameters. A relationship is formed between 2 models when there is a dependency between one or more of their parameters. For example, if a cube is resting on a table, there is a dependency between the position of the cube and the table, and therefore they are related. If a cube is placed on top of another cube, which in turn is resting on a table, then all 3 are related. If 2 cubes are resting on different areas of the same table, then all 3 objects are related through the table.

In general, these dependencies can be represented by a graph structure, and we intend to construct and perform inference on such a structure as future work. As a preliminary step we define a number of common relationships that can occur between pairs of objects, and then look for those. The example of a box sitting on a table is well modelled by a single cube resting on a single plane. The relationship in this case is that one face of the cube that is not visible is conicident with a plane, representing the table. To test for this relationship, having detected a cube, we search for a plane using the likelihood measures defined in Section 5 in the neighbourhood of, and aligned with, each non-visible face of the cube.

7 Model Fitting

Having defined cube and plane models, and their associated likelihood functions, we now describe an algorithm for fitting such models to 3D and image data. It is not feasible to generate and evaluate a set of samples that would effectively explore $\Pr(D|MI)$. Instead we use a coarse-to-fine strategy which exploits the nature of the functions $\Pr(D_3|MI)$ and $\Pr(D_2|MI)$ in order to guide our search for a suitable model. The function $\Pr(D_3|MI)$ relates the model to a set of reconstructed points and is well suited to gross localisation of the object in the scene, due to the relatively smooth nature of the associated probability distribution. The function $\Pr(D_2|MI)$ relates the model to the appearance of the object in the image set, and is typically only applicable when the model is very close to the true location of the object. When this criterion is satisfied, however, it can achieve very precise localisation, as the associated probability distribution is typically strongly peaked. Thus the 3D likelihood function is better suited to initial localisation, while the 2D likelihood is appropriate for further optimisation based on this initial estimate.

The rest of this section deals with fitting a cube model; however, a similar set of steps is required to fit any other model. In fact composite models containing a cube, such as the 'cube on plane' example, are fitted in the same way, as the finding the location of the cube first provides more information than locating the plane.

7.1 Initialisation

The first stage of the method requires that the user identify a point on the object to be modeled in any of the input images. It is not necessary to identify a corner or an edge; rather, the point merely identifies a ray through the reconstruction which intersects the object of interest. A single ray through the scene provides the maximum information from minimal user interaction.

A series of simple templates is spaced along this ray. The idea of the template $T(\{\mathbf{P}_i\}, C, r)$ is to calculate the likelihood of a set of points $\{\mathbf{P}_i\}$, $i = 1 \ldots n$, given a cube with centre C and radius r, integrated over all cube orientations. The radius is taken to be half of the side length of the cube. This function can be evaluated in the form $T(\{\mathbf{P}_i\}, C, r) = \sum_i w(d(\mathbf{P}_i, C), r)$, where $w()$ is a density function representing the distance from the centre of a cube to a set of points equally spaced over the surface of a cube. By generating such a point set for a cube of radius 1, it was determined that this function can be closely approximated by a Gaussian distribution with mean 1.28 and standard deviation 0.162. Given that the ray identified by the user does not necessarily intersect the centre of the cube we require a Gaussian smoothed version of this density.It has been determined empirically that a value of 0.3 provides acceptable results for a cube of radius 1.

Multiplying the function $w(d(\mathbf{P}_i, C), r)$ by a scale factor common to all points has no effect on the ordering of the results, so the absolute scale of the function may be ignored. We thus let $w(d(\mathbf{P}_i, C), r) = \exp(-\left(d(\mathbf{P}_i, C)/r - 1.28\right)^2/0.18)$.

It is assumed that the object sought will fill at least 1% and less than 100% of the image used to identify it. This forms part of the cube model prior $\Pr(I)$, and provides a constraint upon the range of template radii that should be searched at each point along the intersecting ray. A set of hypothetical object sizes is chosen within this range and for each size a set of points along the ray are selected to form the template centres for that object size. The radius of each template is calculated independently for each point along the ray and increases with the distance from the camera. The distance between template centres increases with the calculated radius, and thus also with the distance to the camera. The function $T(\{\mathbf{P}_i\}, C, r)$ is evaluated for each template and the parameter vectors corresponding to function evaluations above the 90th percentile retained. These parameter vectors are used to initialise the optimisation process.

7.2 Optimisation

Each initial parameter vector specifies the centre and radius of a hypothesised cube. This information is used to initialise an iterative search process based upon the likelihood function $\mathcal{J}_3(\{\mathbf{P}_i\}, \mathbf{C})$ specified in equation (3). In order to evaluate this function an orientation is required for the cube hypothesis. This orientation is initialised to be aligned with the camera coordinate system. A Levenberg Marquardt minimisation process is carried out on the cost function $\mathcal{J}_3(\{\mathbf{P}_i\}, \mathbf{C})$. The result of this minimisation is a parameter vector describing the location, radius, and orientation of a cube hypothesis. One such parameter vector is recovered for each initialisation. These vectors are checked to ensure that they are significantly different from each other and that they intersect the ray specified by the user. The remaining parameter vectors may be interpreted as the identifying the local modes of the probability density function associated with $\Pr(D_3|MI)\Pr(I)$.

Having explored $\Pr(D_3|MI)$ we now incorporate $\Pr(D_2|MI)$ in order to find the modes of $\Pr(D|MI)$. The 2-dimensional data likelihood of the model is described in Section 4, and gives rise to the cost function $\mathcal{J}_2(\{I_k\}, \mathbf{C})$ (Equation (4)). Recall that this cost function is based on the image distance between the projected edge of the model and the local intensity gradient maximum normal to the edge, summed across multiple points along each edge.

The 2D and 3D likelihood functions can now be combined to generate a complete data likelihood function. Because they are both log likelihoods, they are combined by addition; however because they are not normalised a scale factor is required to ensure that they each contribute appropriately to the final likelihood. Because the 2D data likelihood is more sensitive to small changes in the cube parameters, it tends to dominate this final optimisation stage, which is appropriate as the image data is more specific than the 3D data.

7.3 Selecting Amongst Hypotheses

The result of the optimisation process is a set of possible object models. Each model in the set is compared against every other to eliminate those similar enough to be labeled duplicates. The remaining set is passed on to the user for examination. This set is expected to contain a small number of possible objects (often only one), one of which will be very close to that required.

In order to facilitate user interrogation of the set of returned hypotheses, each is rendered onto the image set. The user may then select between the various individual object hypotheses using the mouse wheel. The selected object hypothesis is highlighted by being rendered in a different colour. One of the hypothesised object models is typically a very good fit to the image data. There are circumstances, however, where the accuracy achieved by this method may not be suitable for the desired application. It is thus important, if the method is to be of practical value, to provide a means of refining the fit to the point whereby the desired level of accuracy may be guaranteed.

8 Refining Models by Further Interaction

The second stage of the fitting process provides a method whereby further user interaction may be used to further inform the fitting process. As with the earlier stages, the goal is to minimise the tedium associated with this process by extracting the maximum value from each interaction. This requires that automation decrease intelligently as the level of user interaction increases. Eventually the level of automation decreases to the point that the process involved in object specification becomes similar to the image modeling interfaces of systems such as PFTrack by The Pixel Farm or ICARUS [20].

After the best fitting object model is selected by the process described in Section 7.3 it is rendered onto the image set. The user then selects an image from the set in which to carry out the object manipulation functions—typically the one in which the deficiencies of the model are most obvious. Editing is performed by moving a vertex of the model so as to best align its projection with the image. This drag-and-drop operation is performed on as many vertices as necessary to achieve the desired accuracy of fit.

Let the position of the selected 3-dimensional object vertex be $\mathbf{V}_\gamma(\mathbf{C})$, and its desired position in image k be $\mathbf{a}_{\gamma,k}$. We now wish to calculate the transformation that would need to be made to the model parameters $\mathbf{C}$ in order to align the projection of $\mathbf{V}_\gamma(\mathbf{C})$ into image k with $\mathbf{a}_{\gamma,k}$.

If the camera matrix for image k is $\boldsymbol{A}_k$ then we label the distance between the desired vertex location $\mathbf{a}_{\gamma,k}$ and the model vertex projection $\mathbf{V}_\gamma(\mathbf{C})$ as $d_U(\mathbf{a}_{\gamma,k}, \boldsymbol{A}_k\mathbf{V}_\gamma(\mathbf{C}))$. Again assuming a Gaussian error model the negative log likelihood of object $\mathbf{C}(\mathbf{C})$ described by the parameter vector $\mathbf{C}$ for a set of user specified vertex projection points $\mathsf{a} = \{\mathbf{a}_{\gamma,\,k}\}$ is

$$\mathcal{J}_U(\mathsf{a}, \mathbf{C}) = f_U \sum_{\mathbf{a}_{\gamma,\,k} \in \mathsf{a}} d_U(\mathbf{a}_{\gamma,\,k}, \boldsymbol{A}_k \mathbf{V}_\gamma(\mathbf{C}))^2. \tag{7}$$

The fact that this cost function takes the form of a negative log-likelihood means that it may be combined with $\mathcal{J}_3(\{\mathbf{P}_i\}, \mathbf{C})$ and $\mathcal{J}_2(\{I_k\}, \mathbf{C})$ in order to derive a combined data likelihood equation. This requires that appropriate values be selected for the various scale factors, which is done under the assumption that data provided by the user is unlikely to be in error over all but the smallest of distances. The final model estimate is calculated by minimising this joint cost function.

Numerical minimisation of this complex cost function is unfortunately too slow a process to be carried out while the user drags a vertex across an image. Thus, in order to provide the user with an impression of the effect of manipulating the vertices of the object, the projection of the model into the current image is updated using a simplified version of this full likelihood. If the user is editing the location of vertex $\mathbf{V}_\gamma$ of an object $\mathbf{C}(\mathbf{C}_O)$ in image k then the parameters of the object model projected into the image are determined by

$$\mathbf{C}'_N = \arg\min_{\mathbf{C}_N} \mathcal{J}_U(\{\mathbf{a}_{\gamma,\,k}\}, \mathbf{C}(\mathbf{C}_N)) + \|\mathbf{C}_N - \mathbf{C}_O\|. \tag{8}$$

The parameter vector $\mathbf{C}'_N$ thus represents the model containing the newly specified vertex location which is closest to the best existing model estimate. Equation (8) may be evaluated quickly using numerical minimisation.

9 Results

Figure 1 illustrates the sequence on which we test the scene modelling software. This scene contains a number of textured cubes resting on a tabletop. It is therefore well suited for the use of our cube and plane models, but also presents some difficulties. Some of the cubes are quite close to each other, for instance, so as the camera moves, the cubes occlude each other significantly. Additionally, the patterns on some of the surfaces are of greater contrast than, but aligned with the cube edges. These patterns thus provide strong, plausible, but false, edge cues for the 2D matching process. The fact that the method combines likelihoods allows it to overcome these obstacles. An auxiliary advantage of this sequence is that ground truth is available, so we can quantitatively test the accuracy of our algorithm.

The structure from motion process, for instance, recovered the camera rotation to within a degree about each axis, and the camera position is almost always recovered to within 2mm of the ground truth value. Approximately 4000 3D points are generated, forming our 3D data set.

Figure 2 shows the result of the first stage of the model fitting process. To get to this stage, the user has clicked once on the cube in one image. The system

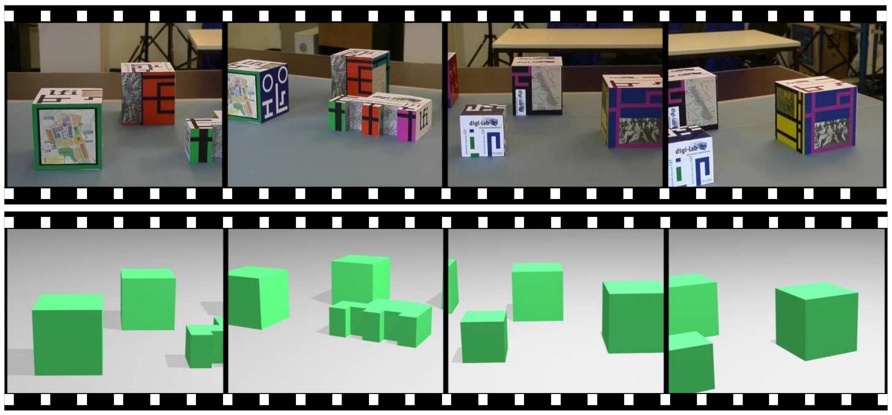

Fig. 1. Frames from the test image sequence and the corresponding ground truth

has searched for cubes in the 3D data, within a region defined by the location of the mouse click, as described in section 7.1. This figure shows the projection of the edges of candidate models considered most probable.

Finally, after further optimisation, only one hypothesised cube remains. It is therefore automatically selected as shown in Figure 3, where it does appear to correspond to the cube in the scene. This entire process happens in under a second and is therefore extremely responsive. The projection of the cube into the image looks visually correct, but to test it quantitatively we compare our result with ground truth data as shown in Table 1. The 3 cubes shown in this table correspond to 3 of the cubes visible in the scene. For each cube, we show the ground truth parameters, the parameters estimated after a single mouse click on the cube in one image, and, in the case of cube 3, the parameters estimated after several clicks on the cube in one image.

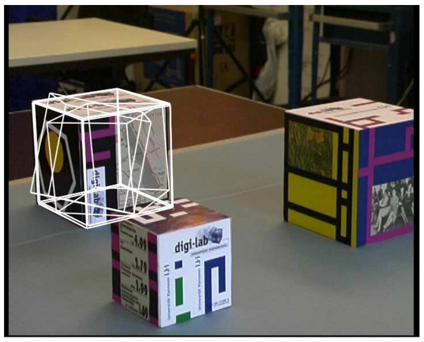

Fig. 2. Initial set of cube hypotheses overlaid on input image

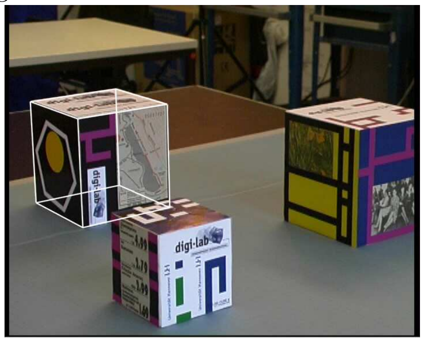

Fig. 3. Final cube estimate, after optimisation.

The estimated cube model allows the plane hypotheses to be evaluated. A plane model is fit to each surface of the cube which is not visible in the image

Table 1. Comparison of estimated cube parameters with ground truth.

	T_x [mm]	T_y[mm]	T_z [mm]	$\boldsymbol{R}_x$ [deg]	$\boldsymbol{R}_y$ [deg]	$\boldsymbol{R}_z$ [deg]	S [mm]
cube1 truth	87.5	7.5	7.5	0.0	0.0	0.0	15.0
cube1 1 click	87.792	8.0074	7.3857	-2.3438	1.5686	2.3601	14.9653
cube2 truth	62.5	37.5	7.5	0.0	0.0	0.0	15.0
cube2 1 click	62.393	37.325	7.6082	1.2733	1.5659	-0.3057	14.9901
cube3 truth	-27.5	3.5	7.5	0.0	0.0	0.0	15.0
cube3 1 click	-28.5762	3.9391	6.9696	-1.5475	0.0291	3.0684	15.8817
cube3 multi click	-27.5354	3.7819	7.3968	-1.5639	-0.0005	3.1364	14.9259

set. Each model is evaluated by projecting the original image set into a reference frame in which the surface of the cube being evaluated sets the coordinate frame. This process is described in Section 5. The images generated for each plane hypothesis are shown in Figure 4. The leftmost image corresponds to the

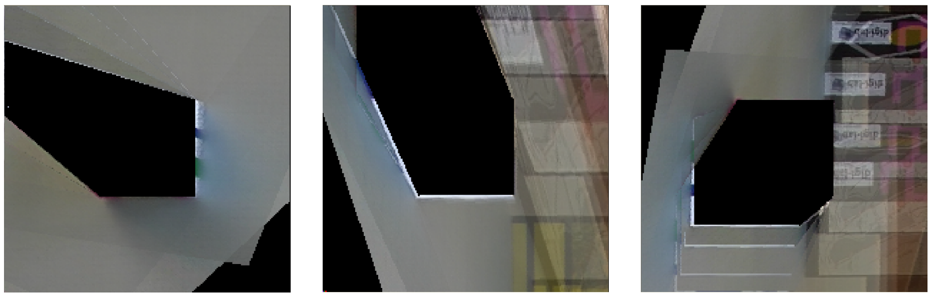

Fig. 4. The reprojection of the image set corresponding to 3 plane hypotheses with visibility masked areas shown in black. The leftmost representing the correct hypothesis

hypothesis which best matches the original scene. This is clearly visible in the homogeneity of the reprojected textures, the images corresponding to the two remaining hypotheses showing far more variation in colour. Figure 5 shows three histograms of the standard deviations of pixel intensity, one corresponding to each hypothesis. Each histogram shows the distribution of standard deviations for each image, the histogram corresponding to the correct hypothesis showing a single peak with a small number of outliers. This is as compared to the histograms associated with the incorrect hypotheses which show a peak associated with the pixels on the table, but also a significant volume of points outside this peak. The medians of the three sets of standard deviations (corresponding to maximum likelihood estimates under the Laplacian distribution model) are 0.45, 1.29 and 0.73 respectively, the lowest correctly identifying the required plane hypothesis.

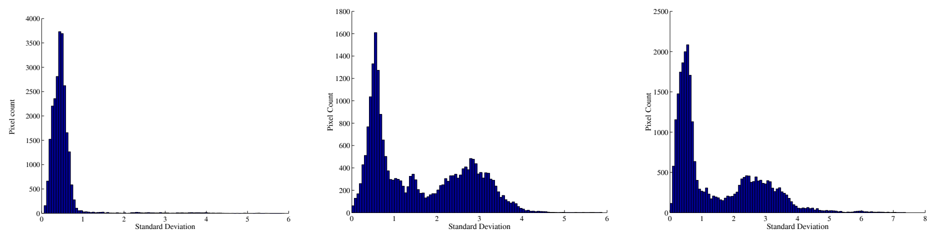

Fig. 5. The histograms of standard deviations for each plane hypothesis. Each graph plots pixel count per histogram bin against (maximum) standard deviation per bin.

As an indication of the advantages of the model-based scene reconstructions generated by the method presented here Figure 6 shows the original scene but with the front cube rotated so as to stand on one of its vertices.

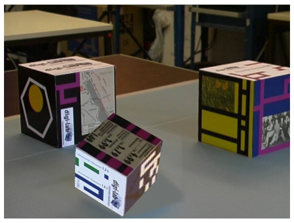

Fig. 6. A synthetic image generated on the basis of a recovered scene model

10 Conclusion

A method has been described for fitting multiple interacting models to image data on the basis of both 2D and 3D criteria. These criteria have been derived so as to minimise the interaction necessary with the fitting process by the appropriate division of prior information between the user and the system. This represents an important step towards the final goal of generating a full semantic model of a scene on the basis of imagery. A number of further research challenges exist, including the development of further classes of object models and the broadening of the relationships that might exist between them. The final goal of this work is a system to generate a semantic interpretation of an entire scene with minimal user input. However, much work remains to be done before this may be achieved.

References

1. Waltz, D.: Understanding line-drawings of scenes with shadows. artificial intelligence. Artificial Intelligence **2** (1971) 79–116

2. Taylor, C., Debevec, P., Malik, J.: Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. ACM SIGGraph, Computer Graphics (1996) 11–20
3. Eos Systems: Photomodeler: A commercial photogrammetry product http://www.photomodeler.com (2005)
4. Robertson, D., Cipolla, R.: An interactive system for constraint-based modelling. In: Proc. 11th British Machine Vision Conference. (2000) 536–545
5. Sturm, P., Maybank, S.: A method for interactive 3d reconstruction of piercewise planar objects from single images. In: Proc. 10th British Machine Vision Conference. (1999) Single View Techniques
6. Shum, H., Han, M., Szeliski, R.: Interactive construction of 3d models from panoramic mosaics. In: Proc. IEEE Computer Vision and Pattern Recognition. (1998) 427–433
7. Criminisi, A., Reid, I., Zisserman, A.: Single view metrology. International Journal of Computer Vision **40** (2000) 123–148
8. Wilczkowiak, M., Sturm, P., Boyer, E.: Using geometric constraints through parallelepipeds for calibration and 3d modeling. IEEE Trans. Pattern Analysis and Machine Intelligence **27** (2005) 194–207
9. Dick, A., Torr, P., Cipolla, R.: Modelling and interpretation of architecture from several images. International Journal of Computer Vision **60** (2004) 111–134
10. Drummond, T., Cipolla, R.: Application of lie algebras to visual servoing. International Journal of Computer Vision **37** (2000) 21–41
11. Ferryman, J.M., Worrall, A.D., Sullivan, G.D., Baker, K.D.: A generic deformable model for vehicle recognition. In: Proceedings British Machine Vision Conference. (1995) 127–136
12. Bastian, J., van den Hengel, A.: Computing surface-based photo-consistency on graphics hardware. In: Proceedings of Digital Image Computing: Techniques and Applications,Cairns, Australia. (2005) 249–258
13. Arun, K., Huang, T., Blostein, S.: Least-squares fitting of two 3-d point sets. IEEE Trans. Pattern Analysis and Machine Intelligence **9** (1987) 698–700
14. Fisher, R., Fitzgibbon, A., Waite, M., Trucco, E., Orr, M.: Recognition of complex 3-d objects from range data. In: Proc. 7th International Conference on Image Analysis and Processing. (1993) 509–606
15. Thormählen, T.: Zuverlässige Schätzung der Kamerabewegung aus einer Bildfolge. PhD thesis, University of Hannover, Fortschritt-Berichte, Reihe 10, VDI Verlag (2006)
16. Huber, P.: Robust estimation of a location parameter. Annals of Mathematical Statistics **35** (1964) 73–101
17. Smith, P., Drummond, T., Cipolla, R.: Motion segmentation by tracking edge information over multiple frames. In: ECCV (2). (2000) 396–410
18. Daucher, N., Dhome, M., Lapreste, J.T., Rives, G.: Modelled object pose estimation and tracking by monocular vision. In: BMVC93. (1993) 249–258
19. Baker, S., Szeliski, R., Anandan, P.: A layered approach to stereo reconstruction. In: Proc. IEEE Computer Vision and Pattern Recognition. (1998) 434–441
20. Gibson, S., Hubbold, R., Cook, J., Howard, T.: Interactive reconstruction of virtual environments from video sequences. Computers & Graphics **27** (2003) 293–301