2002

Surface Reconstruction from Multiple Views using Rational B-Splines and Knot Insertion

BU CS TR 2002-010, to appear in Proc. Conf. on 3D Data Proc., Visualization and Trans., June 2002.

Surface Reconstruction from Multiple Views using Rational B-Splines and Knot Insertion

Matheen Siddiqui
Boston University
Image and Video Computing Group
Boston, MA, USA
siddiqui@cs.bu.edu

Stan Sclaroff
Boston University
Image and Video Computing Group
Boston, MA, USA
sclaroff@cs.bu.edu

Abstract

A method for reconstruction of 3D rational B-spline surfaces from multiple views is proposed. Given corresponding features in multiple views, though not necessarily visible in all views, the surface is reconstructed. First 2D B-spline patches are fitted to each view. The 3D B-splines and projection matricies can then be extracted from the 2D B-splines using factorization methods. The surface fit is then further refined via an iterative procedure. Finally, a hierarchal fitting scheme is proposed to allow modeling of complex surfaces by means of knot insertion. Experiments with real imagery demonstrate the efficacy of the approach.

1 Introduction and Related Work

Recovering models from multiple views is an area of great interest. The results of algorithms for recovering 3D structure and camera motion have many practical applications, such as computer aided design, virtual reality, movie special effects, video coding, etc. Models provide a concise representation for a large class of objects, and can reduce the dimensionality of a reconstruction problem to a few highly meaningful parameters. It thus is reasonable to incorporate model priors directly into reconstruction methods and expect better performance and richer representations.

Structure from motion algorithms make various use of prior information. In the weakest case they assume no priors and are limited to recovering point locations. While useful and robust in many cases, in practice, scenes typically contain structures with strong geometric regularities that can be used to constrain the estimation problem. For example indoor and man made scenes typically contain many planar structures and lines. As a consequence, many works have provided a thorough treatment of these kinds of objects. In particular [16, 19] have found closed form solutions using planar priors.

More general representations have been considered in a bottom-up framework, where many small planar regions are fitted to an object. In particular, planar meshes were explored by [6]. Surfaces have also been modeled as oriented particles or tiny planes with associated texture in [5, 8, 15]. While general, in practice many facets are needed for these methods to successfully approximate a surface.

In many cases objects intrinsically do not have such high complexity and more appropriate classes of models are sought. In [3] and [10], methods have been proposed that can be used when objects are well-approximated by quadric surfaces. In [3], the quadric surface is estimated by relating its silhouette to the projected image. In a different approach, [10] examines the induced flow field of quadric objects.

In this work we further develop the treatment of model and surfaces priors in structure from motion. This general idea has been considered in [20], where bundle adjustment was augmented with a parameterized model. In our work we consider models that are specified as weighted averages of basis points, as in the case of splines. This class of surfaces is ideal for SFM

as they are similar to regular scene points, which are weighted combinations of basis points [7]. The difference is that the blending functions for scene points are linear, while the blending functions for the surfaces we consider are non-linear.

While the principles developed in this work apply for any surface that can be represented as a weighted combination of basis points, such as splines and radial basis surfaces, we make explicit use of rational B-splines to represent 3D surfaces and their projections. Spline curve representations have been used extensively in computer vision in the context of contour tracking and contour pose recovery (e.g., [2, 1]). Splines have also been used in motion estimation [14] and image registration [13].

By directly modeling surfaces in the scene as splines, this work has the advantage over other methods in that points need not be visible in all views. More specifically, given features points in correspondence, but not necessarily visible in all views, we determine the 3D spline control points and camera matrices that explain the observed features. We also make use of the knot vectors that define the blending functions of the spline, to allow hierarchal fitting of surfaces.

The overall fitting procedure is shown in Figure 1. The approach we taken here is to first recover 2D representations of the spline surface as detailed in Section 2.1. From this a 3D surface and camera matricies are extracted and refined as described in Section 2.2 and 2.3, respectively. Additional degrees of freedom are then incorporated into the surface in a hierarchal way using as described in Section 3. The approach is demonstrated in a formulation for recovering a biquadratic rational B-spline surfaces from point correspondences given in multiple views. Experiments with real imagery demonstrate the efficacy of the approach.

2 Formulation

In this section we formulate the model reconstruction problem from multiple views. First consider general surfaces which take on the form:

$$\mathbf{P}(\mathbf{s}) = \mathcal{S}(\mathbf{s}, \mathbf{C}) \tag{1}$$

Here $\mathcal{S}$ represents a particular surface model, while $\mathbf{P}(\mathbf{s})$ denotes a 3D point on that model. Additionally, $\mathbf{C}$ specifies global shape parameters of the model, and

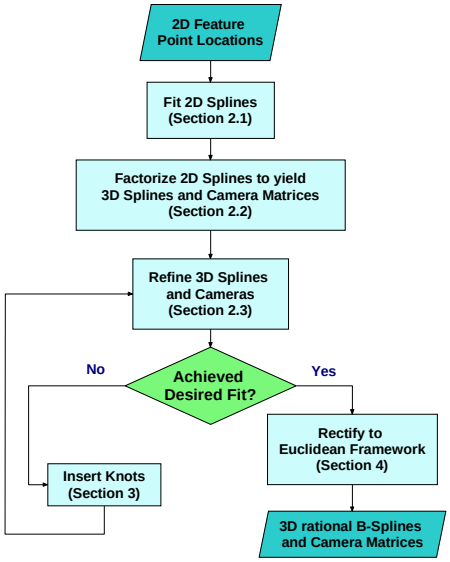


Figure 1. Flow chart of surface reconstruction algorithm

the parameter vector $\mathbf{s}$ specifies the location of a point on the model. The details of the underlying structure of $\mathcal{S}$ depends entirely on the particular model of interest. This representation is quite general and if a strong model is available can be quite useful in reconstruction. In [20], for example, such a model was designed for faces by an artist and was fit to observed image points along with other scene points through a bundle adjustment like optimization.

If such strong priors are not available, however, we need to consider representations that make more general assumptions on the form of the model. We accomplish this by considering models of the form:

$$\begin{bmatrix} \mathbf{P}(\mathbf{s}) \\ 1 \end{bmatrix} \sim \sum_{k=1}^{N_c} \beta_k(\mathbf{s}) \tilde{\mathbf{P}}_k \tag{2}$$

where,

$$\tilde{\mathbf{P}}_k = \begin{bmatrix} \tilde{x}_k \\ \tilde{y}_k \\ \tilde{z}_k \\ \tilde{w}_k \end{bmatrix} = \tilde{w}_k \begin{bmatrix} \mathbf{P}_k \\ 1 \end{bmatrix} \tag{3}$$

Where $\sim$ is defined to mean equality up to scale. Here points on the model are determined up to scale as a weighted combination of N_c points, $\tilde{\mathbf{P}}_k$ and the blending terms are determined by the point parameter $\mathbf{s}$ and

$\beta_k()$. In addition the basis point $\tilde{\mathbf{P}}_k$ can be decomposed into a 3D point $\mathbf{P}_k$ and a weight $\tilde{w}_k$ as shown. This form is important in the context of structure from motion as general scene points in a projective scene are expressed in a similar way. Additionally it is important in the context of surfaces, since rational B-splines are represented this way as well.

This becomes apparent if we remove the scale ambiguity of Equation (2) by dividing out the forth component, which yields:

$$\mathbf{P}(\mathbf{s}) = \frac{\sum_{k=1}^{N_c} \beta_k(\mathbf{s})\tilde{w}_k \mathbf{P}_k}{\sum_{k=1}^{N_c} \beta_k(\mathbf{s})\tilde{w}_k} \tag{4}$$

This is the form of rational splines, if the blending functions, $\beta_k()$, are selected appropriately.

Heretofore we have considered Equation (2) in the most general sense, but now we will limit the discussion to spline surfaces. In particular we will assume the blending functions are formed by the Cox de-Boor functions which are specified by an *order* and a *knot vector* in each parameter direction [4]. Since we are dealing with surfaces, the point parameter, $\mathbf{s}$ will be specified as a 2D coordinate (s,t) and the blending functions $\beta_k(\mathbf{s})$ will be written as $B_k(s,t)$. It is important to note that the following discussion is general in the sense that it can be applied to other types of blending function.

2.1 Reconstructing 2D Splines

In the reconstruction problem we consider feature points to be imaged samples of this spline model. Thus to recover the model we need to find the 3D control points, and $(s_i,t_i)\forall i \in [1,N]$, as well as the camera matrices, $\mathcal{P}^{(j)}$, responsible for the formation of the images. This can be posed as an optimization problem in which we minimize the difference between the observed locations of the features and the locations predicted by the model. To solve this optimization problem an initial estimate is needed. This can be found by first reconstructing the projections of the spline surface in each view.

Consider the image of a spline by the j^{th} camera. This is denoted by:

$$\begin{bmatrix} u_i^{(j)} \\ v_i^{(j)} \\ 1 \end{bmatrix} \sim \mathcal{P}^{(j)} \sum_{k=1}^{N_c} B_k(s_i,t_i)\tilde{\mathbf{P}}_k \tag{5}$$

Bringing the projection matrix into the summation:

$$\begin{bmatrix} u_i^{(j)} \\ v_i^{(j)} \\ 1 \end{bmatrix} \sim \sum_{k=1}^{N_c} B_k(s_i,t_i)\mathcal{P}^{(j)}\tilde{\mathbf{P}}_k \tag{6}$$

$$\sim \sum_{k=1}^{N_c} B_k(s_i,t_i) \begin{bmatrix} \tilde{u}_k^{(j)} \\ \tilde{v}_k^{(j)} \\ \tilde{w}_k^{(j)} \end{bmatrix} \tag{7}$$

From this we see that the image of a 3D rational B-spline surface is itself a rational B-spline surface, which we refer to as a 2D rational B-spline surface. Note that corresponding points viewed across images have the same (s,t) values. Also note that in the case of orthographic cameras, the 3^{rd} row of each camera matrix $P^{(j)}$ is $[0,0,0,1]$ and the weights w_k are the same for all images as well as for the 3D surface.

Given N features in M views, we try to recover these 2D rational splines in each view. To do so, we minimize the fitting error between the observations and the model given by:

$$\sum_{i=1}^{N}\sum_{j=1}^{M} m_i^{(j)} \| \begin{bmatrix} u_i^{(j)} \\ v_i^{(j)} \end{bmatrix} - \frac{\sum_{k=1}^{N_c} B_k(s_i,t_i) \begin{bmatrix} \tilde{u}_k^{(j)} \\ \tilde{v}_k^{(j)} \end{bmatrix}}{\sum_{k=1}^{N_c} B_k(s_i,t_i)\tilde{w}_k^{(j)}} \|^2 \tag{8}$$

where $m_i^{(j)}$ takes values 1 if the i^{th} point is visible in the j^{th} image, and 0 otherwise. This can be minimized with respect to (s_i,t_i), the weights, and control points by using standard non-linear minimization procedures. Instead of considering all the parameters at once, however, we first minimizing Equation (8) with respect to (s_i,t_i) for each point independently. Then Equation (8) is minimized with respect to $\tilde{u}_k^{(j)}\forall k$, and $\tilde{v}_k^{(j)}\forall k$ in each view, and finally with respect to $w_k^{(j)}\forall k$ in each view. This process is then iterated until convergence. Furthermore, due to the $m_i^{(j)}$ term, it is not required that each point be visible in all views.

To start this procedure initial values are needed. While the weights can start at $w_k^{(j)} = 1\ \forall k \forall j$, an initial value of (s_i,t_i) must be found heuristically. In our implementation feature locations, $(u_i^{(j)}, v_i^{(j)})$ in an approximately frontal view of the object are affinely warped to $(\hat{u}_i^{(j)}, \hat{v}_i^{(j)})$ such that , $\hat{u}_i^{(j)} \in [s_{min}, s_{max}]$ and $\hat{v}_i^{(j)} \in [t_{min}, t_{max}]$. These warped locations can

be used as initial (s_i, t_i) values as they reflect the relative location of the features in parameter space, and will be refined later. Given values for (s_i, t_i), and $w_k^{(j)}$, the other parameters, $u_k^{(j)}$ and $v_k^{(j)}$, can be updated via linear least squares. Following this, the iterative minimization procedure can proceed.

2.2 3D Projective Reconstruction of Surface

After the fitting procedure above converges, the 2D reconstruction can be upgraded to 3D. This is possible as the 2D control points are imaged in the same way as other points, and thus standard reconstruction methods can be used to recover both the 3D control points, $\tilde{\mathbf{P}}$ and camera matrices, $\mathcal{P}^{(j)}$. In the perspective case points are related to their projections via:

$$\begin{bmatrix} u_i^j \\ v_i^{(j)} \\ 1 \end{bmatrix} \sim \mathcal{P}^{(j)} \tilde{\mathbf{X}}_i \tag{9}$$

Here the the relation is defined only up to scale; however if we knew these scale factors, or projective depths, we could write:

$$\lambda_i^{(j)} \begin{bmatrix} u_i^j \\ v_i^{(j)} \\ 1 \end{bmatrix} = \mathcal{P}^{(j)} \tilde{\mathbf{X}}_i \tag{10}$$

Given the projective depths for each feature point, camera matricies and structure are related to a measurement matrix [12]:

$$\tilde{\mathcal{Y}} = \begin{bmatrix} \lambda_1^{(1)}\tilde{\mathbf{u}}_1^{(1)} & \dots & \lambda_N^{(1)}\tilde{\mathbf{u}}_N^{(1)} \\ \vdots & \ddots & \vdots \\ \lambda_1^{(M)}\tilde{\mathbf{u}}_1^{(M)} & \dots & \lambda_N^{(M)}\tilde{\mathbf{u}}_N^{(M)} \end{bmatrix} \tag{11}$$

$$= \begin{bmatrix} \mathcal{P}^{(1)} \\ \vdots \\ \mathcal{P}^{(M)} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{X}}_1 \dots \tilde{\mathbf{X}}_N \end{bmatrix} = \mathcal{P}\mathcal{X} \tag{12}$$

where $\tilde{\mathbf{u}}_i^{(j)} = \left[u_i^{(j)}, v_i^{(j)}, 1\right]^T$. This decomposition is not unique as one could post-multiply the camera matricies by a full rank matrix, $\mathbf{H}$ and pre-multiply the point matrix by $\mathbf{H}^{(-1)}$ without changing the measurement matrix.

Typically in perspective reconstructions of this form the homogeneous component of each point is not known and must also be recovered. In our case, we have recovered 2D rational splines for each view. Thus we effectively have recovered the projective depths. If we sample the 2D rational B-spline without dividing through by the homogeneous coordinate, we can write:

$$\lambda_i^{(j)} \begin{bmatrix} u_i^{(j)} \\ v_i^{(j)} \\ 1 \end{bmatrix} = \sum_{k=1}^{N_c} B_k(s_i, t_i) \tilde{\mathbf{P}}_k^{(j)} \tag{13}$$

Thus we can generate the measurement matrix by sampling the 2D rational splines as in Equation (13). Traditional factorization [12] requires all features to be visible in all views. By sampling the 2D rational splines, we can cope with missing features as full measurement matrices are generated. This also allows us to predict the location of partially visible features in all views.

Given the scaled measurement matrix, the camera matrices and 3D control points are then recovered as described in [12]. To do this, the SVD of the measurement matrix $\tilde{\mathcal{Y}}$ is found:

$$\tilde{\mathcal{Y}} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \tag{14}$$

where, $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and $\mathbf{\Lambda}$ is diagonal. In Equation (12), $\mathbf{M}$ is a product of a $3M \times 4$ matrix (the camera matrix) and a $4 \times N$ matrix (the point matrix). Thus ideally $\tilde{\mathcal{Y}}$ should be rank four, but due to measurement and estimation errors this may not be the case. Thus to extract the camera matrices and point locations from $\tilde{\mathcal{Y}}$, the matrix closest to $\tilde{\mathcal{Y}}$ is found. This is computed by truncating $\mathbf{\Lambda}$ to $\mathbf{\Lambda}_{4\times4}$ such that $\mathbf{\Lambda}_{4\times4}$ is a diagonal matrix that comprises the upper left 4×4 block of $\mathbf{\Lambda}$. The camera and point matrices can then be defined to be:

$$\mathcal{P} = \hat{\mathbf{U}}\mathbf{\Lambda}_{4\times4}^{1/2} \tag{15}$$

$$\mathcal{X} = \hat{\mathbf{V}}\mathbf{\Lambda}_{4\times4}^{1/2} \tag{16}$$

where, $\hat{\mathbf{U}}$ consist of the first four columns of $\mathbf{U}$ and $\hat{\mathbf{V}}$ is the first four columns of $\mathbf{V}$. This will give an estimate of the camera matrices, and the individual 3D points on the spline.

From these estimates, we can recover the control

points, $\mathbf{P}_k \forall k$, by minimizing:

$$\sum_{i=1}^{N}\sum_{j=1}^{M} m_i^{(j)} \| \lambda_i^{(j)} \begin{bmatrix} u_i^{(j)} \\ v_i^{(j)} \\ 1 \end{bmatrix} - \mathcal{P}^{(j)} \sum_{k=1}^{N_c} B_k(s_i, t_i) \tilde{\mathbf{P}}_k \|^2 \tag{17}$$

which is the algebraic residual error between the spline and feature points scaled by their projective depths. This can be solved for in closed from using linear least squares; however the solution will not minimize the residual fitting error and its thus necessary to refine the solution using non-linear methods.

2.3 Refining Projective Reconstruction

While the method in Section 2.2 will produce a 3D projective or affine reconstruction, it in general will not optimally fit the data. It is therefore necessary to refine the estimate by minimizing the residual fitting error with respect to the parameters of the surface. Again, the fit is based on the sum of squared differences and takes on the form:

$$\sum_{i=1}^{N}\sum_{j=1}^{M} m_i^{(j)} \| \begin{bmatrix} u_i^{(j)} \\ v_i^{(j)} \end{bmatrix} - \frac{\sum_{k=1}^{N_c} \mathcal{P}_{1,2}^{(j)} \tilde{\mathbf{P}}_k B_k(s_i, t_i)}{\sum_{k'=1}^{N_c} \mathcal{P}_3^{(j)} \tilde{\mathbf{P}}_k B_{k'}(s_i, t_i)} \|^2 \tag{18}$$

where $\mathcal{P}_{1,2}^{(j)}$ is the first two rows of the j^{th} camera matrix and $\mathcal{P}_3^{(j)}$ is the third row of the j^{th} camera matrix.

As before we do not consider all the parameters at once, rather (18) is minimized by iteratively solving for $\mathcal{P}^{(j)} \forall j$, then $(s_i, t_i) \forall i$, and finally $\tilde{\mathbf{P}}_k \forall k$, until a local minium is achieved. In this case, the cost function is non-linearly dependent on (s_i, t_i) and the control points. Levenberg-Marquardt is therefor employed in minimizing with respect to these parameters [9]. When minimizing with respect to the camera matrices however, the first two rows can be solved for linearly given the third row. Thus when updating the camera matrices we alternate between updating the first two rows using linear least squares, and the the third using Levenberg-Marquardt. In the case of orthographic cameras the complexity of this problem is further reduced since $w_k^{(j)}$ are the same $\forall j$ and $\mathcal{P}_3 = [0, 0, 0, 1]$.

3 Adaptive Subdivision

The method described in Section 2 can be used on spline surfaces with an arbitrary number of control points. To model complex surfaces, splines with more control are points needed. However as the number of control points increases, the size of the fitting problem also increases. Fitting complex spline surfaces with many control points can then become computationally demanding and increasingly ill-posed.

To overcome the difficulties of fitting complex surfaces, we make use of a hierarchal fitting scheme and the local control property of splines. This is accomplished by making explicit use of *knot vectors*. In presenting this method we first develop an intuitive understanding of the knot vectors and refer readers to [4] for a deeper review.

A knot vector is an array of numbers arranged in increasing order. The elements of this vector are called *knots*. The knot vector together with the spline *order*, specify completely the blending functions, $B_k(s, t)$ of the spline in Equation (7). One knot vector, τ_s,and order, k_s, is specified for the s direction, and another knot vector, τ_t, and order, k_t, for the t direction.

Intuitively, these knot vectors partition parameter space into regions. The surface is defined over the middle block of these regions. Each valid region corresponds to part of the spline surface. Each control point is associated with a block of these regions and only influences the part of the surface corresponding to these regions. In addition, the number of regions each control point influences is determined by the spline order.

Parts of parameter space that have more regions (or knots) correspond to parts of the surface with more associated control points and thus more flexibility. Since each control point only affects a small block of regions, this flexibility is local to that part of the surface. Thus one can alter these control points to change the spline's local shape without affecting its global shape (i.e, splines exhibit local control).

In addition, we can insert new knots into an existing surface without changing the surface. To do this, additional control points must be computed. This process is known as *knot insertion* [4].

3.1 Adaptive Fitting with Knots

In designing a hierarchal surface fitting procedure, we explicitly use knots and knot insertion methods. First a spline with a small number of control points is fitted to the observed features using the method of Section 2. Following this knots are inserted to bisect

the region that has the largest associated fitting error.

In particular, we assign each region a score equal to the sum of the distances between the model predicted and observed feature location of the features whose (s,t) are within it. A knot is then inserted into τ_s and τ_t so that the region with the largest score is split. This process is illustrated in Figure 2, where region R, is split into R_1, R_2, R_3, and R_4. This will create additional control points, while leaving the shape of surface unaltered. The surface is then further refined using the techniques of Section 2.3, but here we only update the control points that affect the newly created regions (R_1, R_2, R_3,and R_4). These control points will change the shape of surface corresponding to the new regions as well as some neighboring regions. Thus in minimizing the residual fitting error all features associated with parts of the surface that will be modified are considered. This process can then be repeated and additional regions can be split until a desired fit is reached. By splitting regions in this way we adaptively add more control points to parts of the spline surface that do not register well with the data. It is important to note that while the resulting surface may have many control points at each step we are only updating a small subset of them, thus allowing us to fit complex surfaces efficiently.

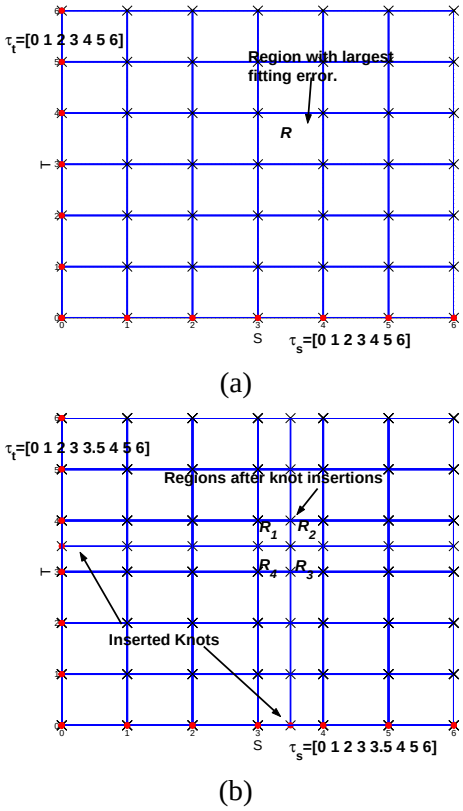


Figure 2. In (a) the region with the largest fitting error R is identified. In (b) knots are added to the knot vectors to split R into R_1, R_2, R_3, and R_4. In doing so new control points are generated.

4 Rectification

The above method succeeds in recovering a projective representation of the surface. In many applications such as feature tracking, augmented reality, and image based rendering this projective reconstruction is sufficient. To get a meaningful 3D reconstruction however, we must rectify the projective reconstruction. In particular we need to find an estimate of the rectifying homography, $\mathbf{H}$, that transforms reconstruction as:

$$\mathcal{P}^{(j)}\mathbf{H} \rightarrow \mathcal{P}_e^{(j)} \sim \mathbf{A}[\mathbf{R}^{(j)}|\mathbf{T}^{(j)}] \quad (19)$$

$$\mathbf{H}^{-1}\mathbf{X}_i \rightarrow \mathbf{X}_{e,i} \quad (20)$$

where $\mathbf{A}$ is an upper triangular matrix of the internal parameters, $\mathbf{R}^{(j)}$ is a rotation matrix, and $\mathbf{T}$ is a translation vector. The rectifying homography can be found if the ground truth for a sufficient number of scene points is known as the homography $\mathbf{H}$ can be computed directly. If this is not the case, the method of the absolute quadric can be employed [17] .

Rectification will yield a set of camera matrices, $\mathcal{P}_e^{(j)}$, and structure, $\mathbf{X}_e$ consistent with a Euclidean reconstruction. Following this, the Euclidean matrices, $\mathcal{P}_e^{(j)}$, can be decomposed into $\mathbf{R}^{(j)}$, $\mathbf{T}^{(j)}$, and $\mathbf{A}$. If we denote $\mathcal{P}_e^{(j)} = [\mathbf{P}_e^{(j)}|\mathbf{p}_e^{(j)}]$, this is accomplished by applying QR factorization on $\mathbf{P}_e^{(j)-1}$ to yield:

$$\mathbf{P}_e^{(j)-1} = \hat{\mathbf{R}}\hat{\mathbf{A}} \quad (21)$$

where $\hat{\mathbf{R}}$ and $\hat{\mathbf{A}}$ are the results of a QR decomposition. Estimates for the rotation matrix and internal parameters are found by:

$$\mathbf{P}_e^{(j)} \sim (\hat{\mathbf{R}}\hat{\mathbf{A}})^{-1} \sim \hat{\mathbf{A}}^{-1}\hat{\mathbf{R}}^{-1} \sim \mathbf{A}\mathbf{R}^{(j)} \quad (22)$$

From this and $\mathbf{p}_e^{(j)}$, $\mathbf{T}^{(j)}$ can be extracted. These estimates, however will often not optimally fit the data. It is thus useful to further minimize the residual error

of the fit using bundle adjustment [18]. In the case of rectifying surfaces it is sufficient to minimize the residual fitting error with respect to the camera parameters, $\mathbf{A}, \mathbf{R}^{(j)}, \mathbf{T}^{(j)} \forall j$, and the control points, without altering the surface parameters $(s_i, t_i) \forall i$.

5 Experiments

In this section we demonstrate our approach by performing reconstructions on both real and artificial data sets. In our synthetic experiments, the 3D configuration shown in Figure 3(a) was considered. The structure in this setup consists of a 10×10 grid of points uniformally sampled on a saddle surface. The cameras are then placed around the object as shown. Reconstructions were computed using the method of Section 2 with bi-quadratic (i.e. $k_s = k_t = 3$) B-splines with six knots for the knot vectors associated with each parametric direction (τ_s and τ_t) . These reconstructions were performed with feature locations perturbed by additive white Gaussian noise with various standard deviations (σ_N) and were compared to the ground truth. This is illustrated in Figure 4. Here the average distance between the true saddle and spline surface is shown for a horizontal and vertical slice of the object as shown. In computing these plots the projectively reconstructed spline surface was first aligned with the ground truth surface using a homography that maps the recovered 3D feature points to their ground truth locations. Distances where then computed by finding the distance from the true saddle object to the closest point on the spline surface. The closest point was determined from dense samples of the spline surface.

In Figure 5 an example reconstruction is shown. Also shown is the distance between the ground truth saddle and the spline estimate. This plot is generated by finding the distance between the saddle and the closest point on the spline at various points on the saddle. From this plot we see the spline approximates the saddle well near the center.

Our reconstruction framework has also been evaluated using video sequences captured with a USB camera at 640×480 pixel resolution as shown in Figures 6, 7, and 8. The sequences depict a number of smooth textured objects whose shape can be modeled via simple rational B-spline patches.

Corresponding features were extracted by selecting points in one frame and tracking them with an iterative pyramidal Lucas-Kanade tracker that is available in OpenCV[1]. Following this, a few frames with disparate views were selected for use in reconstruction. Features in each selected input frame are shown in Figs. 6(a,b), 7(a,b), and 8(a,b) as (o)'s.

In the first example, Figure 6, six views with 67 features were taken. The spline was recovered using the method of Section 2 and rectified using the method of Section 4. The spline model used for this surface consisted of a bi-quadratic spline with six uniformly spaced knots in each parametric direction. Samples of the observed feature points along with the feature locations predicted by the recovered 2D splines are shown in Figures 6(a,b) as (o)'s and (+)'s respectively. Note that the cylindrical cross-section of the bottle is actually a rounded triangular shape. The subsequent 3D surface reconstruction for this patch of the object is shown in 6(c,d). From these plots we see the reconstructed surface appears similar to the true object.

In Figure 7, a cup was reconstructed from six views and 98 features. The spline recovered consists of a bi-quadratic spline with six uniformly spaced knots in each parametric direction. The spline was fit using the method of Section 2, but was rectified using orthographic factorization described in [11]. The reconstruction is shown in Figure 7(c,d). Finally, the reconstruction of a mushroom cap is shown in Figure 8. Five views and 89 features were used in this reconstruction. The recovered spline is a bi-quadratic spline with five uniformly spaced knots in each parametric direction. The spline was fit using the method of Section 2, and rectified using orthographic factorization.

Finally, the hierarchal fitting scheme was evaluated on synthetic object shown in Figure 9(a) and (b). In particular, note the complexity of this object. It consists of many bumps, as well as some discontinuous changes in curvature. Four views were generated and Gaussian noise was added to the generated feature locations with standard deviation of .1. An initial low order bi-quadratic spline with 7 uniformly spaced knots in each of the parametric directions was then reconstructed. This spline was aligned with the ground truth and shown in Figure 9(c). Following this the hierarchal fitting scheme of Section 3 was applied 10 times yielding the result in Figure 9(d). The final reconstruc-

[1] http://www.intel.com/research/mrl/research/opencv/

tion had 17 knots in each parametric direction. From this we can see that the initial reconstruction, which smoothed out many of the details of the surface, can be extended to exhibit some of the details of the true surface, when enough knots are inserted.

The hierarchal fitting scheme was also evaluated on the nine view sequence with 192 features shown in Figure 10. Here the initial spline was bi-quadratic with seven knots in each parametric direction. Subdivision was applied 6 times, resulting in the bi-quadratic patch with 16 knots in each parametric direction shown in Figure 10(c).

6 Discussion and Future Work

As exhibited in the experiments this approach is able to extract the shape of surfaces modeled as 3D rational spline patches. While the basic shape of the object was recovered, the visual quality of the reconstructions depends on how well features were extracted. In particular solutions exhibit oscillation in places of the surface when there were not many or poorly tracked features. This behavior is expected; to compensate, smoothness terms could be added to the objective function, (18). Despite this shortcoming, results of this method are promising.

In future work we hope to enhance this system in several ways. First, we plan to incorporate feature estimation within the framework. Currently, feature estimation is a separate step. We expect better overall performance if feature tracking is incorporated directly within surface estimation framework. By doing this we may incorporate other image features, such as lines and the silhouette edges of the surface.

Another area in which we intend to develop is the initial selection of the (s, t) for each feature point. Presently these are found from a frontal facing view of the object, but perhaps views can be combined to extract the (s, t)'s for arbitrary complex surfaces.

We also plan to extend the surface representation to deal with more complex surfaces by considering closed spline surfaces and by detecting and modeling surface creases.

References

[1] A. Blake and M. Isard. *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*, chapter 7. Springer, 1999.

[2] T. Cham and R. Cipolla. Stereo coupled active contours. In *Proc. CVPR*, pages 1094–1099, 1997.

[3] G. Cross and A. Zisserman. Quadric surface reconstruction from dual-space geometry. In *Proc. ICCV*, pages 25–31, 1998.

[4] G. Farin. *Nurbs: From Projective Geometry to Practical Use*. A K Peters Ltd, 2nd edition, 1999.

[5] P. Fua. From multiple stereo views to multiple 3D surfaces. *IJCV*, 24(1):19–35, 1997.

[6] P. Fua and G. Leclerc. Taking advantage of image-based and geometry-based constraints to recover 3-D surfaces. *CVIU*, 64(1):111–127, 1996.

[7] A. Hartley, R. Zisserman. *Multiple View Geometry in Computer Vision*. MIT Press, 2000.

[8] J. Lombardo and C. Puech. Oriented particles: A tool for shape memory objects modeling. In *Proc. Graphics Interface*, 1995.

[9] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, 1988.

[10] A. Shashua and S.Toelg. The quadric reference surface: Theory and applications. *IJCV*, 23(2):185–198, 1997.

[11] M. Siddiqui and S. Sclaroff. Surface reconstruction from multiple views using rational B-splines. In *Proc. CVPR: Technical Sketches*, 2001.

[12] P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *Proc. ECCV*, pages 709–720, 1996.

[13] R. Szeliski and J. Coughlan. Hierarchical spline-based image registration. *IJCV*, 22(3):199–218, 1997.

[14] R. Szeliski and H. Shum. Motion estimation with quadtree splines. *PAMI*, 18(12):1199–1210, 1996.

[15] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. In *SIGGRAPH*, 1992.

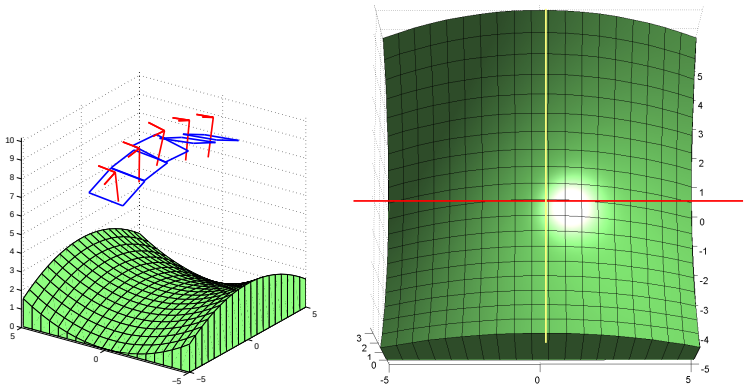
[16] R. Szeliski and P. Torr. Geometrically constrained structure from motion: Points on planes. In *Proc. SMILE*, pages 171–186, 1998.

[17] B. Triggs. Autocalibration and the absolute quadric. In *Proc. CVPR*, pages 609–614, 1997.

[18] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – A modern synthesis. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000.

[19] J. Weng, T. Huang, and N. Ahuja. Motion and structure from point correspondences: A robust algorithm for planar case with error estimation. In *Proc. ICPR*, pages 247–251, 1988.

[20] Z. Z. Y. Shan, Z. Liu. Model-based bundle adjustment with application to face modeling. In *Proc. ICCV*, 2001.



(a) (b)

Figure 3. Here the experimental setup of the synthetic data is shown. In (a) the object is shown along with the cameras used for imaging. In (b) the cross sections which are used for in comparing the ground truth object with the spline estimate are highlighted.

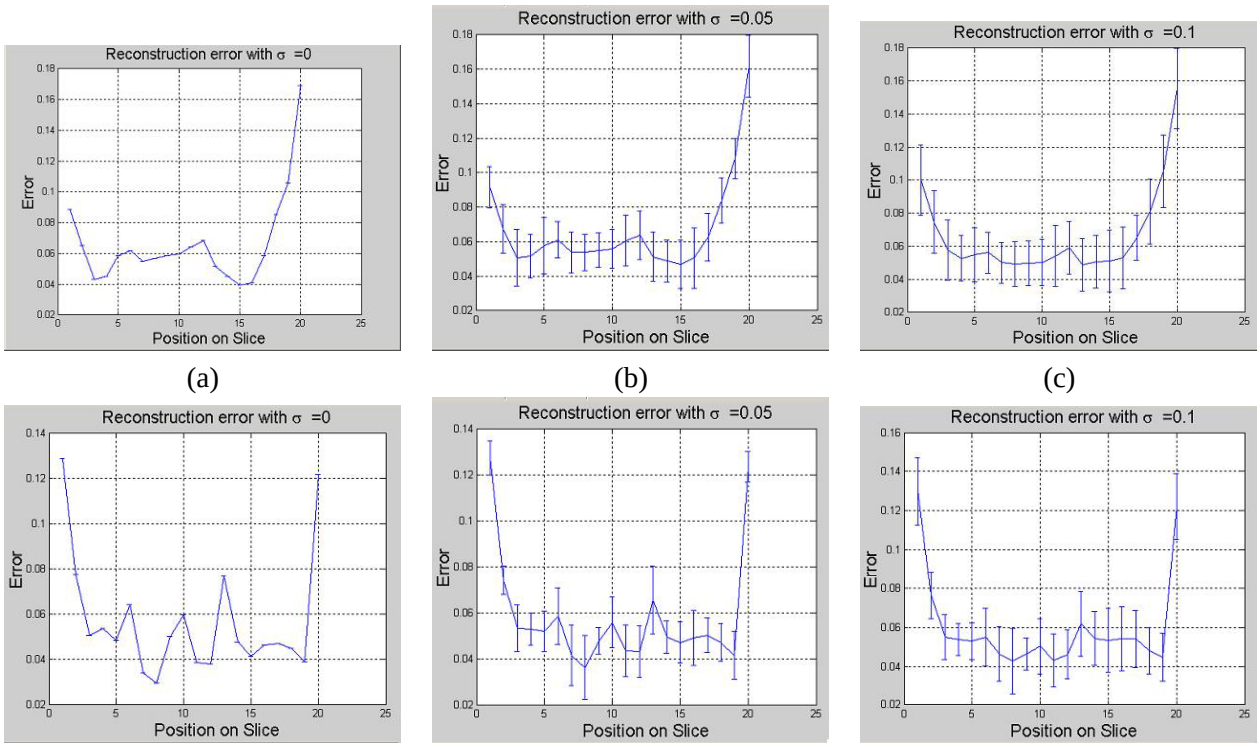
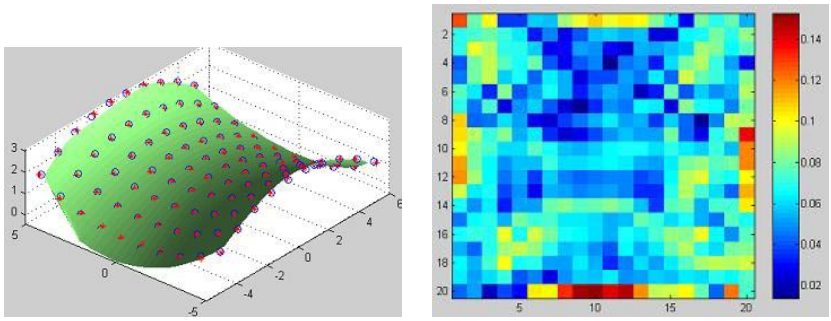
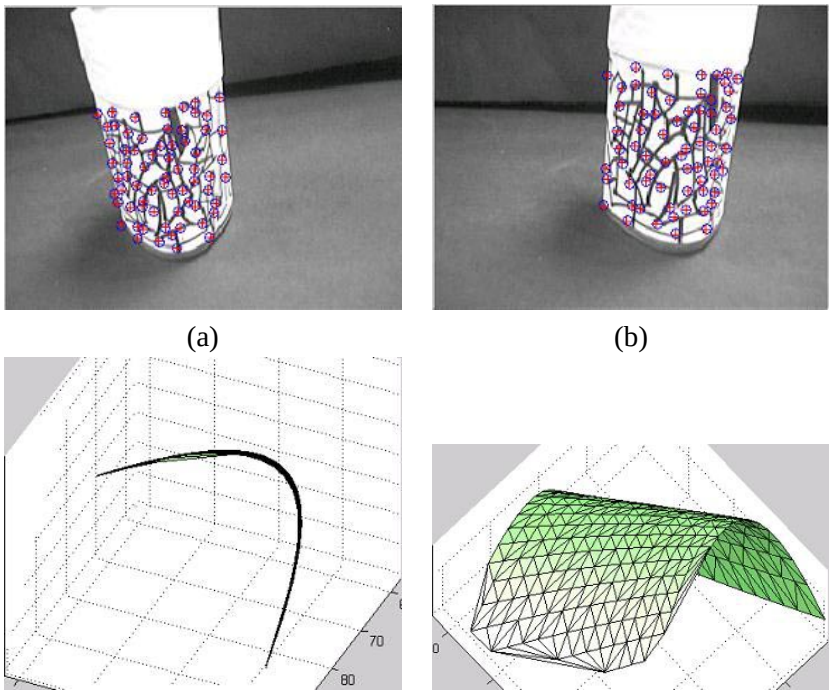


Figure 4. Results of sensitivity analysis. Here the fitting error between the ground truth object and the spline estimate is evaluated along the horizontal and vertical slices shown in Figure 3 (b). The top row corresponds to the horizontal slice and the bottom row the the vertical. The error is computed as average minimum distance from a point on the true object to the projectively aligned reconstruction. Each column corresponds to the the level of Gaussian noise added to the feature observation generated in the configuration shown in Figure 3(a).



(a) (b)

Figure 5. Example reconstruction of the synthetic data generated from the configuration shown in 3 (a). In (a) the reconstruction is shown. The (o)'s represent the ground truth sample point locations while the (*)'s indicate the spline estimates for those features. Also shown in green is the spline surface estimate. In (b) the aligned error between the actual saddle surface and the spline estimate is shown. The error is the minimum distance between samples on the true saddle and the the spline estimate. These distances are found by from samples of the spline surface.



(a) (b)

(c) (d)

Figure 6. Example reconstruction of a wedge shaped bottle. Here the reconstruction algorithm is applied to a six frame sequence. Two frames of the sequence are show in (a) and (b). Also shown are the 67 tracked features as (o)'s and the spline predicted feature locations as (+)'s. in (c) and (d) are two views of the reconstructed surface. Notice the recovered wedge shape is consistent with the true shape of the object.

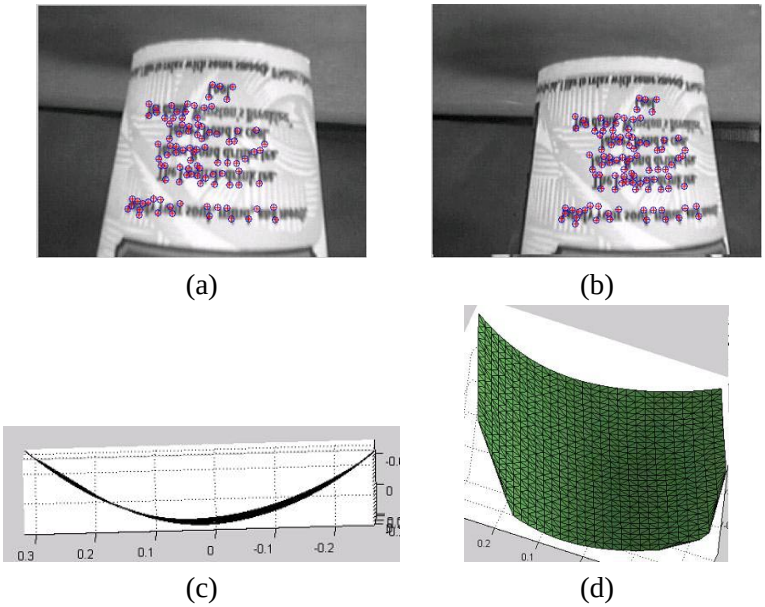


Figure 7. Example reconstruction of a cup. Here the reconstruction algorithm is applied to a six frame sequence. Two frames of the sequence are shown in (a) and (b). Also shown are the 98 tracked features as (o)'s and the spline predicted feature locations as (+)'s. in (c) and (d) are two views of the reconstructed surface. Notice the cylindrical shape of the spline reconstruction is consistent with the true shape of the object.

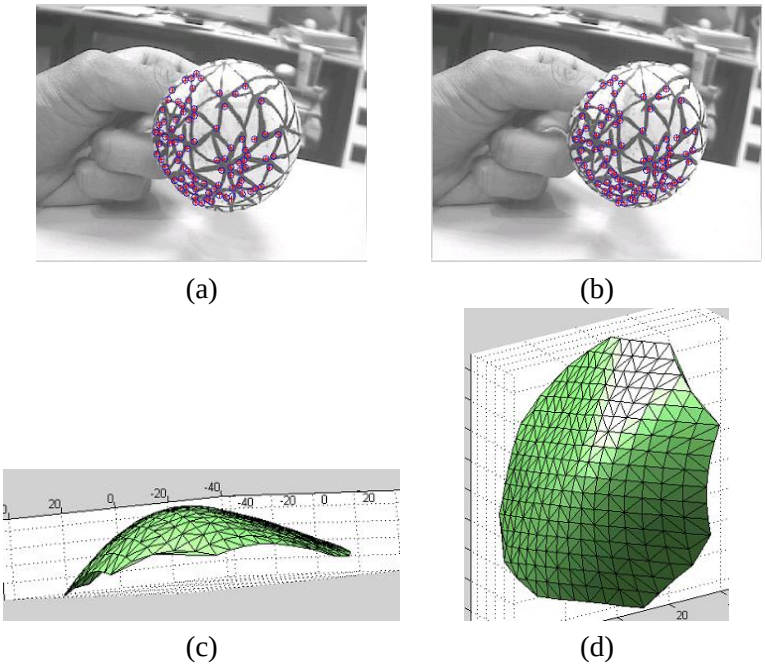


Figure 8. Example reconstruction of a mushroom. Here the reconstruction algorithm is applied to a five frame sequence. Two frames of the sequence are show in (a) and (b). Also shown are the 89 tracked features as (o)'s and the spline predicted feature locations as (+)'s. in (c) and (d) are two views of the reconstructed surface.

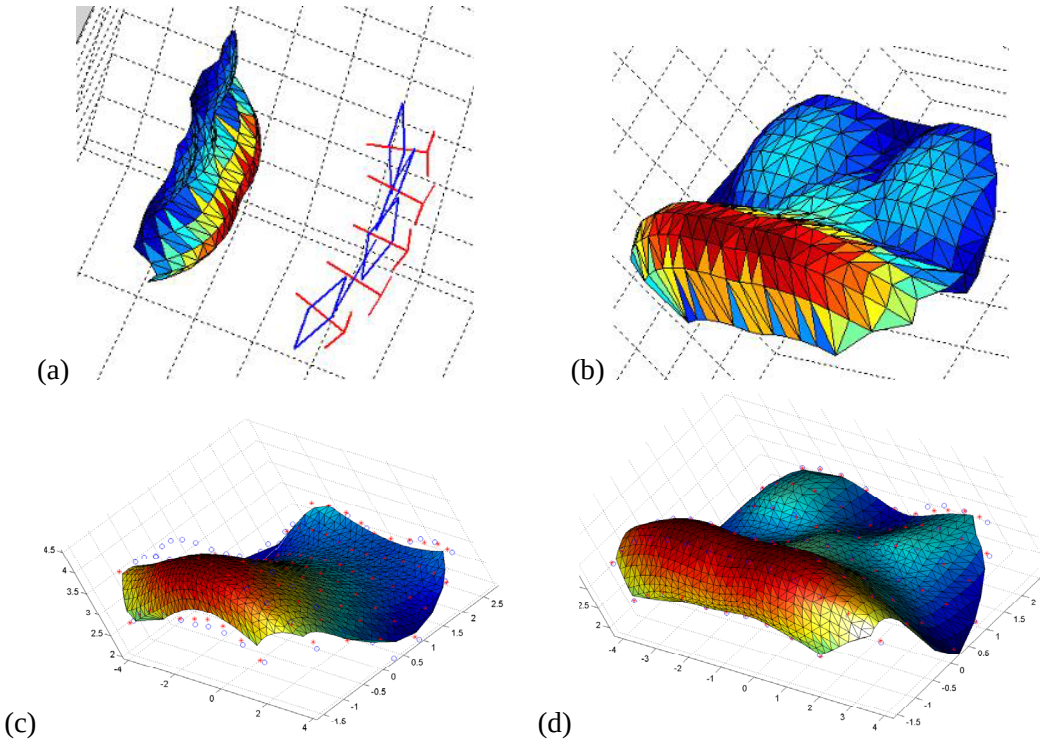


Figure 9. Example reconstruction using hierarchal fitting. Here the synthetic scene shown in (a) is reconstructed by using the hierarchal fitting scheme described in Section 3. The actual object is shown in (b). A reconstruction using a bi-quadratic spline with 7 knots in each parameteric direction is shown in (c). Here the reconstruction is projective and was aligned by means of a homography that maps the estimated feature locations, the (+)'s, to the ground truth, the (o)'s. In (d), the reconstruction after 10 subdivisions is shown. Notice that in (c) the general shape is recovered with many of the details smoothed out, while in (d) more surface detail is recovered.

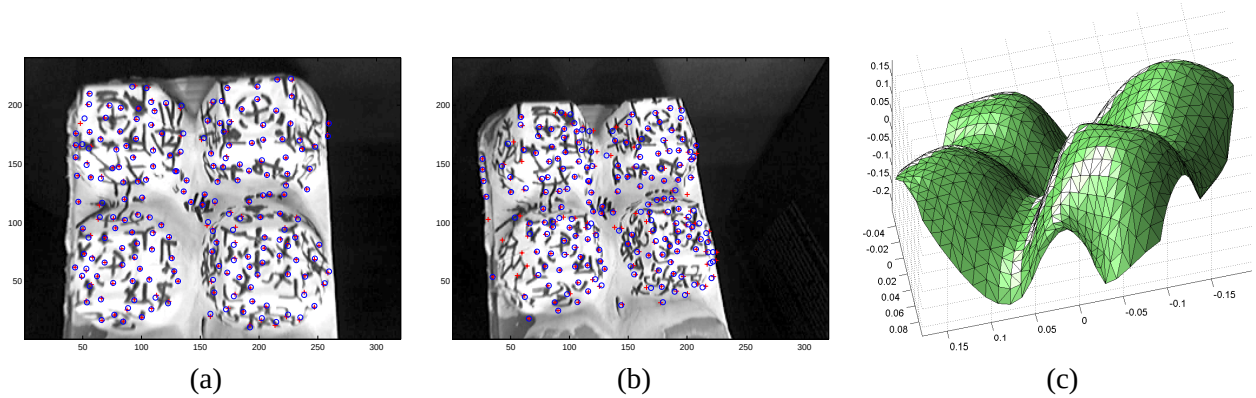


Figure 10. Example hierarchical reconstruction of an egg carton from a nine frame sequence. Two frames of the sequence are shown in (a,b). Also shown are the 192 tracked features as (o)'s and the spline predicted feature locations as (+)'s. Some of the features are not present in all the frames. The reconstructed surface is shown in (c). The shape of the spline reconstruction is consistent with the true shape of the object.