Single View Modeling of Free-Form Scenes

Li Zhang[†] Guillaume Dugas-Phocion[‡] Jean-Sebastien Samson[‡] Steven M. Seitz[†]
[†]Department of Computer Science and Engineering, University of Washington, Seattle, WA, 98195
[‡]Ecole Polytechnique, 91128 Palaiseau Cedex, France

Abstract

This paper presents a novel approach for reconstructing free-form, texture-mapped, 3D scene models from a single painting or photograph. Given a sparse set of user-specified constraints on the local shape of the scene, a smooth 3D surface that satisfies the constraints is generated. This problem is formulated as a constrained variational optimization problem. In contrast to previous work in single view reconstruction, our technique enables high quality reconstructions of free-form curved surfaces with arbitrary reflectance properties. A key feature of the approach is a novel hierarchical transformation technique for accelerating convergence on a non-uniform, piecewise continuous grid. The technique is interactive and updates the model in real time as constraints are added, allowing fast reconstruction of photorealistic scene models. The approach is shown to yield high quality results on a large variety of images.

1 Introduction

One of the most impressive features of the human visual system is our ability to infer 3D shape information from a single photograph or painting. A variety of strong single-image cues have been identified and used in computer vision algorithms (e.g. shading, texture, and focus) to model objects from a single image. However, existing techniques are not capable of robustly reconstructing free-form objects with general reflectance properties. This deficiency is not surprising given the ill-posed nature of the problem–from a single view it is not possible to differentiate an image of an object from an image of a flat photograph of the object. Obtaining good shape models from a single view therefore requires invoking domain knowledge.

In this paper, we argue that a reasonable amount of user interaction is sufficient to create high-quality 3D scene reconstructions from a single image, *without* placing strong assumptions on either the shape or reflectance properties of the scene. To justify this argument, an algorithm is presented that takes as input a sparse set of user-specified constraints, including surface positions, normals, silhouettes, and creases, and generates a well-behaved 3D surface satisfying the constraints. As each constraint is specified, the system recalculates and displays the reconstruction in real time. The algorithm yields high quality results on images with limited perspective distortion.

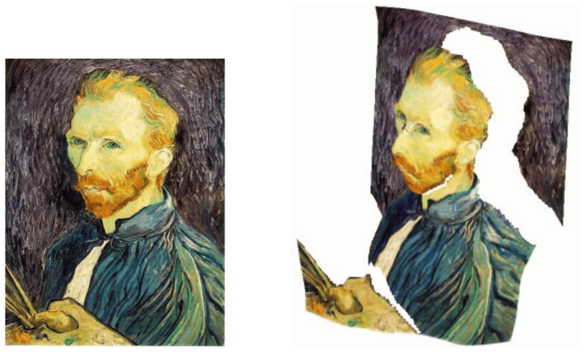


Figure 1: The 3D model at right is generated from a single image and user-specified constraints.

We cast the single-view modeling problem as a constrained variational optimization problem. Building upon previous work in hierarchical surface modeling [20, 8, 21], the scene is modeled as a piecewise continuous surface represented on a quad-tree-based adaptive grid and is computed using a novel hierarchical transformation technique. The advantages of our approach are:

- A general constraint mechanism: any combination of point, curve, and region constraints may be specified as image-based constraints on the reconstruction.
- Adaptive resolution: the grid adapts to the complexity of the scene, i.e., the quad-tree representation can be made more detailed around contours and regions of high curvature.
- Real-time performance: a hierarchical transformation technique is introduced that enables 3D reconstruction at interactive rates.

A technical contribution of our algorithm is the formulation of a hierarchical transformation technique that handles discontinuities. *Controlled-continuity stabilizers* [24] have been proposed to model discontinuities in inverse visual reconstruction problems. Whereas hierarchical schemes [20, 8] and quad-tree splines [21] have been put forth as fast solutions for solving related variational problems, our approach for integrating discontinuity conditions into a hierarchical transformation framework is shown to yield significant performance improvements over these prior methods.

The remainder of the paper is structured as follows. Section 2 formulates single-view modeling as a constrained optimization problem in a high dimensional space. In order to solve this large scale optimization problem efficiently with adaptive resolution, a novel hierarchical transformation technique is introduced in Section 3. Section 4 presents experimental results and Section 5 concludes.

1.1 Previous Work on Single View Modeling

The topic of 3D reconstruction from a single image is a long-standing problem in the computer vision literature. Traditional approaches for solving this problem have isolated a particular cue, such as shading [9], texture [19], or focus [15]. Because these techniques make strong assumptions on shape, reflectance, or exposure, they tend to produce acceptable results for only a restricted class of images. Of these, the topic of *shape from shading*, pioneered by Horn [9], is most related to our approach in its use of variational techniques and ability to fit free-form models from normal fields.

More recent work by a number of researchers has shown that moderate user-interaction is highly effective in creating 3D models from a single view. In particular, Horry et al. [10] and Criminisi et al. [4] reconstructed piecewise planar models based on user-specified vanishing points and geometric invariants. Shum and Szeliski [18] generated similar models from panoramas using a constraint system based on user input. The *Façade* system [6] modeled architectural scenes using collections of simple primitives from one or more images, also with the assistance of a user. A limitation of these approaches is that they are limited to scenes composed of planes or other simple primitives and do not permit modeling of free-form scenes. A different approach is to use domain knowledge; for example, Blanz and Vetter [1] have obtained remarkable reconstructions of human faces from a single view by employing a database of previously-acquired head models.

A primary source of inspiration for our work is a series of papers on the topic of *Pictorial Relief* [13]. In this work, Koenderink and his colleagues explored the depth perception abilities of the human visual system by having several human subjects hand-annotate images with relative distance or surface normal information. They found that humans are quite proficient at specifying local surface orientation, i.e., normals, and that integrating a dense user-specified normal field leads to a well-formed surface that approximates the real object, up to a depth scale. Interestingly, the depth scale varies across individuals and is influenced by illumination conditions. We believe that the role of this depth scale is mitigated in our work, due to the fact that we allow the user to view the reconstruction from any viewpoint(s) during the modeling process–the user will set the normals and other constraints so that the model appears correct from *all* viewpoints, rather than just the original view. The surface integration technique used by Koenderink et al. is not attractive as a general purpose modeling tool, due to the large amount of human labor needed to annotate every pixel or grid point in the image. Although it is also based on the principles put forth in the Pictorial Relief work, our modeling technique is much more efficient, works from sparse constraints, and incorporates discontinuities and other types of constraints in a general-purpose optimization framework.

An interesting alternative to the approach advocated in this paper is to treat the scene as an intensity-coded depth image and use traditional image editing techniques to sculpt the depth image [27, 12, 16]. While our framework allows direct specification of depth values, we found that surface normals are easier to specify and provide more intuitive surface controls. This conclusion is consistent with Koenderink's findings [13] that humans are more adept at perceiving local surface orientation than relative depth.

2 A Variational Framework for Single View Modeling

The subset of a scene that is visible from a single image may be modeled as a piecewise continuous surface. In our approach, this surface is reconstructed from a set of user-specified constraints, such as point positions, normals, contours, and regions. The problem of computing the best surface that satisfies these constraints is cast as a constrained optimization problem.

2.1 Surface Representation

In this paper, the scene is represented as a piecewise continuous function, $f(x, y)$, referred to as the *depth map*. Samples of f are represented on a discrete grid, $g_{i,j} = f(id, jd)$, where the i and j samples correspond to pixel coordinates of the input image, and d is the distance between adjacent samples, assumed to be the same in x and y. Denote $\mathbf{g}$ as the vector whose components are $g_{i,j}$.

A set of four adjacent samples, A=(i, j), B=$(i + 1, j)$, C=$(i + 1, j + 1)$, and D=$(i, j + 1)$ define the corners of a grid *cell*. Note that a cell, written as A-B-C-D, is specified by its vertices listed in counter-clock-wise order.

The technique presented in this paper reconstruct the smoothest surface that satisfies a set of user-specified constraints. A natural measure of surface smoothness is the *thin plate* functional [24]:

$$Q_0(\mathbf{g}) = \frac{1}{2d^2}\sum_{i,j}[\alpha_{i,j}(g_{i+1,j} - 2g_{i,j} + g_{i-1,j})^2 \\ +2\beta_{i,j}(g_{i+1,j+1} - g_{i,j+1} - g_{i+1,j} + g_{i,j})^2 \\ +\gamma_{i,j}(g_{i,j+1} - 2g_{i,j} + g_{i,j-1})^2] \quad (1)$$

where $\alpha_{i,j}$, $\beta_{i,j}$, and $\gamma_{i,j}$ are weights that take on values of 0 or 1 and are used to define discontinuities, as described in Section 2.2.2.

2.1.1 Piecewise Continuous Surface Representation

While it is convenient to represent a surface by a grid of samples, users should have the freedom to interact with a continuous surface by specifying constraints at any location with sub-grid accuracy. Given a sampled surface $g_{i,j}$, we represent the continuous surface $f(x, y)$ using a triangular mesh. Specifically, each grid cell is divided into four triangles by inserting a vertex at the center with depth defined as the average of the depths of the four corner samples, and adding edges connecting the new vertex with the four corners. The resulting mesh defines a piecewise planar surface over the cell. The depth of each point in the cell can be expressed as a barycentric combination of the depth values of four corner samples. Grid cells that intersect discontinuity curves are omitted from the representation and appear as gaps in the reconstruction.

2.2 Constraints

Our technique supports five types of constraints: point constraints, depth discontinuities, creases, planar region constraints, and fairing curve constraints. Point constraints specify the position or the surface normal of any point on the surface. Surface discontinuity constraints identify tears in the surface, and crease constraints specify curves across which surface normals are not continuous. Planar region constraints determine surface patches that lie on the same plane. Fairing curve constraints allow users to control the smoothness of the surface along any curve in the image. Each constraint is described in detail in the following subsections.

2.2.1 Point Constraints

A point constraint sets the depth and/or the surface normal of any point in the input image. A position constraint is specified by clicking at a point in the image to define the (sub-pixel) position (x_0, y_0), and then dragging up or down to specify the depth value. A surface normal is specified by rendering a figure representing the projection of a disk sitting on the surface with a short line pointing in the direction of the surface normal (Figure 2(a)). This figure is superimposed over the point in the image where the normal is to be specified and manually rotated until it appears to align with the surface in the manner proposed by [13]. In order to uniquely determine the normal from its image plane projection, *we assume orthographic projection*.

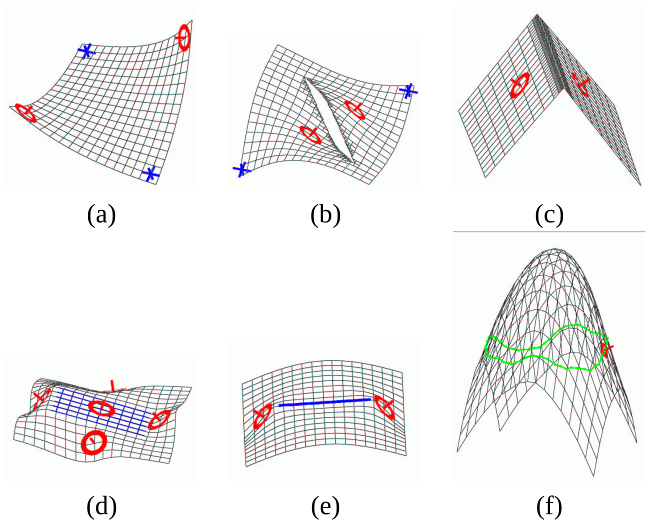


Figure 2: Modeling constraints. (a) The effects of position (blue crosses) and surface normal constraints (red disks with needles). (b) A depth discontinuity constraint creates a tear. (c) A crease constraint (green curve). (d) The blue region is a planar region constraint. (e) A fairing curve minimizing curvature. (f) A fairing curve minimizing torsion makes the surface bend smoothly given a single normal constraint–this type of constraint is useful for modeling silhouettes.

A position constraint $f(x_0, y_0) = f_0$ defines the following constraint

$$c_{00}g_{i,j} + c_{10}g_{i+1,j} + c_{01}g_{i,j+1} + c_{11}g_{i+1,j+1} = f_0 \quad (2)$$

where (x_0, y_0) is located in grid cell $[id, (i+1)d]\times[jd, (j+1)d]$ and c_{00}, c_{01} ,c_{10}, and c_{11} are the barycentric coordinates of (x_0, y_0), as described in Section 2.1.1. Specifying the normal of a point (x_0, y_0) to be $(N_x, N_y, N_z)^{\mathrm{T}}$, defines the following pair of constraints

$$\frac{f(x_0 + d, y_0) - f(x_0 - d, y_0)}{2d} = -\frac{N_x}{N_z} \quad (3)$$

$$\frac{f(x_0, y_0 + d) - f(x_0, y_0 - d)}{2d} = -\frac{N_y}{N_z} \quad (4)$$

Substituting Eq. (2) for $f(x_0 \pm d, y_0)$ and $f(x_0, y_0 \pm d)$ yields two linear constraints on $\mathbf{g}$. An example of the effects of position and normal constraints is shown in Figure 2(a).

2.2.2 Depth Discontinuities and Creases

A depth discontinuity is a curve across which surface depth is not continuous, creating a tear in the surface. A crease is a curve across which the surface normal is not continuous while the surface depth *is* continuous. Depth discontinuities and creases are introduced to model important features in real-world imagery. For example, mountain ridges can be modeled as creases and silhouettes of objects can be modeled as depth discontinuities. These features can be easily specified by users with a 2D graphics interface.

Depth discontinuities and creases are modeled by defining the weights $\alpha_{i,j}$, $\beta_{i,j}$, and $\gamma_{i,j}$ in the smoothness objective function of Eq. (1). Given a depth discontinuity curve, let A-B-C be a set of three consecutive colinear grid points that cross the curve, and D-E-F-G a cell that the curve intersects. For each such tuple A-B-C, the term $(g_A - 2g_B + g_C)^2$ is dropped from Q_0 by setting α_B or γ_B to 0. For each such cell D-E-F-G, the term $(g_D - g_E - g_F + g_G)^2$ is also dropped by setting β_G to 0. Each crease curve is first *scan converted* [7] to the sampling grid points. Then, all the terms $(g_A - 2g_B + g_C)^2$ are dropped if B is on the curve; all the terms $(g_D - g_E - g_F + g_G)^2$ are dropped if either edge D-E or edge F-G is on the curve. Otherwise, all the weights are 1 by default. Examples of depth discontinuity and crease constraints are shown in Figures 2(b) and (c) respectively.

2.2.3 Planar Region Constraints

The necessary and sufficient conditions for surface planarity over a region R are $f_{xx}(x,y) = f_{xy}(x,y) = f_{yy}(x,y) = 0$, $\forall (x,y) \in R$, and define the following constraints on $\mathbf{g}$

$$g_A - 2g_B + g_C = 0 \tag{5}$$

$$g_D - g_E - g_F + g_G = 0 \tag{6}$$

for all three consecutive colinear grid points A-B-C in R, and for all cells D-E-F-G in R. An example of a planar region constraint is shown in Figure 2(d).

2.2.4 Fairing Curve Constraints

It is often very useful for users to control the smoothness of the surface both along and across a specific curve. For example, surface depth is made to vary slowly *along* a curve in Figure 2(e), and the surface gradient is made to vary slowly *across* a curve in Figure 2(f). Fairing curves provide better control of the shape of the surface along salient contours such as silhouettes, and are achieved as follows.

Suppose that a user specifies a curve $\tau(l) = (x(l), y(l))^{\mathrm{T}}$ in the image. To maximize the smoothness along the curve, the following integral is minimized

$$Q_d(\tau) = \int_l \left(\frac{d^2}{dl^2} f(\tau(l))\right)^2 dl \tag{7}$$

The gradient of the surface across τ is $(\nabla f)^{\mathrm{T}} \mathbf{n}_\tau$, where $\nabla f = (f_x, f_y)^{\mathrm{T}}$ is the gradient of the surface $f(x,y)$ at the point $\tau(l)$ and $\mathbf{n}_\tau(l) = (-\frac{d}{dl}y(l), \frac{d}{dl}x(l))^{\mathrm{T}}$ is the normal of τ. To make the surface gradient across $\tau(l)$ have small variation, the integral

$$Q_s(\tau) = \int_l \left(\frac{d}{dl}((\nabla f)^{\mathrm{T}} \mathbf{n}_\tau)\right)^2 dl \tag{8}$$

is minimized. Note that $\frac{d}{dl}((\nabla f)^{\mathrm{T}} \mathbf{n}_\tau)$ is the derivative of the surface gradient *across* the curve with respect to the curve parameter.

The terms $\frac{d^2}{dl^2} f(\tau(l))$ and $\frac{d}{dl}((\nabla f)^{\mathrm{T}} \mathbf{n}_\tau)$ may be discretized as

$$\frac{d^2}{dl^2} f(\tau(l_i)) = f(\tau(l_{i+1})) - 2f(\tau(l_{i-1})) + f(\tau(l_{i-1}))$$

$$((\nabla f)^{\mathrm{T}} \mathbf{n}_\tau)(l_i) = f((\tau + \frac{d}{2} n_\tau)(l_{i+1})) - f((\tau - \frac{d}{2} \mathbf{n}_\tau)(l_{i+1}))$$

$$\frac{d}{dl}((\nabla f)^{\mathrm{T}} \mathbf{n}_\tau)(l_i) = ((\nabla f)^{\mathrm{T}} \mathbf{n}_\tau)(l_{i+1}) - ((\nabla f)^{\mathrm{T}} \mathbf{n}_\tau)(l_i)$$

where $\{\tau(l_i)\}$ are sampling points on the curve. Consequently, Eqs. (7) and (8), can be expressed as quadratic forms of $\mathbf{g}$. The resulting equations are added, with weights ζ_τ and η_τ, into Eq. (1), resulting in a modified surface smoothness objective function $Q(\mathbf{g})$:

$$\begin{aligned} Q_c(\tau) &= \zeta_\tau Q_d(\tau) + \eta_\tau Q_s(\tau) \\ Q(\mathbf{g}) &= Q_0(\mathbf{g}) + \sum_\tau Q_c(\tau) \end{aligned} \tag{9}$$

We call $\zeta_\tau Q_d(\tau)$ the *curvature* term and $\eta_\tau Q_s(\tau)$ the *torsion* term. Note that $Q(\mathbf{g})$ is a quadratic form.

2.3 Linearly Constrained Quadratic Optimization

Based on the surface objective function and constraints presented in Section 2.1 and 2.2, finding the smoothest surface that satisfies these constraints may be formulated as a linearly constrained quadratic optimization. Point constraints and planar region constraints introduce a set of linear equations, Eqs. (2-6), for the depth map $\mathbf{g}$, expressed as $\mathbf{Ag} = \mathbf{b}$. Surface discontinuity and crease constraints define weights α, β, and γ and fairing curve constraints introduce $\sum_\tau Q_c(\tau)$ in Eq. (9). $Q(\mathbf{g})$ is a quadratic form and can be expressed as $\mathbf{g}^{\mathrm{T}}\mathbf{H}\mathbf{g}$, where $\mathbf{H}$ is the Hessian matrix. Consequently, our linearly constrained quadratic optimization is defined by

$$\begin{cases} \mathbf{g}^* = \arg\min_{\mathbf{g}}\{Q(\mathbf{g}) = \mathbf{g}^{\mathrm{T}}\mathbf{H}\mathbf{g}\} \\ \texttt{subject to } \mathbf{A}\mathbf{g} = \mathbf{b} \end{cases} \tag{10}$$

The Lagrange multiplier method is used to convert this problem into the following augmented linear system

$$\begin{bmatrix} \mathbf{H} & \mathbf{A}^{\mathrm{T}} \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{g} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix} \tag{11}$$

The Hessian matrix $\mathbf{H}$ is a diagonally-banded sparse matrix. For a grid of size N by N, $\mathbf{H}$ is of size N^2 by N^2, with band width of $O(N)$ and about 13 non-zero elements per row. Direct methods, such as LU Decomposition, are of $O(N^4)$ time complexity, and are therefore do not scale well for large grid sizes. Iterative methods are more applicable. We use the *Minimum Residue* method [17], designed for symmetric non-positive-definite systems. However, the linear system arising from Eq. (10) is often poorly conditioned, resulting in slow convergence of the iterative solver. To address this problem, a hierarchical basis preconditioning approach with adaptive resolution is presented in the next section.

3 Hierarchical Transformation with Adaptive Resolution

The reason for the slow convergence of the Minimum Residue method is that it takes many iterations to propagate a constraint to its neighborhood, due to the sparseness of $\mathbf{H}$. The first row of Figure 4 shows an example of this constraint propagation process, where the two normal constraints generate only two small ripples after 200 iterations. Multigrid techniques [23] have been applied to this type of problem, however, they are tricky to implement and require a fairly smooth solution to be effective [20]. Szeliski [20] and Gortler et al. [8] use hierarchical basis functions to accelerate the solution of linear systems like Eq. (11). We review their approach next, to provide a foundation for our work which builds upon it.

In the hierarchical approach, a regular grid is represented with a pyramid of coefficients [3], where the number of coefficients is equal to the original number of grid points. The coarse level coefficients in the pyramid determine a low resolution surface sampling and fine level coefficients determine surface details, represented as displacements relative to the interpolation of the low resolution sampling. To convert from coefficients to depth values, the algorithm starts from the coarsest level, doubles the resolution by linearly interpolating the values of current level, adds in the displacement values defined by the coefficients in the next finer level, moves to the next finer level, and repeats the procedure until the finest resolution is obtained. Using similar notation as Szeliski's [21], the process can be written

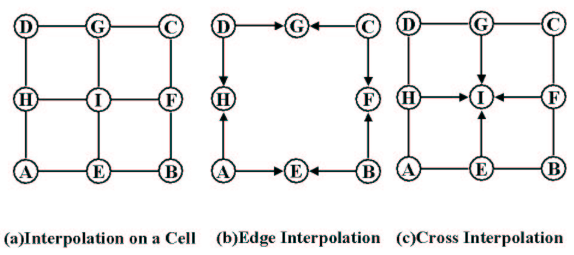


Figure 3: A cell is the primitive for 2D hierarchical transformation. The depth at the center point I is interpolated from the midpoints E, F, G, and H, which are in turn interpolated along each edge of the cell.

```
procedure  CoefToDepth($coef$)
   for $l = L - 1$ down to 1
      for every grid point $P$ in level $l$
         $depth_P = coef_P + \sum_{Q \in N_P} w_{P,Q} * depth_Q$
   return depth
end  CoefToDepth
```

where L is the number of levels in hierarchy, $coef_P$ is the hierarchical coefficient for P, N_P is the set of grid points in level $l-1$ used in interpolation for P in level l, and $w_{P,Q}$ is a weight that will be described later. Level 0 consists of a single cell, with coefficients defined to be the depth values at the corners of the cell.

In previous work, the weights $w_{P,Q}$ were defined to average all the points in N_P, resulting in a simple averaging operation for computing P from N_P. This approach implicitly assumes local smoothness within the region defined by N_P, resulting in poor convergence in the presence of discontinuities. In practice, this choice of weights causes the artifact that modifying the surface on one side of a discontinuity boundary disturbs the shape on the other side during the iterative convergence process. As a result, it takes longer to converge to a solution, and results in unnatural convergence behavior. The latter artifact is a problem in an incremental solver where the evolving surface is displayed for user consumption, as is done in our implementation. To address this problem, we next introduce a new interpolation rule to handle discontinuities between the grid points in N_P.

The basic unit in the 2D hierarchial transformation technique is the cell shown in Figure 3, where the depth for corners A, B, C, and D has already been computed and the task is to transform coefficients at E, F, G, H, and I to depth values at these points. With the same notation as in the procedure, CoefToDepth, $N_E = \{A, B\}$, $N_F = \{B, C\}$, $N_G = \{C, D\}$,$N_H = \{D, A\}$, and $N_I = \{E, F, G, H\}$. g_E, g_F, g_G, and g_H are first interpolated from A, B, C, and D along edges, and then offset by their respective coefficients $\tilde{g}_E, \tilde{g}_F, \tilde{g}_G$, and $\tilde{g}_H$. Second, g_I is interpolated from g_E, g_F, g_G, and g_H and offset by its coefficient, $\tilde{g}_I$. The two interpolation steps above use *continuity-based interpolation* with weights defined as

$$w_{P,Q} = \begin{cases} \frac{e_{P,Q}}{\sum\limits_{Q \in N_P} e_{P,Q}} & \texttt{if} \sum\limits_{Q \in N_P} e_{P,Q} > 0; \\ 0 & \texttt{otherwise.} \end{cases}$$
$$\texttt{where}$$
$$e_{P,Q} = \begin{cases} 1 & \texttt{if edge P} - \texttt{Q is continuous;} \\ 0 & \texttt{otherwise.} \end{cases} \tag{12}$$

In the absence of discontinuities, the proposed *continuity-based weighting* scheme is the same as simple averaging schemes used in previous work [20, 21]. In the presence of discontinuities, only locally continuous coarse level grid points are used in the interpolation. The new scheme prevents interference across discontinuity boundaries and consequently accelerates the convergence of the Minimum Residue algorithm. The second and third rows of Figure 4 show a performance comparison between standard hierarchical transformation and our transformation with continuity-based weighting on a simple surface modeling problem with one discontinuity curve. The improvement of our algorithm is quite evident in this example. In the third row, our new transformation both accelerates the propagation of constraints and removes the interference across the discontinuity boundary. We have found this kind of behavior very typical in practice and find that adding continuity-based weighting yields dramatic improvements in system performance.

To summarize our approach in brief, instead of solving Eq. (11) directly, we solve the hierarchical coefficients $\tilde{\mathbf{g}}$ of the grid point $\mathbf{g}$ instead. The conversion from $\tilde{\mathbf{g}}$ to $\mathbf{g}$ is implemented by the procedure, CoefToDepth, with continuity-based weighting. The procedure implements a linear transformation and can be described by a matrix $\mathbf{S}$ [20]. Substituting $\mathbf{g} = \mathbf{S}\tilde{\mathbf{g}}$ into Eq. (10) and applying the Lagrange Multiplier method yields the transformed linear system [8]:

$$\begin{bmatrix} \mathbf{S}^\mathsf{T}\mathbf{H}\mathbf{S} & \mathbf{S}^\mathsf{T}\mathbf{A}^\mathsf{T} \\ \mathbf{A}\mathbf{S} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{g}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix} \tag{13}$$

The matrix $\mathbf{S}^\mathsf{T}\mathbf{H}\mathbf{S}$ is shown to be better conditioned [20], resulting in faster convergence. The number of floating point operations of the procedure $CoefToDepth$ and its adjoint [20] is approximately $4N^2$ for a grid size of $N \times N$. Considering that there are around 13 non-zero elements per row in $\mathbf{H}$, the overhead introduced by $\mathbf{S}$ in multiplying $\mathbf{S}^\mathsf{T}\mathbf{H}\mathbf{S}$ with a vector is about 30%. Given the considerable reduction in number of iterations shown in Figure 4, the total run time is generally much lower using a hierarchical technique, even with this overhead.

Adaptive Surface Resolution

As an alternative to solving for the surface on the full grid, it is often advantageous to use an adaptive grid, with higher resolution used only in areas where it is needed. For example, the surface should be sampled densely along a silhouette and sparsely in areas where the geometry is nearly planar. We support adaptive resolution by allowing the user to specify the grid resolution for each region via a user-interface. Subdivision may also occur automatically– in our implementation, discontinuity and crease curves are automatically subdivided to enable accurate boundaries. A quad-tree representation is used to represent the adaptive grid. By modifying our hierarchical transformation technique to operate on a quad-tree grid, as in [21], the run time of the algorithm is proportional to the number of subdivided grid points, which is typically much smaller than the full grid.

Modifying the algorithm to operate on a quad-tree requires the following changes. First, the triangular mesh representation in Section 2.1.1 is adapted so that each inserted vertex is connected to all the grid points on the cell, not just to the four corners. Second, expressions for the first and second derivatives of f in terms of $\mathbf{g}$ should be derived from the quad-tree representation, e.g., by interpolating a regular grid neighborhood around each point from the triangular mesh. Finally, special care should be taken to approximate surface and curve integrals by summations, e.g., Eq. (1), on the non-uniform grid, by weighing each term in the summation according to the size of the local neighborhood. The full details of these modifications are omitted here for lack of space, but can be found online [29].

4 Experimental Results

We have implemented the approach described in this paper and applied it to create reconstructions of a wide variety of objects. Only three of these results are presented in this section but higher resolution images and 3D VRML models can be found online [29]. We encourage the reader to peruse these results online to better gauge the quality of the reconstructions.

Smooth objects without position discontinuities are especially easy to reconstruct using our approach. As a case in point, the Jelly Bean image in the first row of Figure 5 requires only isolated normals and creases to generate a compelling model, and can be created quite rapidly (about 20 minutes, including time to specify constraints) using our interactive system. The first row of Figure 5 shows the input image, quad-tree grid with constraints, a view of the quadtree from a novel viewpoint, and a texture mapped rendering of the same view. For this example, the user worked with a 32×32 grid that was automatically subdivided as the crease curves were drawn. This model has 144 constraints in all, 3396 grid points, and required 25 seconds to converge completely on a 1.5GHz Pentium 4 processor, using our hierarchical transformation technique with continuity-based weighting. The system is designed so that new constraints may be added interactively at any time during the modeling process–the user does not have to wait until full convergence to specify more constraints. The second row of Figure 5 shows a single-view reconstruction of The Great Wall of China. This example was much more challenging than the Jelly Bean, due to the complex scene geometry and significant perspective distortions. Despite these obstacles, a 3D model was reconstructed that appears visually convincing from a significant range of views. This model has 135 constraints, 2566 grid points, and required 40 seconds to converge completely.

An interesting application of single view modeling techniques is to reconstruct 3D models from paintings. In contrast to other techniques [6, 10, 1, 4], our approach does not make strong assumptions about geometry, making it amenable to impressionist and other non-photorealistic works. Here we show a reconstruction created from a self-portrait of van Gogh. This model has 264 constraints, 3881 grid points, and required 45 seconds to converge. This was the most complex model we tried, requiring roughly 1.5 hours to design. For purposes of comparison, it takes 70 seconds to converge without using the hierarchical transformation and 3 minutes using the hierarchical transformation without continuity-based weighting, indicating that an inappropriately-weighted hierarchical method can perform significantly worse than not using a hierarchy at all. Note, however, that there is significant room for optimization in our implementation, and we expect that the timings for both hierarchical methods could be improved by a factor of 1.5 or 2.

5 Conclusions

In this paper, it was argued that a reasonable amount of user interaction is sufficient to create high-quality 3D scene reconstructions from a single image, *without* placing strong assumptions on either the shape or reflectance properties of the scene. To justify this argument, an algorithm was presented that takes as input a sparse set of user-specified constraints, including surface positions, normals, silhouettes, and creases, and generates a well-behaved 3D surface satisfying the constraints. As each constraint is specified, the system recalculates and displays the reconstruction in real time. A technical contribution is a novel hierarchial transformation technique that explicitly models discontinuities and enables surface computation at interactive rates. The approach was shown to yield very good results on real images.

There are a number of interesting avenues for future research in this area. In particular, single-view modeling has the inherent limitation that only visible surfaces in an image can be modeled, leading to distracting holes near occluding boundaries. Automatic hole filling techniques could be developed that maintain the surface and textural attributes of the scene. Another important extension would be to generalize to perspective projection as well as other useful projection models like panoramas.

References

[1] V. Blanz, T. Vetter, ”A morphable model for the synthesis of 3D faces”, *ACM SIGGRAPH Proceedings*, pp. 187–194, 1999.

[2] J.-Y. Bouguet and P. Perona, “3D photography on your desk”, *Int’l Conf. on Computer Vision*, pp. 43-50, 1998.

[3] P. J. Burt, E. H. Adelson, “The Laplacian pyramid as a compact image code”, *IEEE Trans. on Comm.* vol. 31, no. 4, pp. 532-540, 1983.

[4] A. Criminisi, I. Reid, and A. Zisserman, “Single view metrology”, *Int’l Conf. on Computer Vision*, pp.434-442, September 1999.

[5] B. Curless and M. Levoy, “A volumetric method for building complex models from range images”, *ACM SIGGRAPH Proceedings*, pp. 303-312, 1996.

[6] P. Debevec, C. Taylor, and J. Malik, “Façade: modeling and rendering architecture from photographs”, *ACM SIGGRAPH Proceedings*, pp. 11-20, 1996.

[7] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, *Computer Graphics: Principles and Practice*, Addison-Wesley Publishing Company, Inc., pp. 72-91, 1990.

[8] S. Gortler and M. Cohen, “Variational modeling with wavelets”, *TR-456-94*, Dept of Computer Science, Princeton Univ, 1994.

[9] B. K. P. Horn, “Height and gradient from shading”, Int’l J. of Computer Vision, vol. 5, no. 1, pp. 37-75, 1990.

[10] Y. Horry, K. Anjyo, K. Arai, “Tour into the picture: using a spidery mesh interace to make animation from a single image”, *ACM SIGGRAPH Proceedings*, pp. 225-232, 1997.

[11] T. Igarashi, S. Matsuoka and H. Tanaka, “Teddy: a sketching interface for 3D freeform design”, *ACM SIGGRAPH Proceedings*, pp. 409-416, 1999.

[12] S.B. Kang, “Depth painting for image-based rendering applications”, Tech. Rep. CRL, Compaq Computer Corporation, Cambridge Research Lab., Dec. 1998.

[13] J. J. Koenderink, “Pictorial Relief”, *Phil. Trans. of the Roy. Soc.: Math., Phys, and Engineering Sciences*, 356(1740), pp. 1071-1086, 1998.

[14] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade and D. Fulk, “The digital michelangelo project: 3D scanning of large statues”, *ACM SIGGRAPH Proceedings*, pp. 131-144, 2000.

[15] S. K. Nayar, Y. Nakagawa, “Shape from focus”, *IEEE Trans. on PAMI*, vol.16, no.8, pp. 824-831, 1994.

[16] B. M. Oh, M. Chen, J. Dorsey, and F. Durand, “Image-based modeling and photo editing”, *ACM SIGGRAPH Proceedings*, pp. 433-442, 2001.

[17] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical recipes in C*, Cambridge University Press, pp. 83-89, 1988.

[18] H.-Y. Shum, M. Han, and R. Szeliski. “Interactive construction of 3D models from panoramic mosaics”, *IEEE Conf. on CVPR*, pp. 427-433, June 1998.

[19] B. J. Super, A. C. Bovik, “Shape from texture using local spectral moments”, *IEEE Trans on PAMI*, vol. 17, no. 4, pp. 333-343, 1995.

[20] R. Szeliski, “Fast surface interpolation using hierarchical basis functions”, *IEEE Trans. on PAMI*, vol. 12, no. 6, pp. 513-528, June 1990.

[21] R. Szeliski, H.-Y. Shum, “Motion estimation with quadtree splines”, *IEEE Trans. on PAMI*, vol. 18, no. 12, pp. 1199-1209, 1996.

[22] R. Szeliski and R. Zabih, “An experimental comparison of stereo algorithms”, *Int’l Workshop on Vision Algorithms*, pp. 1-19, September 1999.

[23] D. Terzopoulos, “Image analysis using multigrid relaxation methods”, *IEEE Trans. on PAMI*, vol. 8, no. 2, pp. 129-139, June 1986.

[24] D. Terzopoulos, “Regularization of inverse visual problems involving discontinuites”, *IEEE Trans. on PAMI*, vol. 8, no. 4, pp. 413-424, June 1986.

[25] A. N. Tikhonov, V. Y. Arsenin, *Solutions of Ill-Posed problems*, Washington, DC: Winston, 1977.

[26] W. Welch and A. Witkin, “Variational surface modeling”, *ACM SIGGRAPH Proceedings*, pp. 157-166, 1992.

[27] L. Williams, “3D paint”, *Proceedings of the Symposium on Interactive 3D Graphics Computer Graphics*, pp. 225-233, 1990.

[28] H. Yserentant, “On the multi-level splitting of finite element spaces”, *Numerische Mathematik*, vol. 49, pp. 379-412, 1986.

[29] “Single view modeling project website”, http:grail.cs.washington.eduprojectssvm.

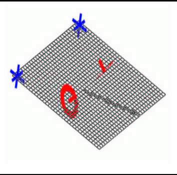
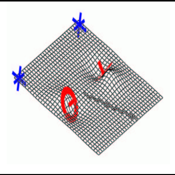
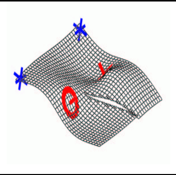
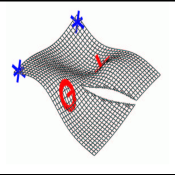
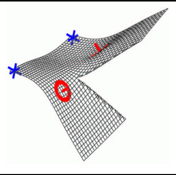
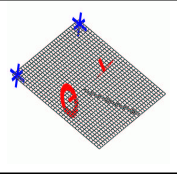
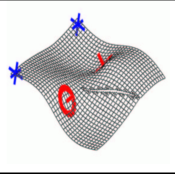
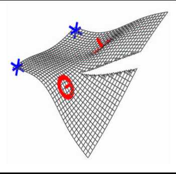
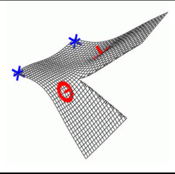
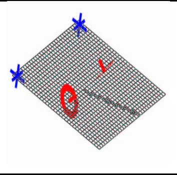
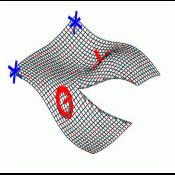
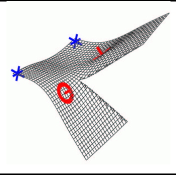
Methods	Iteration 0	Iteration 200	Iteration 1200	Iteration 2500	Iteration 9500
No hierarchical transformation					
Hierarchical transformation without continuity-based weighting					
Novel hierarchical transformation with continuity-based weighting					

Figure 4: Performance comparison of solving Eq. 11 by using no hierarchical transformation, traditional transformation, and our novel transformation in terms of number of iterations. The model has approximately 1400 grid points, and 4 constraints.

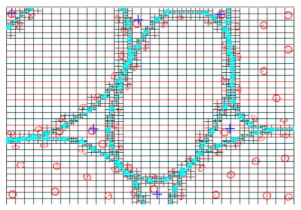
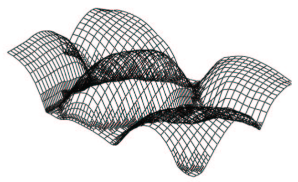
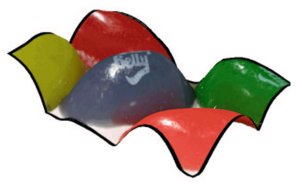
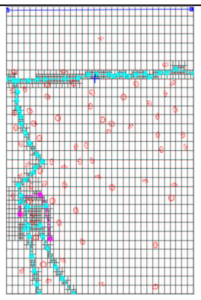
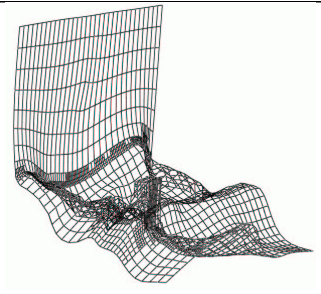
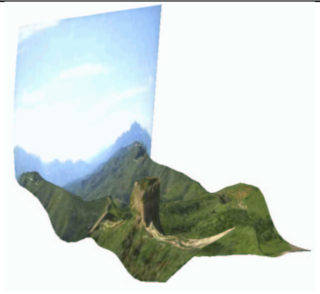
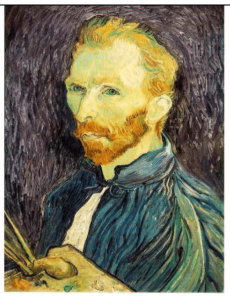
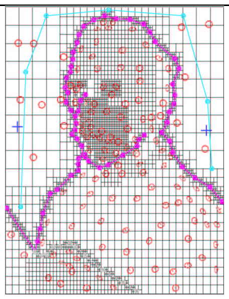
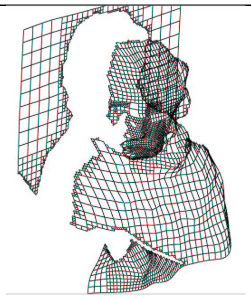
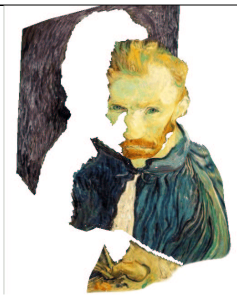
original image	constraints	3D wireframe	novel view
			
			
			

Figure 5: Examples of single view modeling on different scenes. From left to right, the columns show the original images, user-specified constraints on adaptive grids, 3D wireframe rendering, and textured rendering.