Semidefinite Programming Heuristics for Surface Reconstruction Ambiguities

Ady Ecker, Allan D. Jepson, and Kiriakos N. Kutulakos

Department of Computer Science, University of Toronto

Abstract. We consider the problem of reconstructing a smooth surface under constraints that have discrete ambiguities. These problems arise in areas such as shape from texture, shape from shading, photometric stereo and shape from defocus. While the problem is computationally hard, heuristics based on semidefinite programming may reveal the shape of the surface.

1 Introduction

An important problem in surface reconstruction is the handling of situations in which there are not enough constraints to uniquely determine the surface shape. In these under-constrained situations there are multiple interpretations of the surface that are consistent with the available constraints. The ambiguities can be continuous, such as unknown depth, or discrete, such as in/out reversal. In this paper we deal with constraints that have discrete ambiguities. We show that the problem of integrating a smooth surface under ambiguous constraints can be addressed with semidefinite programming (SDP).

SDP has been applied to a wide range of combinatorial optimization problems. For a general introduction to SDP see [1,2,3]. Recently, SDP-based approximation algorithms have been developed for several computer vision problems, such as image restoration [4,5], segmentation [4,6], graph matching [7,8,9] and finding correspondences in stereo [10]. Zhu and Shi [11] used SDP to solve in/out reversal ambiguities of surface patches in shape from shading.

In this paper we show that a similar mathematical formulation applies to other surface reconstruction problems. Our primary interest is resolving the inherent ambiguities in shape from texture. Similar ambiguities arise in two-light photometric stereo and shape from defocus.

The general approach starts by representing the surface as a spline, i.e. the shape is controlled by a set of continuous variables. Additional discrete variables are used to form the ambiguous constraints. A quadratic cost function measuring surface smoothness and constraint satisfaction is defined. The continuous variables are eliminated, leading to a quadratic cost function in the discrete variables only. An SDP relaxation embeds the discrete variables in a continuous high dimensional space. Finally, a rounding step sets the discrete variables and proposes a 3D shape.

The problems we deal with are larger than those considered by Zhu and Shi, and standard Goemans-Williamson random hyperplane rounding technique [12] will usually produce sub-optimal results. We describe several heuristics that can improve the solution considerably.

D. Forsyth, P. Torr, and A. Zisserman (Eds.): ECCV 2008, Part I, LNCS 5302, pp. 127–140, 2008.

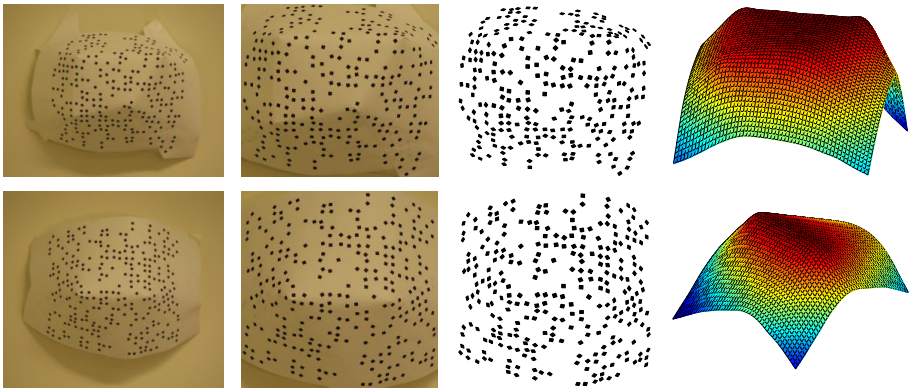

Fig. 1. Left to right: surfaces textured with squares, input images, parallelograms extracted from the images and the computed surfaces

2 Problem Formulation

We show below that several surface reconstruction problems can be written in the form

$$\underset{\boldsymbol{v},\boldsymbol{d}}{\operatorname{argmin}} \|\boldsymbol{A}\boldsymbol{v} - \boldsymbol{B}\boldsymbol{d}\|^2 \ , \tag{1}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are matrices, $\boldsymbol{d}\in\{-1,1\}^n$ is a vector of discrete decision variables, and $\boldsymbol{v}\in\mathbb{R}^m$ is a vector of continuous parameters that controls the surface. We represent the surface as a spline, i.e. a linear combination of basis functions

$$z(x,y) = \sum_{i=1}^{m} \boldsymbol{b}_i(x,y)\boldsymbol{v}_i \ . \tag{2}$$

The specific bases we use are described in the appendix. Controlling the surface by a relatively low dimensional vector of parameters $\boldsymbol{v}$ reduces the computational load and prevents over-fitting noisy constraints. We discuss several specific problems of this form next. Algorithms for solving (1) are discussed in Sect. 3.

2.1 Shape from Two-Fold Ambiguous Normals

Traditional shape from texture deals with estimating surface normals from the distortion of texture under projection. Under orthographic projection, normal estimates usually have a two-fold tilt ambiguity, because the projections of local planar patches with normals $(p,q,1)$ and $(-p,-q,1)$ are identical. We address the problem of estimating the shape of a surface given a large number of ambiguous normal estimates, as demonstrated in Fig. 1.

For sparse texture the problem is under-constrained, since the surface can be integrated for any choices of the normals. In addition, under orthographic projection there

is one global in/out reversal ambiguity, and a continuous ambiguity in absolute depth. However, if we make the assumption that the surface is smooth, we can identify the more probable shapes of the surface. In related work, Forsyth [13,14] proposed alternating between optimizing surface smoothness and selecting the normals. We show in Sect. 3 that by using a quadratic smoothness term the problem can be converted into an entirely discrete optimization problem.

Denote the partial derivatives of the surface by $p = \frac{dz}{dx}$, $q = \frac{dz}{dy}$. The texture observed at a specific image point, say (x_i, y_i), provides two choices for the surface derivatives, namely $(p_i, q_i)\boldsymbol{d}_i$, with $\boldsymbol{d}_i = \pm 1$. By differentiating (2) we can express the derivatives of the spline surface in terms of the vector $\boldsymbol{v}$. This provides two known row vectors $\boldsymbol{a}_{p_i}$ and $\boldsymbol{a}_{q_i}$ such that

$$(0, \ldots, 0, p_i, 0, \ldots, 0)\boldsymbol{d} = \boldsymbol{a}_{p_i} \cdot \boldsymbol{v} \quad , \quad (0, \ldots, 0, q_i, 0, \ldots, 0)\boldsymbol{d} = \boldsymbol{a}_{q_i} \cdot \boldsymbol{v} \ , \tag{3}$$

where $\boldsymbol{d}$ is a n-vector formed from the sign bits $\boldsymbol{d}_i$.

To regularize the surface we use a quadratic smoothness term. The smoothness term can be expressed in terms of the spline parameters using a matrix $\boldsymbol{E}$ such that $\|\boldsymbol{E}\boldsymbol{v}\|^2$ is the smoothness energy (see the appendix for more details). The smoothness energy is weighted with a regularization parameter λ that balances between the smoothness energy and the constraint error. Together, these terms can be written in the form of (1),

$$\underset{\boldsymbol{v},\boldsymbol{d}}{\operatorname{argmin}} \left\| \underbrace{\begin{bmatrix} \sqrt{\lambda}\boldsymbol{E} \\ \boldsymbol{A}' \end{bmatrix}}_{\boldsymbol{A}} \boldsymbol{v} - \underbrace{\begin{bmatrix} 0 \\ \boldsymbol{B}' \end{bmatrix}}_{\boldsymbol{B}} \boldsymbol{d} \right\|^2 . \tag{4}$$

Here $\boldsymbol{A}'$ and $\boldsymbol{B}'$ are matrices formed from the constraints in (3), with each constraint in (3) appearing as a single row in $\boldsymbol{A}'\boldsymbol{v} = \boldsymbol{B}'\boldsymbol{d}$.

2.2 Two-Light Photometric Stereo

In standard photometric stereo of a Lambertian surface, at least three light sources at known positions are used to determine the surface normals. For a given light source, the image brightness at a point constrains the corresponding surface normal to a circle on the unit sphere. Two light sources may limit the normal to two possibilities, which are the intersections of the two circles. The third light disambiguates the normal. In special cases, two lights are sufficient. This occurs when the two circles on the unit sphere touch at a point, or when one of the intersection points of the two circles is on the occluded half-hemisphere. Onn and Bruckstein [15] studied photometric stereo of Lambertian surfaces using two lights. Their method uses the points that are uniquely determined by two lights to divide the image into regions. Inside each region integrability is used to choose between the two possibilities of the normal. However, detecting the boundaries of these regions on a discrete grid is susceptible to errors, especially when the surface has discontinuous derivatives.

Our formulation for two-light photometric stereo avoids region detection and adds a surface smoothness prior. We assume knowledge of points (x, y) where we have two choices for the surface derivatives, (p_1, q_1) or (p_2, q_2). Onn and Bruckstein [15] derived the formulas for the possible derivatives in the Lambertian case. For other shading

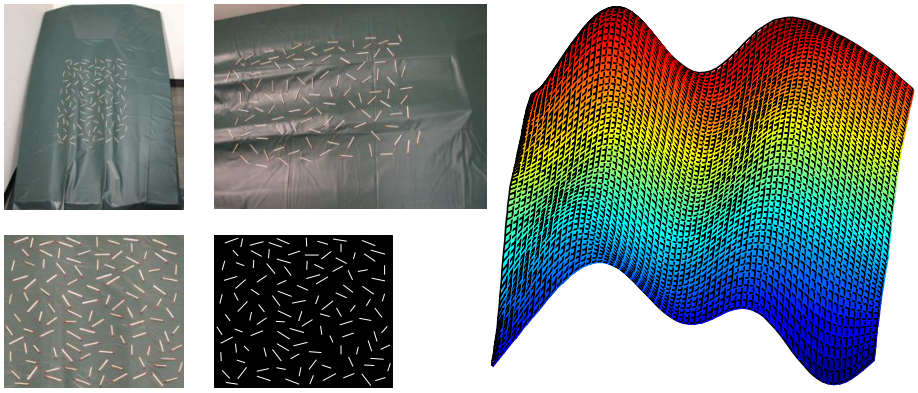

Fig. 2. Top row: setup of 112 matchsticks on a smooth surface. Bottom row: the input image (left) and extracted segments (right). Right: computed surface using 50 basis functions.

models these choices may be determined experimentally. However, more complicated shading models might require more images, since the corresponding two curves on the unit sphere might intersect more than twice. We don't deal with these cases here.

The two choices for the surface derivatives at point (x, y) can be expressed as functions of a sign bit $\boldsymbol{d}_{xy} = \pm 1$

$$\begin{aligned} \begin{pmatrix} p \\ q \end{pmatrix} &= \begin{pmatrix} p_{sum} \\ q_{sum} \end{pmatrix} + \begin{pmatrix} p_{diff} \\ q_{diff} \end{pmatrix} \boldsymbol{d}_{xy} \quad , \\ \begin{pmatrix} p_{sum} \\ q_{sum} \end{pmatrix} &= \frac{1}{2} \begin{pmatrix} p_1 + p_2 \\ q_1 + q_2 \end{pmatrix} \quad , \quad \begin{pmatrix} p_{diff} \\ q_{diff} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} p_1 - p_2 \\ q_1 - q_2 \end{pmatrix} \quad . \end{aligned} \tag{5}$$

As in the previous subsection, the spline derivatives can be written as $p = \boldsymbol{a}_p \cdot \boldsymbol{v}$, $q = \boldsymbol{a}_q \cdot \boldsymbol{v}$. Collecting the equations for all points and adding the smoothness term we arrive at

$$\underset{\boldsymbol{v},\boldsymbol{d}'}{\operatorname{argmin}} \left\| \begin{bmatrix} \sqrt{\lambda}\boldsymbol{E} \\ \boldsymbol{A}' \end{bmatrix} \boldsymbol{v} - \begin{bmatrix} 0 \\ \boldsymbol{B}' \end{bmatrix} \boldsymbol{d}' - \begin{pmatrix} 0 \\ \boldsymbol{b}' \end{pmatrix} \right\|^2 = \underset{\boldsymbol{v},\boldsymbol{d}'}{\operatorname{argmin}} \left\| \begin{bmatrix} \sqrt{\lambda}\boldsymbol{E} \\ \boldsymbol{A}' \end{bmatrix} \boldsymbol{v} - \begin{bmatrix} 0 & 0 \\ \boldsymbol{B}' & \boldsymbol{b}' \end{bmatrix} \begin{pmatrix} \boldsymbol{d}' \\ 1 \end{pmatrix} \right\|^2 \quad . \tag{6}$$

Here the nonzero entries of the matrix $\boldsymbol{B}'$ are made of p_{diff}, q_{diff}, and the vector $\boldsymbol{b}'$ is made of p_{sum}, q_{sum} according to (5). A standard transformation to bring (6) to the form of (1) is to solve for $\boldsymbol{d} = \begin{pmatrix} \boldsymbol{d}' \\ 1 \end{pmatrix}$. Note that the cost of a pair $(\boldsymbol{v}, \boldsymbol{d})$ in (1) equals the cost of $(-\boldsymbol{v}, -\boldsymbol{d})$. Thus, if after solving (6) the last coordinate of $\boldsymbol{d}$ is -1, we need to negate the solution.

2.3 Segments of Known Length

Assume a collection of segments of known 3D length (or pairs of features with known 3D distances) is detected on a smooth surface, as shown in Fig. 2. Given an orthographic

view, each segment can have a front/back reversal. Similar problems were considered by Naito and Rosenfeld [16] and Koenderink and van Doorn [17].

The depth difference of the segment's endpoints is constrained by

$$z_i - z_j = \boldsymbol{d}_{ij}\sqrt{l^2 - r_{ij}^2} = \Delta_{ij}\boldsymbol{d}_{ij} \ , \tag{7}$$

where $\boldsymbol{d}_{ij} = \pm 1$, l is the 3D length of the segment and r_{ij} is the observed length in the image. l, r_{ij} (and hence Δ_{ij}) are assumed to be known. By (2), the depth z_i at a point (x_i, y_i) is a linear combination of the spline bases at that point. That is, z_i can be expressed as $\boldsymbol{a}_i \cdot \boldsymbol{v}$, where $\boldsymbol{a}_i$ is a known row vector, and similarly for z_j. Each depth constraint can be written as

$$(\boldsymbol{a}_i - \boldsymbol{a}_j)\boldsymbol{v} = (0, \ldots, 0, \Delta_{ij}, 0, \ldots, 0)\boldsymbol{d} \ . \tag{8}$$

Collecting these equations over all constraints and adding the smoothness term can be written as in (4).

2.4 Shape from Defocus

In shape from defocus, depth is estimated by measuring blur level differences between multiple images taken with different focus and aperture camera settings. Here we consider a simple case involving just two images, both taken with the same focus setting. One image is obtained using a small aperture and is assumed to be sharp. The second image is taken with a large aperture and exhibits more blurring. The blur of a local region is measured as the standard deviation σ of a Gaussian that needs to be convolved with the sharp image to match the blurred image. The estimated $\sigma > 0$ for an image patch corresponds to a two-fold ambiguity for the depth, with one depth in front of the in-focus plane and the other behind. The formulas relating σ and the camera parameters to the two possible depths were developed by Pentland [18]. In Sect. 4 we experiment with a first-order simplified model that assumes the depth is proportional to $\pm\sigma$. This model falls naturally into the form (4), where now the matrix $\boldsymbol{A}'$ contains the depths of the spline bases vectors, and the matrix $\boldsymbol{B}'$ the estimated σ for a collection of points. Alternatively, Pentland's model could be expressed in the same general form using sums and differences as in (5) and (6).

Note that this example is meant only to demonstrate the formulation, as the practical use of this approach is rather limited. For points far from the in-focus plane the blur in the second image may be too heavy for us to reliably resolve σ. Moreover, for points in the second image which are nearly at the focused depth, it is again difficult to get an unbiased estimate of σ. As a consequence, useful information from the aperture change is only available over a narrow range of depths. In practice it is simpler to place the object on one side of the in-focus plane or take more images with different focus settings.

3 SDP Rounding Heuristics

We now turn into solving problems of the form (1). For any vector $\boldsymbol{d}$, the optimal $\boldsymbol{v}$ is

$$\boldsymbol{v} = \boldsymbol{A}^{+}\boldsymbol{B}\boldsymbol{d} \ , \tag{9}$$

where $\boldsymbol{A}^+$ is the pseudo-inverse of $\boldsymbol{A}$. Plugging $\boldsymbol{v}$ back into equation (1) we get $\|(\boldsymbol{A}\boldsymbol{A}^+\boldsymbol{B}-\boldsymbol{B})\boldsymbol{d}\|^2 = \boldsymbol{C}\bullet\boldsymbol{X}$, where $\boldsymbol{C} = (\boldsymbol{A}\boldsymbol{A}^+\boldsymbol{B}-\boldsymbol{B})^t(\boldsymbol{A}\boldsymbol{A}^+\boldsymbol{B}-\boldsymbol{B}) = \boldsymbol{B}^t(\boldsymbol{I}-\boldsymbol{A}\boldsymbol{A}^+)\boldsymbol{B}$, $\boldsymbol{X} = \boldsymbol{d}\boldsymbol{d}^t$, and $\bullet$ is the inner product of matrices ($\boldsymbol{C}\bullet\boldsymbol{X} = \sum \boldsymbol{C}_{ij}\boldsymbol{X}_{ij}$). Therefore, the problem is reduced into a combinatorial optimization problem of finding the discrete vector $\boldsymbol{d}\in\{-1,1\}^n$ which minimizes $\boldsymbol{C}\bullet(\boldsymbol{d}\boldsymbol{d}^t)$. Once $\boldsymbol{d}$ is found, $\boldsymbol{v}$ is given by (9) and the 3D shape is given by (2).

Unfortunately the general problem is NP-hard and difficult to approximate [19]. Semidefinite programming is widely used to find approximate solutions for problems of this kind. The standard SDP relaxation requires the matrix $\boldsymbol{X}$ to be symmetric positive semidefinite (instead of rank one) with ones on the main diagonal (since $\boldsymbol{d}_i^2 = 1$), i.e. solving

$$\underset{\boldsymbol{X}}{\operatorname{argmin}}\ \boldsymbol{C}\bullet\boldsymbol{X} \quad \text{s.t.}\ \ \boldsymbol{X}_{ii} = 1\ ,\ \ \boldsymbol{X}\succeq 0\ . \tag{10}$$

Since this problem is convex, the relaxation can be solved in polynomial time using an SDP solver. A discrete vector $\boldsymbol{d}$ is obtained from the continuous solution matrix $\boldsymbol{X}$ in a rounding phase. The Goemans-Williamson random hyperplane rounding scheme [12] uses the Cholesky factorization of the matrix $\boldsymbol{X}$, $\boldsymbol{X} = \boldsymbol{R}\boldsymbol{R}^t$. Let $\boldsymbol{u}_i$ denote the i-th row of $\boldsymbol{R}$. Since $\boldsymbol{X}_{ii} = \boldsymbol{u}_i\cdot\boldsymbol{u}_i^t = 1$, the rows of $\boldsymbol{R}$ can be viewed as an embedding of the decision variables into the unit sphere in $\mathbb{R}^n$ (this embedding is not unique). Rounding is done by picking a random hyperplane with normal $\boldsymbol{N}$ and setting $\boldsymbol{d}_i = sign(\boldsymbol{u}_i\cdot\boldsymbol{N})$.

For matrices $\boldsymbol{C}$ with nonnegative entries arising from the max-cut problem, this scheme provides a strong, provable expected approximation ratio of at least 0.878. However, this result does not apply directly to our problem for several reasons. First, their analysis is for the cost of the resulting cut, not the quadratic cost function itself. Secondly, our matrices may have negative elements. Thirdly, we are interested in the shape of the surface, not the number of correctly classified sign bits. A small number of misclassifications may have large influence on the shape or may not be visible at all. Note that it is possible to have different solutions with very different shapes but similar objective values. It is also possible that the correct surface is not the minimal solution (e.g. when the correct surface is not smooth, or when there are insufficient or noisy constraints). This depends on the instance of the problem.

In our setting, random hyperplane rounding typically requires a huge number of iterations to produce high quality results. We apply a series of heuristics for improvement:

1. Instead of picking plane normals from a uniform distribution on the sphere, we use the principle singular vectors of $\boldsymbol{R}$ (that correspond to largest singular values). For example, if we had to choose a single plane, a good choice for the normal would be the principle singular vector. Such "inertial" splitting methods have been previously used for other embeddings [20,21], but to our best knowledge not for the SDP embedding. To widen the choices of planes, we randomly pick normals as a weighted linear combination of the singular vectors that correspond to the k-largest singular values, i.e.

$$\boldsymbol{N} = \sum s_i\lambda_i\boldsymbol{w}_i\ , \tag{11}$$

where $s_i \sim N(0,1)$ and $\boldsymbol{w}_i$ is the singular vector corresponding to the singular value λ_i. Finding these singular vectors can be done efficiently by power-iteration methods.

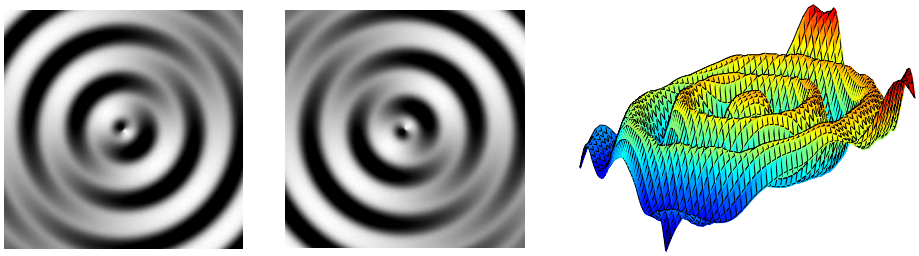

Fig. 3. Surface computed from a synthetic photometric stereo pair

2. Instead of making the decision based on a single normal, we randomly select a pair of normals $\boldsymbol{N}_1, \boldsymbol{N}_2$ according to (11), and perform a circular sweep for normals in the plane spanned by $\boldsymbol{N}_1, \boldsymbol{N}_2$. To do this, we project the points embedded in $\mathbb{R}^n$ on this plane (the first plane we check is the one spanned by the two principle singular vectors). Then we perform a circular sweep in this plane, as described in [22]. Basically, the sweep rotates a line through the origin that separates the points into two groups, and picks the partition with the lowest cost. We noticed that these angular sweeps can be made more efficient by careful bookkeeping similar to the Kernighan-Lin (K-L) algorithm [23]. Note that the cost of splitting n points in $\mathbb{R}^n$ based on a single normal is $O(n^2)$. However, scanning a series of n normals, where at each transition a single point moves to the other side of the sweep line, can also be carried out in $O(n^2)$.
3. The k-best results from the circular sweep phase are refined with the K-L algorithm [23,24,25,26]. This is a local search procedure that will clean up a small number of misplaced vertices. We terminate this algorithm early if no progress is made in 50 consecutive iterations [25]. The lowest cost solution among these trials is returned.

4 Two-Fold Ambiguity Results

Figure 1 demonstrates reconstruction from ambiguous normals. To simplify texture extraction we used square texture elements (see Sect. 5 for derivation of normals from parallelograms). The SDP solver we used is DSDP [27]. In Fig. 3 we computed a surface from a pair of synthetic images of a Lambertian surface using two-light photometric stereo. The two possibilities for the surface normal are computed on a 29×29 grid. Points having only a single possible normal and points in attached shadow were removed. The system had 425 discrete variables. Our program made the correct decision at each of these points. However, the average deviation from the true surface is 19% (note that two corners are in shadow).

Results for segments of known length are shown in Figs. 2 and 4. In Fig. 2, the image was taken from a distance of about 10m with a zoom lens to approximate orthographic projection. As suggested by Naito and Rosenfeld [16], the 3D length of the segments is estimated as the maximum over the 2D lengths of all segments in the image. Figure 4 shows 1521 randomly oriented line segments tangent to a synthetic surface (top and

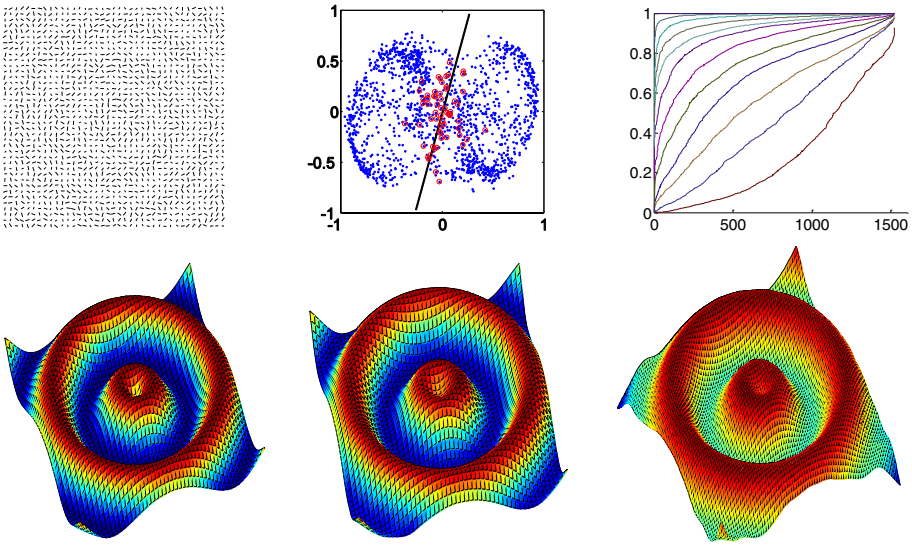

Fig. 4. Top left: segments of known and equal length on synthetic surface. Top center: projection of the SDP embedding on the subspace of the first two principle components. Misclassified points with respect to the ground truth are circled. Top right: plot of the squared-length of the projections of the embedded points on the first principle subspaces (see text). Bottom left: original surface. Bottom middle: output of the program. Bottom right: output using random hyperplane rounding.

bottom left). The top-center plot shows the projection of the SDP embedding on the subspace of the first two principle components. The splitting line is the lowest energy circular cut for this projection. To check the appropriateness of the projection on a low-dimensional subspace we projected the SDP embedding on the subspaces spanned by the first $1, 2, 3, \ldots$ principle vectors. The top-right plot is the squared-length of these projections, where the 1521 values are sorted. The lowest curve is the distribution of magnitudes of the projection on the subspace of the first principle vector; the second curve is the distribution of magnitudes for the projection on the subspace of the first two principle vectors, etc. It is evident that the points were embedded near a low dimensional subspace, and this is used for more efficient rounding. The solution (bottom-center) was computed using 300 spline bases, 1000 circular sweeps on random planes, and K-L runs on the best 100 vectors. The program made 15 wrong decisions, and the average height deviation from the truth surface is 1%. In comparison, a run of the Goemans-Williamson random hyperplane rounding (bottom right) with 10^4 trials produced 82 misclassifications, with 2.6% average height deviation. While these numbers depend very much on the particular instance, this example shows that the SDP approach to ambiguities can deal with much larger instances than demonstrated by Forsyth [13].

Figure 5 demonstrates shape reconstruction from defocus using two images taken with a narrow and a wide aperture. The camera is focused at the middle of the shape. The blur level is estimated over a grid of 15×60 points. To estimate the blur level, we convolved a window of the narrow-aperture image around each grid point with Gaussians of various sizes and pick the size that best matches the blurred image. To resist

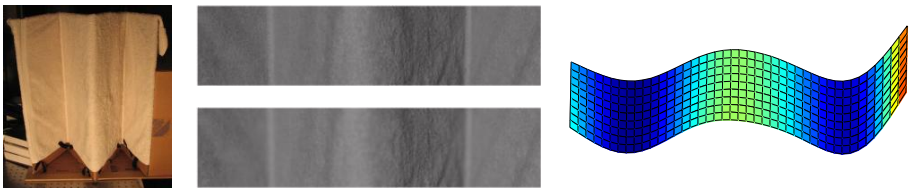

Fig. 5. Shape reconstruction from defocus. Left: the imaged surface. Middle: the sharp (narrow aperture) and blurred (wide aperture) input images. Right: the computed surface.

noise, we pick the median of 9 windows near a grid point. Points where the estimated blur level is very low or very high were removed since, for these points, the relation between the estimated defocus and the depth is inaccurate.

5 Four-Fold Ambiguities

The previous sections looked at problems where each decision had two options. In this section we demonstrate an extension to discrete ambiguities with four options. As a model, we look at a shape from texture problem. Suppose a collection of similar triangles are scattered on a smooth surface viewed orthographically. All triangles are scaled versions of a known triangle. In addition, we assume that one edge on each triangle can be identified. For instance, if the texture elements are rectangles, as in Fig. 6, we can identify the diagonal. This information leads to four-fold ambiguity since there are two ways to match the image segments with the edges of the known triangle.

Algebraically, let (x_i, y_i, z_i), $i = 1, 2, 3$, be the three vertices of a triangle in the image. Denote $dx_{ij} = x_i - x_j$, $dy_{ij} = y_i - y_j$, $dz_{ij} = z_i - z_j = p \cdot dx_{ij} + q \cdot dy_{ij}$, where p, q are the slopes of the triangle's plane. Since the triangle is similar to the model triangle, the 3D length ratios are known

$$r_1 = \frac{dx_{12}^2 + dy_{12}^2 + dz_{12}^2}{dx_{13}^2 + dy_{13}^2 + dz_{13}^2} \quad , \quad r_2 = \frac{dx_{23}^2 + dy_{23}^2 + dz_{23}^2}{dx_{13}^2 + dy_{13}^2 + dz_{13}^2} \ . \tag{12}$$

This leads to two quadratic equations in p, q. Simple manipulations lead to a quadratic equation in q^2 that can be solved for the positive root (details are omitted). Switching between r_1,r_2 gives another solution. In the general case, the four solutions are of the form $\pm(p_1, q_1)$ and $\pm(p_2, q_2)$.

In the SDP literature, the max-k-cut problem was studied by Frieze and Jerrum [28] and de Klerk, Pasechnik, and Warners [29]. While an ideal encoding requires two bits to encode four possibilities, their encoding uses four bits: a single indicator bit set to 1 and the rest 0. Since the matrix $\boldsymbol{X}$ has $O(n^2)$ entries, redundant encoding makes the SDP problem 4 times larger, which is a significant factor for current SDP solvers. There is a natural encoding of this problem with two sign bits for each constraint using average and offset vectors similar to the sums and difference vectors of Sect. 2.2. However, we found that if the two bits are completely independent, the rounding phase becomes more difficult and results get worse. Instead, our encoding uses two variables d_1, d_2 for each

triangle that ideally would take the values $-1, 0, 1$. We add the constraint $d_1 \cdot d_2 = 0$ so that only one decision variable is active at a time

$$p = d_1 p_1 + d_2 p_2 \quad , \quad q = d_1 q_1 + d_2 q_2 \quad , \quad d_1 \cdot d_2 = 0 \ . \tag{13}$$

By expressing p, q for each triangle using the spline parameters and adding the smoothness term, we arrive at a system of the form (1) in $2n$ discrete variables. The SDP relaxation is modified to

$$\underset{\boldsymbol{X}}{\text{argmin}} \ \boldsymbol{C} \bullet \boldsymbol{X} \quad \text{s.t.} \ \boldsymbol{X}_{2i-1,2i-1} + \boldsymbol{X}_{2i,2i} = 1 \ , \ \boldsymbol{X}_{2i-1,2i} = 0 \ , \ \boldsymbol{X} \succeq 0 \ . \tag{14}$$

After solving the SDP problem (14) for $\boldsymbol{X}$, Cholesky factorization $\boldsymbol{X} = \boldsymbol{R}\boldsymbol{R}^t$ embeds the decision variables into a sphere in $\mathbb{R}^{2n}$. Let $\boldsymbol{u}_i$ denote the i-th row of $\boldsymbol{R}$, so that $\boldsymbol{X}_{ij} = \boldsymbol{u}_i \cdot \boldsymbol{u}_j^t$. For each decision there are two orthogonal vectors, $\boldsymbol{u}_{2i-1}$,$\boldsymbol{u}_{2i}$, such that $\|\boldsymbol{u}_{2i-1}\|^2 + \|\boldsymbol{u}_{2i}\|^2 = 1$. The rounding phase has to decide which of the two vectors is active, and round the active variable to either 1 or -1. In the ideal case, the vectors associated with the inactive variables would concentrate near the origin. While concentration can be observed, deciding which variable is active by picking the longer vector of each pair is not powerful enough. We execute the following heuristics:

1. A column of the matrix $\boldsymbol{X}$ provides an indication about the orthogonality between a row vector $\boldsymbol{u}_i$ of $\boldsymbol{R}$ to the other rows. We multiply $\boldsymbol{u}_i$ by α_i, the magnitude of the i-th column of $\boldsymbol{X}$. This has the effect of moving vectors orthogonal to the rest (in particular close to 0) towards the origin. In addition, the largest singular vectors of the modified vectors become less affected by the inactive variables.
2. The vectors $\alpha_i \boldsymbol{u}_i$ are projected on a plane as in (11), and a line is swept circularly. For each pair, the point with the largest projection on the sweep line is considered active, and the sign of the projection is the rounded value (events in this circular sweep occur at angles where the projections of the points on the sweep line are equal in absolute value). We repeat this step for different planes and store the best circular cut for each plane.
3. For every variable, we estimate the probability p_i it is inactive as the percentage of best cuts where it was inactive. Clearly, $p_{2i-1} + p_{2i} = 1$. We modify the diagonal of the matrix $\boldsymbol{C}$ to reflect this knowledge by setting $\boldsymbol{C}_{ii} = \boldsymbol{C}_{ii} + \mu p_i$ (μ is a tuning parameter). The SDP is solved a second time with the modified matrix $\boldsymbol{C}$. Due to the constraint $\boldsymbol{X}_{2i-1,2i-1} + \boldsymbol{X}_{2i,2i} = 1$, vectors which are believed to be inactive are pushed towards the origin.
4. We repeat steps 1,2 with the modified $\boldsymbol{C}$, keep the best solutions and run the K-L algorithm as a final step. The modification of the K-L algorithm to four possibilities is straightforward.

The method is demonstrated in Figs. 6 and 7. In Fig. 6, A collection of 415 similar rectangles at random orientations is overlaid on a 3D surface. Our program uses triangles made of the diagonal and two sides of each rectangle. Only the proportions of the model triangle are assumed known. The circular sweep in the plane of the first two principle components makes 148 errors in the first round (out of 830), and after modifying the $\boldsymbol{C}$ matrix, makes 114 errors in the second round. Note that a large number of

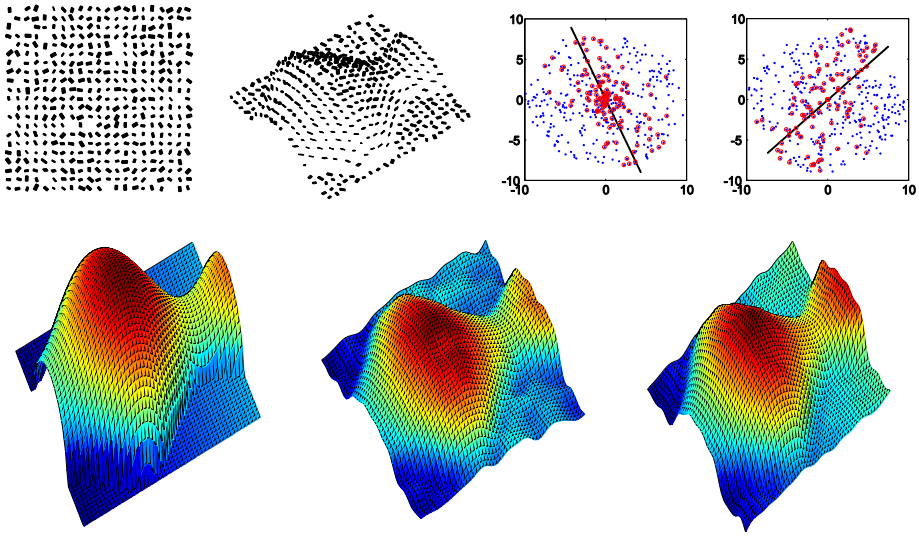

Fig. 6. Reconstruction from similar rectangles. Top left: the input image is made of similar rectangles overlaid on a 3D surface. Top right: the circular cuts of the projections on the plane of the two principle directions for the first and second SDP embeddings. Bottom Left: original surface. Bottom middle: output of the program. Bottom right: spline surface using the ground truth decision vector.

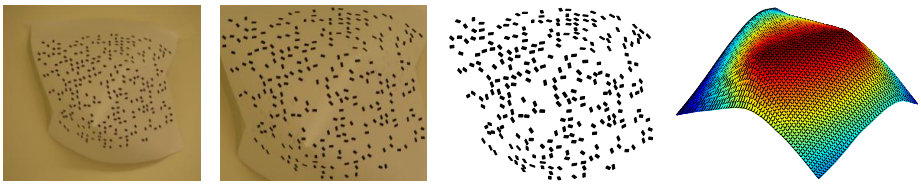

Fig. 7. Left to right: surface textured with rectangles in ratio 1:2, input image, parallelograms extracted from the image and the computed surface

inactive variables are concentrated too close to the origin than could be distinguished in this figure. The final computed surface (bottom middle) makes 80 errors. The average height difference between the original and computed surfaces is 5%. Wiggles in the computed surface arise because only 300 basis are used. Similar wiggles occur with a spline that uses the ground truth decision vector (bottom right). In this example, the cost function of the computed surface is lower than the cost of the ground truth spline.

6 Conclusions

Several depth cues, such as texture, shading and defocus, are inherently ambiguous at the local level. In this paper we examined integration of discrete constraints arising from these ambiguities with continuous objectives like surface smoothness. Problems

of this form involve both continuous and discrete variables, and can be transformed into an entirely discrete optimization problem, which is computationally hard. For an approximate solution we used SDP relaxation. We improved the rounding phase using a combination of heuristics, namely projection on planes in the subspace of the largest principle components, efficient circular sweep, and the K-L algorithm. These general heuristics were shown to be useful in our setting, and are potentially useful for other SDP applications as well.

Compared to other energy minimization approaches, such as belief propagation or graph cuts, our approach is global. Any discrete decision directly influences the cost of the entire surface. The optimization is not based on local neighborhoods. Using a global approach is sometimes necessary, since knowledge of a label at a point could say little about a neighboring label. Another important property of SDP is that there is no need for initialization. However, if a good starting vector $\boldsymbol{d}$ is available, it can be exploited by performing the rounding phase on a linear combination of $\boldsymbol{X}$ and $\boldsymbol{d} \cdot \boldsymbol{d}^t$ [30].

Our focus has been on developing a general framework for solving problems of the form (1), involving ambiguous discrete constraints. In practice, after binary decisions are made, the surface can be re-integrated using more robust integration methods. While the use of smoothness was demonstrated to resolve certain shapes, for many surfaces smoothness alone is insufficient and additional unambiguous constraints are required. A natural extension to the approach presented here would be to replace the spline with a shape basis, e.g. for particular shapes such as faces [31]. The general form of (1) allows for many other variations, such as adding linear constraints (e.g. specifying depths or normals at specific points [32]), or using shading information to disambiguate some normals [33].

References

1. Helmberg, C.: Semidefinite programming for combinatorial optimization. Technical Report ZIB-Report ZR-00-34, TU Berlin (2000)
2. Laurent, M., Rendl, F.: Semidefinite programming and integer programming. In: Handbook on Discrete Optimization, pp. 393–514. Elsevier, Amsterdam (2005)
3. Todd, M.J.: Semidefinite optimization. Acta Numerica 10, 515–560 (2001)
4. Keuchel, J., Schnorr, C., Schellewald, C., Cremers, D.: Binary partitioning, perceptual grouping, and restoration with semidefinite programming. PAMI 25(11), 1364–1379 (2003)
5. Keuchel, J.: Multiclass image labeling with semidefinite programming. In: ECCV 2006, pp. 454–467 (2006)
6. Carl Olsson, A.E., Kahl, F.: Solving large scale binary quadratic problems: Spectral methods vs. semidefinite programming. In: CVPR 2007, pp. 1–8 (2007)
7. Bai, X., Yu, H., Hancock, E.: Graph matching using spectral embedding and semidefinite programming. In: BMVC 2004, pp. 297–307 (2004)
8. Yu, H., Hancock, E.R.: Graph seriation using semi-definite programming. In: Brun, L., Vento, M. (eds.) GbRPR 2005. LNCS, vol. 3434, pp. 63–71. Springer, Heidelberg (2005)
9. Schellewald, C., Schnörr, C.: Probabilistic subgraph matching based on convex relaxation. In: Rangarajan, A., Vemuri, B.C., Yuille, A.L. (eds.) EMMCVPR 2005. LNCS, vol. 3757, pp. 171–186. Springer, Heidelberg (2005)
10. Torr, P.: Solving markov random fields using semi definite programming. In: Proc. Ninth International Workshop on Artificial Intelligence and Statistics (2003)

11. Zhu, Q., Shi, J.: Shape from shading: Recognizing the mountains through a global view. In: Proc. CVPR 2006, pp. 1839–1846 (2006)
12. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM 42(6), 1115–1145 (1995)
13. Forsyth, D.: Shape from texture and integrability. In: Proc. ICCV 2001, pp. 447–452 (2001)
14. Forsyth, D.: Shape from texture without boundaries. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2350, pp. 225–239. Springer, Heidelberg (2002)
15. Onn, R., Bruckstein, A.: Integrability disambiguates surface recovery in two-image photometric stereo. Int. J. Comput. Vision 5(1), 105–113 (1990)
16. Naito, S., Rosenfeld, A.: Shape from random planar features. CVGIP 42(3), 345–370 (1988)
17. Koenderink, J., van Doorn, A.: Shape from chebyshev nets. In: Burkhardt, H.-J., Neumann, B. (eds.) ECCV 1998. LNCS, vol. 1406, pp. 215–225. Springer, Heidelberg (1998)
18. Pentland, A.P.: A new sense for depth of field. PAMI 9(4), 523–531 (1987)
19. Arora, S., Berger, E., Hazan, E., Kindler, G., Safra, M.: On non-approximability for quadratic programs. In: Proc. FOCS 2005, pp. 206–215 (2005)
20. Chan, T.F., Gilbert, J.R., Teng, S.H.: Geometric spectral partitioning. Technical Report Tech. Report CSL-94-15, Xerox PARC (1995)
21. Fjallstrom, P.O.: Algorithms for graph partitioning: A survey. Linkoping Electronic Articles in Computer and Information Science 3(10) (1998)
22. Burer, S., Monteiro, R.D.C., Zhang, Y.: Rank-two relaxation heuristics for max-cut and other binary quadratic programs. SIAM J. on Optimization 12(2), 503–521 (2002)
23. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. The Bell system technical journal 49(1), 291–307 (1970)
24. Fiduccia, C., Mattheyses, R.: A linear-time heuristic for improving network partitions. In: Proc. 19th Design Automation Conference, pp. 175–181 (1982)
25. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Sci. Comput. 20(1), 359–392 (1998)
26. Krishnan, K., Mitchell, J.E.: A semidefinite programming based polyhedral cut and price approach for the maxcut problem. Comp. Optim. Appl. 33(1), 51–71 (2006)
27. Benson, S.J., Ye, Y.: Algorithm 875: DSDP5: Software for semidefinite programming. ACM Trans. Math. Software 34(3) (2008)
28. Frieze, A., Jerrum, M.: Improved approximation algorithms for maxk-cut and max bisection. Algorithmica 18(1), 67–81 (1997)
29. de Klerk, E., Pasechnik, D.V., Warners, J.P.: On approximate graph colouring and max-k-cut algorithms based on the theta-function. J. Comb. Optim. 8(3), 267–294 (2004)
30. Rendl, F., Rinaldi, G., Wiegele, A.: Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. Technical report, Alpen-Adria-Universitt Klagenfurt, Inst. f. Mathematik (2007)
31. Atick, J.J., Griffin, P.A., Redlich, A.N.: Statistical approach to shape from shading: Reconstruction of three-dimensional face surfaces from single two-dimensional images. Neural Computation 8(6), 1321–1340 (1996)
32. Zhang, L., Dugas-Phocion, G., Samson, J.S., Seitz, S.M.: Single view modeling of free-form scenes. In: Proc. CVPR 2001, pp. 990–997 (2001)
33. White, R., Forsyth, D.: Combining cues: Shape from shading and texture. In: Proc. CVPR 2006, pp. 1809–1816 (2006)

Appendix

In this section we describe in more detail the spline (2) and smoothness energy used in our implementation. These were chosen mainly for the sake of simplicity, and other splines and energies (e.g. thin-plate spline) could be used. To simplify notation assume the image is square. We used tensor-product spline bases

$$\boldsymbol{b}_{ij}(x,y) = \boldsymbol{b}_i(x)\boldsymbol{b}_j(y) \ , \tag{15}$$

where $\boldsymbol{b}_i, \boldsymbol{b}_j$ are singular vectors associated with small singular values of the matrix

$$\boldsymbol{D} = \begin{bmatrix} 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & & \cdots & 1 & -2 & 1 \end{bmatrix} \ . \tag{16}$$

For the smoothness energy we used the sum of squared second derivatives over the image. The energy of a basis function of the form (15), with $\|\boldsymbol{b}_i\| = \|\boldsymbol{b}_j\| = 1$, is

$$e_{ij}^2 = \|\boldsymbol{D}\boldsymbol{b}_i\|^2 + \|\boldsymbol{D}\boldsymbol{b}_j\|^2 \approx \iint \left(\frac{d^2\boldsymbol{b}_{ij}}{dx^2}\right)^2 + \left(\frac{d^2\boldsymbol{b}_{ij}}{dy^2}\right)^2 dx\, dy \ . \tag{17}$$

Note that for tensor-product splines we need to integrate only one-dimensional functions. The vectors $\boldsymbol{D}\boldsymbol{b}_i$ are proportional to the left singular vectors of $\boldsymbol{D}$ and hence orthogonal. Since the basis functions $\boldsymbol{b}_i$ are orthogonal, as are $\boldsymbol{D}\boldsymbol{b}_i$, the smoothness of a spline governed by $\boldsymbol{v}$ can be written as $\|\boldsymbol{E}\boldsymbol{v}\|^2$, where $\boldsymbol{E}$ is a diagonal matrix made of the elements e_{ij} for each basis used. The advantage of this approach over Fourier basis is that the basis vectors are not cyclic.