# 3D Surface Reconstruction Using Graph Cuts with Surface Constraints*

Son Tran and Larry Davis

Dept. of Computer Science, University of Maryland,
College Park, MD 20742, USA
{sontran, lsd}@cs.umd.edu

**Abstract.** We describe a graph cut algorithm to recover the 3D object surface using both silhouette and foreground color information. The graph cut algorithm is used for optimization on a color consistency field. Constraints are added to improve its performance. These constraints are a set of predetermined locations that the true surface of the object is likely to pass through. They are used to preserve protrusions and to pursue concavities respectively in the first and the second phase of the algorithm. We also introduce a method for dealing with silhouette uncertainties arising from background subtraction on real data. We test the approach on synthetic data with different numbers of views (8, 16, 32, 64) and on a real image set containing 30 views of a toy squirrel.

## 1 Introduction

We consider the problem of reconstructing the 3D surface of an object from a set of images taken from calibrated viewpoints. The information exploited includes the object's silhouettes and its foreground color or texture. 3D shape recovery using silhouettes constitutes a major line of research in computer vision, the shape-from-silhouette approach. In methods employing silhouettes only (see e.g. [1]), voxels in a volume are carved away until their projected images are consistent with the set of silhouettes. The resulting object is the visual hull. In general, the visual hull can be represented in other forms such as bounding edges ([2]), and can be reconstructed in a number of different ways. The main drawback of visual hulls is that they are unable to capture concavities on the object surface ([3]).

A 3D surface can also be reconstructed using color or texture consistency between different views. Stereo techniques find the best pixel matching between pairs of views and construct disparity maps which represent (partial) shapes. Combining from multiple stereo maps has been studied, but is quite complicated ([4]). Space carving ([5]) and recent surface evolution methods (e.g. [6], [7]) use a more general consistency check among multiple views.

The combination of both silhouettes and foreground color to reconstruct an object's surface has been studied in a number of recent papers ([7], [8], [9]).

* This work is supported by the NSF grant IIS-0325715 entitled ITR: New Technology for the Capture, Analysis and Visualization of Human Movement.

A. Leonardis, H. Bischof, and A. Pinz (Eds.): ECCV 2006, Part II, LNCS 3952, pp. 219–231, 2006.

Our work is motivated by [8] and [10] where the graph cut algorithm serves as the underlying 3D discrete optimization tool. The near global optimality properties of the graph cut algorithm are discussed in [11]. As noted in [8] and in other works however, the graph cut algorithm usually prefers shorter cuts, which leads to protrusive parts of the object surface being cut off. We overcome this limitation with a two-phase procedure. In the first phase (phase I), protrusions are protected during the optimization by forcing the solution to pass close to a set of predetermined surface points called "constraint points". In the second phase (phase II), concavities on the object surface are aggressively pursued. Silhouette uncertainties, which are important in practice but have been ignored in previous research ([8], [9], ...) are also taken into account.

### 1.1 Related Works

The application of reliable surface points to constrain the reconstruction of a surface appears in a number of recent papers ([2], [7], [9], ...). Isidoro et al ([7]) refine the shape and texture map with an EM-like procedure; the evolution of the shape at each iteration is anchored around a set of locations called frontier points. Cheung et al ([2]) use another set of points called color surface points to align multiple visual hulls constructed at different times to obtain a closer approximation to the object's true surface. Usually, these points have no special patterns on the surface. In some cases, however, they might lie on continuous curves such as the rims in [9], where each (smooth and closed) rim is a contour generator. The mesh of rims can be used to partition the surface into local patches. Surface estimation is then performed individually for each patch, with some interaction to ensure certain properties such as smoothness.

The identification of these surface points is typically based on the silhouettes and color/photo consistency. A frontier point in [7] is the point with lowest texture back-projection error among those on the evolving surface that project onto a single silhouette point. Frontier points are recomputed at each iteration. The rims in [9] are built with a rim mesh algorithm. In order for the mesh to exist, certain assumptions have to be made, the most limiting one being no self-occlusion. In [2], the colored surface points are searched for along bounding edges which collectively represent the surface of the object.

Surface reconstruction methods that use color or texture such as [2], [8], [7], [9] and most stereo algorithms involve optimization. The original space carving algorithm ([5]) used a simple greedy algorithm. Other examples of local methods include stochastic search ([7]) and, recently, surface evolution using level sets or PDEs (e.g. [6]). Local techniques are often sensitive to initialization and local minimum. Here, we use the 3D graph cut algorithm which is more global in scope ([11]). It was applied in [3] to solve the occupancy problem and in [10] for 3D image segmentation. The work described in [7] has similar motivation to ours: developing a constrained graph cut solution to object surface recovery. Their constraints are based on the rim mesh mentioned above. Multiple interconnected sub-graphs are built, with one for each rim mesh face. Our constraint points are not required to form rims and we use only one graph; our formulation is most

similar to [8], which is the departure point for our research. Section 2 describes the basic steps of the formulation from [8].

## 2 Volumetric Graph Cuts

Following [8], we first construct the visual hull $V$ from the set of $N$ image silhouettes, denoted $\{Sil_i\}$. $V$ is used as the initial approximation to the object shape. A photo consistency field for all voxels $v \in V$ is constructed and used as the graph on which a graph cut optimization is performed. Visibility for a voxel $v \in V$, $Vis(v)$, is approximated with the visibility of the closest voxel to $v$ on the surface $S_{out}$ of $V$. The consistency score for $v$, $\rho(v)$ is the weighted normalized cross correlation (NCC) between the pairs of local image patches that $v$ projects to in the different views:

$$\rho(v) = \sum_{C_i, C_j \in Vis(v)} w(pos(C_i, C_j)) NCC(p(C_i, v), p(C_j, v)) \quad (1)$$

where $w(pos(C_i, C_j)$ is a weight depending on the relative position of the two camera centers $C_i$ and $C_j$ (small when the difference between the viewing angles of the $i$-th and $j$-th cameras is large and vice versa); $p(C_i, v)$ is the local image patch around the image of $v$ in the $i$-th image $I_i$ .

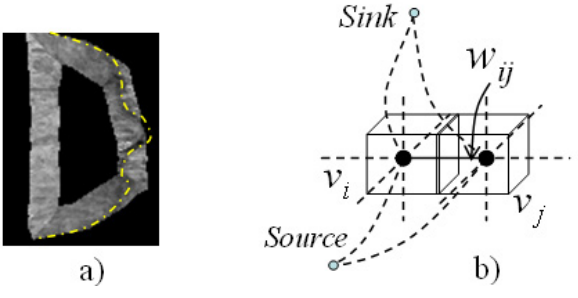


**Fig. 1.** a) a slice of the photo consistency field, yellow line denotes the true surface. b) Nodes and edges in the graph $G$.

If the surface, $S_{out}$, of the visual hull, $V$, is not far from the actual surface $S^*$, then with consistency computed this way, voxels that lie on $S^*$ would have smallest $\rho$ values (Figure 1.a). Therefore, finding $S^*$ can be formulated as an energy minimization problem, where the energy is defined as

$$E(S) = \iint_S \rho(x) dA \quad (2)$$

A graph cut algorithm can be used to solve this problem in a manner similar to [12] and [10]. Each voxel is a node in the graph, $G$, with a 6-neighbor system for edges. The weight for the edge between voxel (node) $v_i$ and $v_j$ is defined as

$w(v_i, v_j) = 4/3\pi h^2(\rho(v_i) + \rho(v_j))/2$ (Figure 1.b), where $h$ is the voxel size. $S_{out}$ and $S_{in}$ – the surface inside $V$ at a distance $d$ from $S_{out}$ – form an enclosing volume in which $S^*$ is assumed to lie. Similar to [12] and [9], every voxel $v \in S_{in}(S_{out})$ is connected to the $Sink$ ($Source$) node through an edge with very high weight. With the graph $G$ constructed this way, the graph cut algorithm is then applied to find $S^*$.

## 3 Graph Cut with Surface Point Constraints

As mentioned in [8], the above procedure suffers from the limitation that the graph cut algorithm prefers shorter cuts. This produces inaccurate surfaces at protrusions, which are often cut off ([8]). We address this problem by constraining the solution cut to pass through certain surface points. First we show how to identify those points. Next, we show how to enforce the solution cut to pass through or close to them. Finally, methods for dealing with silhouette uncertainty are included.

### 3.1 Constraint on Surface Points

Assume, to begin with, that the set of silhouettes has absolute locational certainty. Every ray $(C_i, p_i^j)$ from a camera center $C_i$ through a point $p_i^j$ on the silhouette $Sil_i$ has to touch the object surface at at least one point $P$ ([2], [9]) (Figure 2.a). In [2], the authors search for $P$ along this ray. We, additionally, take into account the discretization of the silhouette and make the search region not a single ray $(C_i, p_i^j)$ but a surface patch $s \subset S_{out}$ where $s = \{v \mid v \in S_{out}$ and $v$ *projects to* $p_i^j$ *through* $C_i\}$. Since every voxel on $S_{out}$ has to project onto some point on some silhouette $\{Sil_i\}$, the union of all $s$ is $S_{out}$. Therefore, $S_{out}$ is completely accounted for when we search for all $P$'s. In [7], the authors also use the projection from object space to silhouettes to find the search regions for their set of constraint points. However, these regions, and therefore the resulting constraint points, lie on an evolving surface and have to be recomputed at each step of their iterative procedure. Here, the determination of $P$ is done only once and is based on $S_{out}$, the surface of the original visual hull.

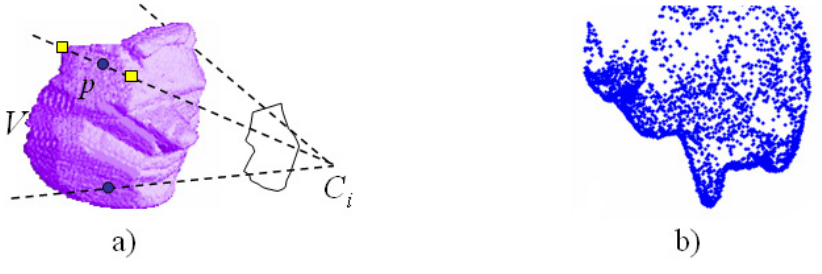


**Fig. 2.** a) Rays touch $V$'s surface at $p$, b) Example of the set of constraint points, **P**

Let $\mathbf{P}$ denotes the set of all such $P$'s. To identify the location of each $P \in \mathbf{P}$ within its corresponding search region, we use color or texture information from the image foreground. Ideally, the images of such voxels should have zero consistency score $\rho$ or zero color variance. Practically, they are voxels whose projections have the lowest $\rho$ within a search region. Figure 2.b shows an example of the constraint points, $\mathbf{P}$, for the synthetic face that is used in the experiments in section 5. Note that their distribution is quite general and they do not obviously form rims. This creates difficulties for approaches that assume exact silhouette information such as [9] and [13] . By marking which sub-regions of $S_{out}$ are produced by which camera, $\mathbf{P}$ can be constructed in time linear in the number of voxels in $S_{out}$.

If the average number of points on a silhouette is $n_s$, then the number of points in $\mathbf{P}$ is $N.n_s$. Many of them lie on protrusive parts of the object surface. In general, $\mathbf{P}$ provides a large set of constraints for the graph cut optimization.

### 3.2 Graph Cut with Surface Constraint Points

Given the set of surface constraint voxels, $\mathbf{P}$, we want to construct a cut that passes through every voxel $p \in \mathbf{P}$. Unfortunately, it is difficult to introduce such constraints directly into the 3D graph cut algorithm. Instead, we adopt an indirect approach by blocking the solution surface from cutting a continuous region that connects $p$ and $S_{in}$. Figure 3.a illustrates the blocking region: it is a curve $bl(p)$ from the surface point $p \in \mathbf{P}$ to $S_{in}$. More generally, a blocking region can be represented as a blurred volume around the blocking curves using a Gaussian blurring function. We next describe how to construct $bl(p)$.

Let $D(S)$ and $\nabla D(S)$ denote the 3D distance transform of a surface $S$ and the gradient of the distance transform, respectively. For each $p \in \mathbf{P}$, the corresponding curve $bl(p)$ is constructed using $\nabla D(S_{out})$ and $\nabla D(S_{in})$ as follows. First, starting from $p$, we move along $\nabla D(S_{out})$ for a small distance $l$. Second, we follow $-\nabla D(S_{in})$ until $S_{in}$ is met. Points are added into $bl(p)$ as we move. To avoid redundancy, if a point is met that has been added to some previously constructed $bl(p')$, we stop collecting points for $bl(p)$. This procedure is carried out for all points in $\mathbf{P}$.


**Fig. 3.** a) Blocking regions (curves). b) Locational uncertainties (gray areas) of the contour extracted from a difference image.

$D(S_{out})$ can be considered as an implicit shape representation with the zero-level set being $S_{out}$; so, the normal of $S_{out}$ at a surface point $p$ is the gradient of $D(S_{out})$, i.e. $\nabla D(S_{out})$, evaluated at that point. Therefore, in the first step, we initially move in the direction of the normal of $S_{out}$ at $p$. Given that $p$ is assumed to be on the true surface, $S^*$, by moving this way, we will reduce the chance of erroneously "crossing" $S^*$. After a small distance $l$, we could have continued to move along $\nabla D(S_{out})$. However, we switch to moving along $-\nabla D(S_{in})$ for the following reasons. First, if we have a group of constraint points that are close together, then their respective $bl(p)$'s built by using $\nabla D(S_{out})$ will usually meet and collapse into a single curve well before $S_{in}$ is reached. Such a merge is not desirable when the graph cut weight from a voxel $v$ in $bl(p)$ to the $Sink$ node is not set to infinity, but to some other smaller value. (This is necessary for dealing with noise and discretization ambiguities - see below). Second, there are places where the above gradient fields vanish, and we must either abandon constructing the current $bl(p)$ or need several bookkeeping steps such as making small random jumps to take care of this issue. Of the two gradient fields, $\nabla D(S_{in})$ is more homogenous and this happens less frequently to it.

This procedure constructs the set of all blocking curves **BL** through which the solution cut should not pass. This constraint might be incorporated into the graph cut algorithm by setting the weights of the edges from each voxel in **BL** to the $Sink$ node to be infinity. However, the set **P** (and hence **BL**) often contains false positives, so this strategy can lead to significant errors. Therefore, instead, for every voxel $v \in \mathbf{BL}$, we set $w(v, Sink) = 4/3\pi h^2$, where $h$ is the voxel size. This is the maximum weight for the edges between any two neighboring voxels in $V$. This uniform weight setting works well provided that the silhouette set is accurate, as shown in experiments on synthetic data in section 5.

**Incorporating silhouette uncertainties.** When dealing with real image sequences, errors in background subtraction and from the morphological operations typically employed to find silhouettes introduce artifacts ([3]). So, there is always uncertainty in silhouette extraction. We would, of course, like our silhouette to be as accurate as possible. But we still need to measure the local positional uncertainty of the silhouette and incorporate this uncertainty into the surface estimation algorithm. We extract silhouettes in the following way. First a background image, $I_{bgr}$, is subtracted from the image $I$, with $\triangle I = |I - I_{bgr}|$. Then, a small threshold $\theta_I = 2\sigma_{noise}$ is applied to $\triangle I$ to get the largest connected component $BW_{obj}$, which is assumed to contain the object's true silhouette. Next, along the boundary of $BW_{obj}$, we find the set $P_{fix}$ - a set of high confidence silhouette points - where$P_{fix} = \{p \mid \triangle I > \Theta_I\}$ and $\Theta_I$ is a large threshold. Finally, an active contour method is applied to $\triangle I$ with points in $P_{fix}$ being part of the contour and fixed. So, we first identify boundary sections with high likelihood of being on the silhouette and recover the rest of the silhouette with an active contour. The associated uncertainties for points on contours are measured with a quadratic function as described below.

The uncertainties on the location of the silhouette affect the process of finding **P**. To account for them, we need to determine the probability that a point in **P**

is really on $S^*$. It is estimated with a combination of cues from silhouette and photometric consistency, i.e.

$$\Pr(p \in \mathbf{P}) \sim \Pr(PhotoConsistency(p), a \in Sil_i) \tag{3}$$

where $p$ projects to the point $a$ on the silhouette $Sil_i$ through the camera center $C_i$. Assuming that photo consistency and silhouette uncertainty are independent, we have

$$\Pr(p \in \mathbf{P}) \sim \Pr(PhotoConsistency(p))\Pr(a \in Sil_i) \tag{4}$$

$$\sim \rho(p)\Pr(a \in Sil_i) \tag{5}$$

where, similar to [3], $\Pr(a \in Sil_i)$ is a truncated linear function of $|\triangle I(a)|^2$. (Figure 3.b illustrates uncertainty measure along the contour extracted from a difference image).

The search region, $s \subset S_{out}$, for a constraint voxel $p$ described in section 3.1 is now extended to a sub-volume around $s$ with a thickness proportionate to $\Pr(a \in Sil_i)$. Note that the extension is also outwards in addition to inwards. To determine the color consistency value for the searched points that are outside $V$ which haven't been computed so far, we dilate $V$ with a small disk (e.g. a disk of $5 \times 5$ pixels) and proceed with the $\rho$ computation described in section 2. Instead of applying uniform weight to the edges connecting voxels in $\mathbf{BL}$ to the $Sink$ node, we now weight these edges for $p \in \mathbf{P}$ and for voxels that are in the associated $bl(p)$ using $\Pr(p \in \mathbf{P})$.

## 4 A Second Phase to Handle Concavities

As discussed in section 3.1, the set of surface constraint points, $\mathbf{P}$, provides a large set of constraints on surface recovery which tend to best capture protrusive parts of the object's surface. So, the surface reconstructed by the first stage of recovery (phase I) is generally accurate over such areas. This is supported by the experiments described in section 5. On the other hand, it is well known that the silhouette does not contain information about concave regions of the surface ([3]). In addition, the graph cut algorithm, which prefers shorter cuts, will not follow a concavity well unless we "aggressively" pursue it.

We propose the procedure in figure 4 as a second phase to correct the estimation of the surface over concave regions.

We first (step 1) divide all of the voxels on the surface $S_I$ into three groups. The first group,$\mathbf{P}_{surf}$, has small $\rho$ (or high photo consistency); the second group, $\mathbf{P}_{outside}$, consists of voxels with high $\rho$ ; and the last group consists of the remaining voxels. Percentile$(S, \theta)$ returns the $\rho$ value which is the $\theta$-th percentile of the $\rho$ score for $S_I$. The parameters $\theta_1$ and $\theta_2$ determine the size of each group. In general, their appropriate values depend on the properties of the surface under consideration. Although as we observed, the final result is not very sensitive to these parameters. For the experiments in section 5, $\theta_1$ and $\theta_2$ are set to 0.7 and 0.95 respectively.

Let $S_I$ be the surface constructed by the algorithm in phase I.

*Step 1.* From $S_I$, extract two sets of points $\mathbf{P}_{surf}$ and $\mathbf{P}_{outside}$,

$$\mathbf{P}_{surf} = \{v \mid v \in S_I \text{ and } \rho(v) < \text{Percentile}(S_I, \theta_1)\} \tag{6}$$

$$\mathbf{P}_{outside} = \{v \mid v \in S_I \text{ and } \rho(v) > \text{Percentile}(S_I, \theta_2)\} \tag{7}$$

*Step 2.* Using the procedure in section 3.2 to find $\mathbf{BL}_{inside} = \cup_{v \in \mathbf{P}_{surf}} bl(v)$.
Set the weight $w(v, Sink)$ for all $v \in \mathbf{BL}_{inside}$ using the previous method.

*Step 3.* Get $\mathbf{BL}_{outside} = \cup_{v \in \mathbf{P}_{outside}} bl(v)$ with the procedure in section 3.2
For all $v \in \mathbf{BL}_{outside}$ and $v \notin \mathbf{BL}_{inside}$

$$w(v, Source) = c.\Pr(v \textit{ is outside } S^*) = c. \int_{d(v)}^{\infty} \exp(-p^2/\sigma_{surf}^2) dp \tag{8}$$

where $c$ is a normalizing constant, $d(v)$ is the distance from $v$ to $S_{out}$.
The weights for all remaining voxels are set using photo consistency scores as before.

*Step 4.* Perform the graph cut algorithm to extract the final surface, $S_{II}$.

**Fig. 4.** The steps of the second phase

Since all voxels in $\mathbf{P}_{surf}$ lie on $S_I$ and have high photo consistency (small $\rho$), we assume that they belong to or are very close to the true surface $S^*$. Therefore, in step 2, we connect them and all the voxels in their associated $\mathbf{BL}_{inside}$ to the $Sink$ node. Essentially, we treat $\mathbf{P}_{surf}$ in a similar way to the set of constraint points, $\mathbf{P}$, in phase I.

On the other hand, the voxels in $\mathbf{P}_{outside}$ have low photo consistency (high $\rho$), so in step 3 we connect them to the $Source$ node. By doing so, we effectively assume that these voxels are outside the true surface $S^*$ (and hence do not belong to the object's occupancy volume). The reasons we do this are as follows. Any such voxel is unlikely to lie on the actual surface $S^*$, so is either inside or outside of it. If such a voxel were inside the true surface $S^*$, then the surface region on $S^*$ that "covers" it would either be protrusive (case 1 - fig. 5) or concave (case 2 - fig. 5). If this region were protrusive (case 1), then it would likely have been captured by the constraint points, P, so would have been included in $S_I$ by phase I. If that region were concave (case 2), then the phase I graph cut algorithm would have included the region in $S_I$, instead of $\mathbf{P}_{outside}$, because it would have incurred a smaller cutting cost. This is because voxels that lie on that region would have low $\rho$, while the voxels in $\mathbf{P}_{outside}$ have high $\rho$ and form even more concave (or "longer") surface regions. Therefore, voxels in $\mathbf{P}_{outside}$ are assumed to be outside of $S^*$ (case 3 - fig. 5), the only remaining possibility.

Moreover, the region of $S^*$ that lies "under" $\mathbf{P}_{outside}$ is assumed to be concave. Therefore, to better recover it, we bias the solution cut inwards by treating the blocking curves $\mathbf{BL}_{outside}$ differently. Voxels on these curves are assumed to be outside $S^*$ with a probability distribution that decreases as the distance of these

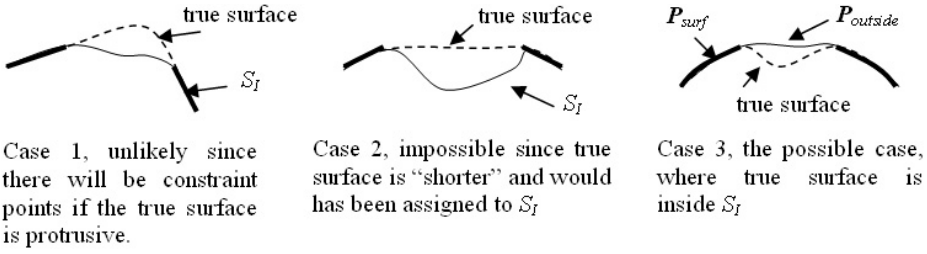


**Fig. 5.** Possible displacements of $S_I$ and $S^*$. The solid curve represents $S_I$ with bold segments for $\mathbf{P}_{surf}$ and thin segments for $\mathbf{P}_{outside}$. Of these cases, only case 3 is likely.

voxels from $S_{out}$ increases (note that we use $S_{out}$ instead of $S_I$). We model the probability of the surface location as a Gaussian distribution $N(S_m, \sigma^2_{surf})$, where $S_m$ is a "mean surface" midway between $S_{out}$ and $S_{in}$. The variance $\sigma^2_{surf}$ is set to be $(1/4d)^2$ for the experiments in section 5, where $d$ is the distance from $S_{in}$ to $S_{out}$. This leads to approximating the probability that a voxel $v$ is outside of $S^*$ with the cumulative distribution of $N(S_m, \sigma^2_{surf})$, and so the weight from voxels in $\mathbf{BL}_{outside}$ to the *Source* node is computed using (8) in step 3.

## 5 Experimental Results

We demonstrate the performance of our approach on both synthetic and real data (640 × 480 images). Volumetric discretization are 256 × 256 × 256 for all

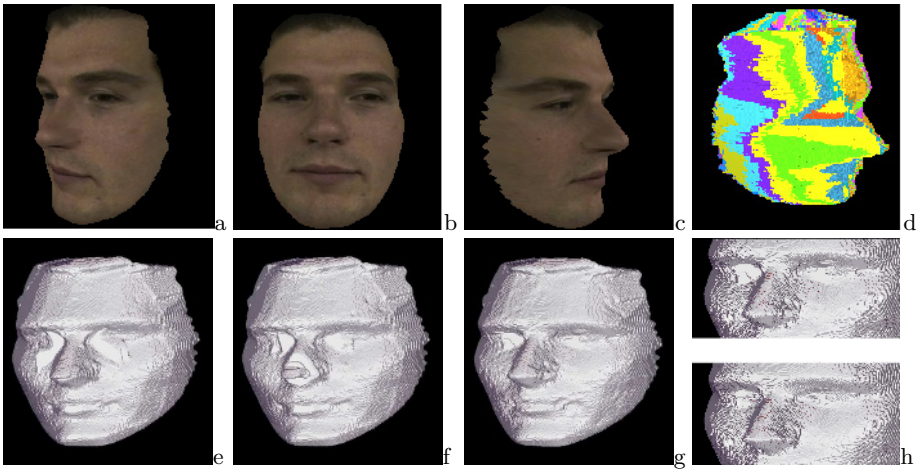


**Fig. 6.** Synthetic face reconstruction: a-c) Three of the images collected; d) visual hull $V$; e-f) using basic step, $\lambda = .3$ and .1; g) using constraint points $P$ after phase I; h) after phase II (bottom) as compared to after phase I (top)

experiments. The synthetic experiment is with a textured head (figure 6.a-c). Note that the nose is quite protrusive and the eye areas are concave. For the results in figure 6, twenty images and the associated calibration information were constructed. Figure 6.d shows the visual hull $V$ obtained from the silhouettes. Each colored patch of the surface $S_{out}$ is "carved" by some camera. Patches from any single camera may not be connected and so are rims ([9]). Moreover, if self-occlusion occurs, some patches may not contain any true surface points at all. Figure 6.e and 6.f show the result of using the basic algorithm from [8] described in section 3.1 with different ballooning factors, $\lambda$, to overcome the preference of the algorithm to shorter cuts. As can be seen, if $\lambda$ is too high (0.3), the protrusive parts (the nose) are preserved, but the concave regions (the eyes) suffer. Lowering $\lambda$ (0.1) helps to recover concave areas but at the price of losing protrusive parts. Figure 6.g shows the result of phase I when constraint points are used. Protrusive parts are well preserved now. However, the concave regions still are still not accurately recovered: the eye areas are nearly flat. Figure 6.h compares the results of phase I (the top part) and phase II (the bottom part), where the eye areas are now improved.

In the second experiment, we measure the reconstruction errors of the synthetic face when different numbers of views are used (8, 16, 32, and 64). In generating images, the viewing direction of the camera is always towards the center of the face. For every set of views, the camera is placed in positions that are arbitrary, but distributed roughly even in front of the face. For the basic algorithm, $\lambda$ is set to 0.15 to get a balance between the recovery of protrusions and concavities. Since the ground truth for the face is given as a cloud of points, $G_0$, we use the 3D distance transform to measure the recovery error $E$. Specifically, for a surface $S$, $E(S, G_0) = (D(S, G_0) + D(G_0, S))/(|S| + |G_0|)$, where $D(S, G_0)$ is the sum of distances from all points in $S$ to $G_0$. $E(S, G_0)$ is thus the average distance between points in S and $G_0$ (in voxel units). Figure 7 shows the

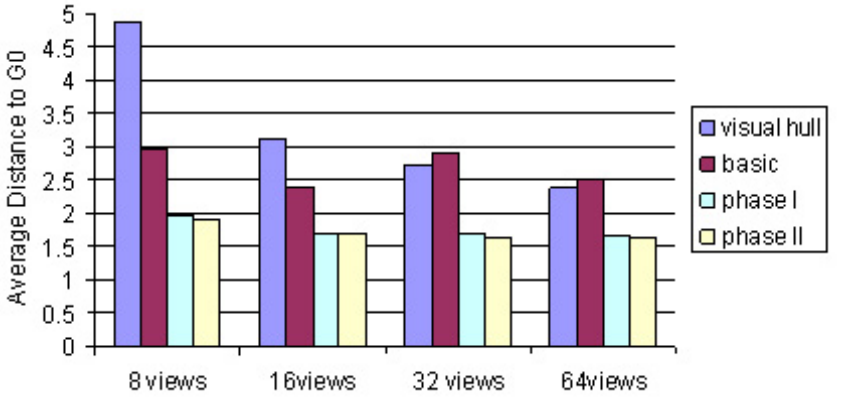


**Fig. 7.** Recovery errors for different set of views. For each group, from left to right, values are respectively for visual hull, basic algorithm and our phase I, II results. (A unit along the y-axis corresponds to the size of a voxel).

reconstruction errors. The visual hull $V$ produces quite a large error with 8 views but is noticeably better as the number of views increases. For the basic algorithm, with $\lambda = 0.15$, some protrusions are cut off. Note that since the cutting off effects can have unpredictable consequences, the reconstruction error can increase as the number of views increases (although not significantly). Adding more views in this case turns out to be "helping" the nose of the face to be more cut off. As a result, the visual hull may produce better results for larger number of views. Our methods behave consistently and produce better performance. Our result with 8 views, although with no discernible improvement for more than 16 views, is better than the visual hull with 64 views. The error of our method compared to the basic algorithm, is reduced roughly 33%. Note that in term of average error distance, phase II is not much better than phase I. This is because the focus of phase II is only on small (concave) portions left by phase I ($\theta_2 = 0.95$, section 4).

In the third experiment, 30 real images of a colored plastic squirrel were collected. We imaged the object under natural lighting conditions with a cluttered background, and moved the camera around the object. Due to self-shadowing and the arrangement of the light sources, the object is well lit on one side and poorly lit on the other side (see figures 8.a and 8.b for examples). The color information from the poorly lit side is noisy and tends to saturate to black. These 30 images are divided roughly even for both sides. The object's actual size is about $300 \times 150 \times 300$ mm$^3$ (width-length-height); this is also the size of the discretized volume used. Camera calibration was done using a publicly available tool box with the principal point's uncertainty from $1.4 - 1.7$ pixels. Silhouette extraction is performed using the method described in section 3.2. The silhouettes can be 1 to 5 pixels off from the "true" silhouettes. Figure 8.c show the visual hull constructed from them. Assuming that these silhouettes are exact leads to undesirable consequences. Figure 8.d shows the result of the basic algorithm. Even when we add the set of constraint points, our algorithm (phase I) still produces bad results: a number of incorrect bumps and dents on the surface. Figure 8.e, top row, zooms in on some of them (the image are smoothen for better visualization). Adding silhouette uncertainties (bottom row) produce much improved results. To allow for comparison with the basic algorithm, the dilated visual hull discussed at the end of section 3.2 is also used for it.

For the well lit side of the object, figure 8.f shows the result of the basic algorithm and figure 8.g shows the result of our methods (phase I). Figure 8.h compares the two results on several places: the top row is for the basic algorithm and the bottom row is for ours. The phase I and phase II give nearly the same result. In other words, phase II has little effects on this well-illuminated side.

For poorly lit side of the object, figure 8.k shows the result of the basic algorithm, figure 8.l is for phase I and figure 8.m is for phase II. Note the difference between the two legs and along the tail.

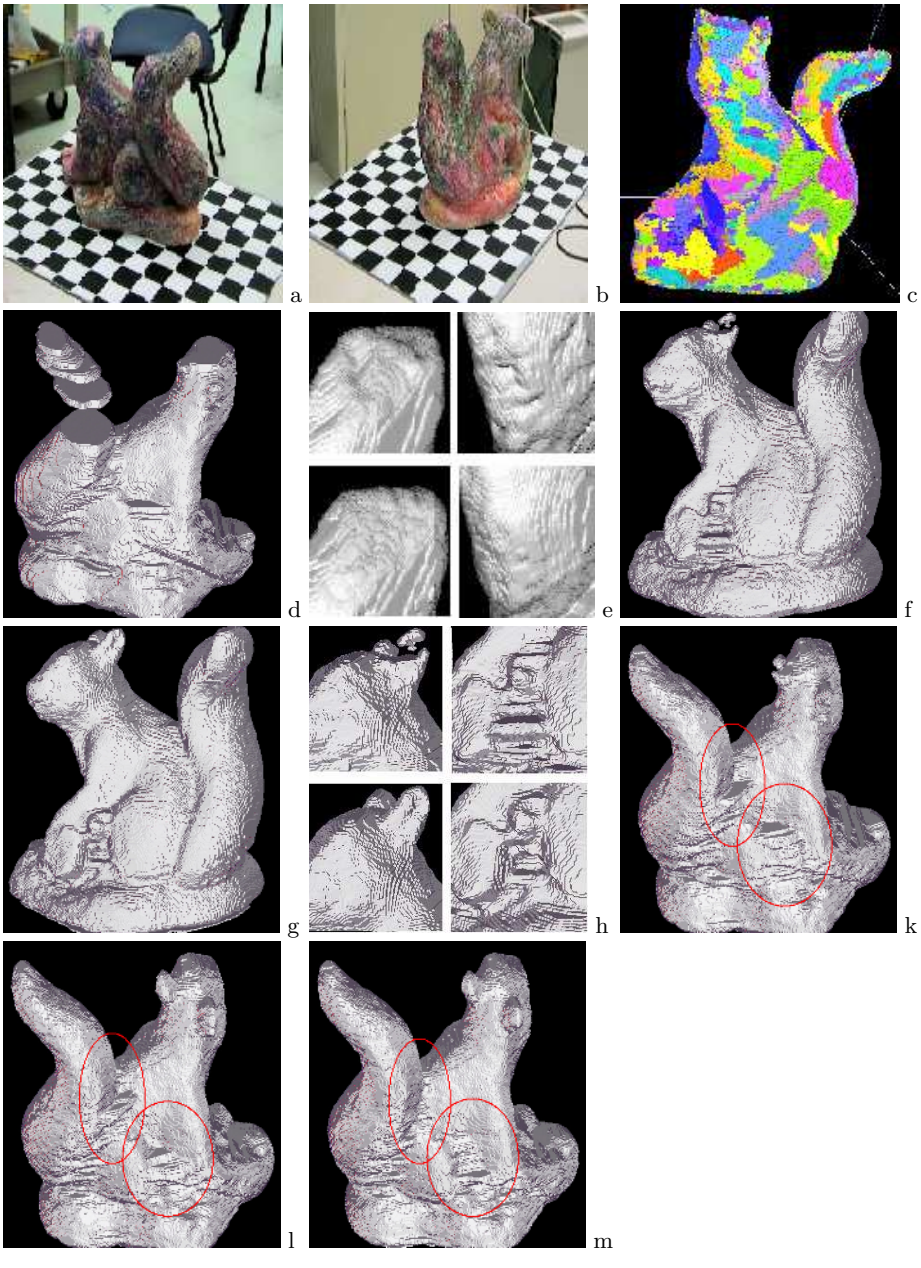


**Fig. 8.** Reconstruction of the squirrel object. a-b) two of the images collected; c) the visual hull $V$; d-e) the result of the basic algorithm and our phase I when silhouettes are assumed exact (see text). Well lit area results: f) the basic algorithm; g) our phase I algorithm; h) some detailed comparison between the basic algorithm (top row) and the final result of phase I (bottom row). Poorly lit area results: k) the basic, l) phase I and m) phase II algorithms. Note the differences inside the red circles.

## References


1. Szeliski, R.: Rapid octree construction from image sequences. CVGIP: Image Understanding **57** (1993) 23–32
2. Cheung, G.K.M., Baker, S., Kanade, T.: Visual hull alignment and refinement across time: A 3d reconstruction algorithm combining shape-from-silhouette with stereo. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR-2003). (2003) 375–382
3. Snow, D., Viola, P., Zabih, R.: Exact voxel occupancy with graph cuts. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR-2000). (2000) 345–352
4. Paris, S., Sillion, F., Long, L.: A surface reconstruction method using global graph cut optimization. In: Proc. Asian Conf. Computer Vision (ACCV-2004). (2004)
5. K. Kutulakos, K., Seitz, S.: A theory of shape by space carving. In: Proc. IEEE Int'l Conf. Computer Vision (ICCV-1999). (1999) 307–314
6. Solem, J., Kahl, F., Heyden, A.: Visibility constrained surface evolution. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR-2005). (2005) 892–900
7. Isidoro, J., Sclaroff, S.: Stochastic refinement of the visual hull to satisfy photometric and silhouette consistency constraints. In: Proc. IEEE Int'l Conf. Computer Vision (ICCV-2003). (2003) 1335–1342
8. Vogiatzis, G., Torr, P., Cippola, R.: Multi-view stereo via volumetric graph-cuts. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR-2005). (2005) 391–399
9. Sinha, S.N., Pollefeys, M.: Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In: Proc. IEEE Int'l Conf. Computer Vision (ICCV-2005). (2005) I:349–356
10. Boykov, Y., Kolmogorov, V.: Computing geodesics and minimal surfaces via graph cuts. In: Proc. IEEE Int'l Conf. Computer Vision (ICCV-2003). (2003) 26–33
11. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. and Machine Intell. **23** (2001) 1222–1239
12. Boykov, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In: Proc. IEEE Int'l Conf. Computer Vision (ICCV-2001). (2001) 105–112
13. Esteban, C.H., Schmitt, F.: Silhouette and stereo fusion for 3d object modeling. In: Proc. 4th Int'l Conf. on 3D Digital Imaging and Modeling (3DIM 2003). (2003) 46–53