A novel approach to view planning in shape acquisition from range data

Chi-Wing Fu and Florin Cutzu and Andrew Hanson
Department of Computer Science
Indiana University
Bloomington, Indiana, 47405
USA
E-mail: {cwfu,florin,hanson}@cs.indiana.edu

Version: November 14, 2002

We present an image-based approach to the determination of the next best view (NBV) in the incremental construction of object models from multiple range images. No *a priori* constraints are imposed on object shape, which is considered completely unknown. Rather than reconstruct a 3-D model like most current methods do, we use image-based representations to model the captured range information. Our method requires only conventional graphics hardware, and since it uses image-based rendering, which is independent of object complexity, it makes near-real-time NBV determination for arbitrarily complex objects possible in practice.

Key Words: Next Best View; Range Data; Laser scanner; 3-D Shape; Surface geometry; Visually-guided robot manipulation & navigation.

1. INTRODUCTION

1.1. Problem statement

A range finder or scanner is a device that measures distance from a fixed reference point (camera) to various points in a 3-D scene. The range scanner outputs an array of distances to a grid of sampled locations in the scene, or a *range image*. The reconstruction of the shape of 3-D objects from depth maps provided by laser rangefinders requires that multiple object views be scanned. It is desirable to devise scanning strategies that minimize the number of such range images, since a brute-force data-gathering strategy—uniformly sampling of the viewing sphere for example—would result in numerous, redundant measurements of the object's surface (such approaches typically use tens of scanning viewpoints, resulting in substantial overlap among the scanning views). Note that if information regarding the geometry of the object was available, the problem of view planning would be greatly simplified. However, in the most general case the geometry of the object is not known and therefore the next view to be scanned (the next best view, abbreviated NBV) must be decided solely on the basis of the range data captured at previous viewpoints.

The significance of the NBV problem lies extends beyond finding efficient ways of building models of 3-D objects to problems in robot navigation [20], object manipulation, automatic inspection [19], and computational geometry (the art gallery problem).

In this paper we describe an efficient, image-based method for planning the sequence of viewpoints—the next best view, or the NBV sequence—used by a laser scanner in acquiring the shape of a 3-D object of unknown, arbitrary shape.

1.2. Related work

The problem of view planning for 3-D object reconstruction from range data has been studied since the mid-eighties. Despite this intensive study, finding an efficient solution applicable to objects of arbitrary geometry and topology remains an open problem.

According to Tarabanis *et al* [18], a paper which surveys the closely related problem of sensor planning, one can classify existing NBV determination methods into *generate-and-test methods* [10, 12, 8] and *synthesis methods* [2, 14]. Generate-and-test methods generate candidate next views, tests them using an optimality criterion reflecting the amount of novel object information that can be gained at the novel viewpoint, and selects the best. Synthesis methods, on the other hand, use various constraints on the viewing space to limit the potential NBV locations. The complete solution space for a particular constraint is derived and combined with other solution spaces, or searched using an optimization criterion. Our method falls in the generate-and-test category.

Existing approaches to the NBV problem can also be classified based on how they represent the unexplored space. Connoly [2] (the first paper on the subject of NBV determination), and Tarbox and Gottslich [19], used octrees to represent viewed and unviewed space. Marver and Bajcsy [10] use occluding edges to represent geometrical information, focusing only the locations at the borders of the occlusions to compute the next best view. Their approach relied on triangulation based scanners and thus is unsuitable for large scenes. Whaite and Ferrie [21] assumed that the scanned object can be approximated by a collection of parts that can be modeled with superquadrics. They defined a uncertainty measure for the object model at a given moment in the scan, and the best next view was chosen to maximizes the reduction in uncertainty. Banta et al [3] used an occupancy grid to represent object geometry and an approximation to the surface curvature of the current geometry was used to choose three possible viewpoints, which are then evaluated by ray tracing the entire model. Pito [13] explicitly represented the seen-unseen (scanned-unscanned) boundary on the surface of the object and found the new viewpoints of the range camera that views as much of this boundary as possible. Reed and Allen [14] model all object occlusions and the scanned surfaces and determine occlusion-free viewpoints for portions of the occlusion boundary.

Our method belongs to the class of generate-and-test methods, and, like other methods [14, 10] is based on reasoning about object self-occlusion: we use the occlusion boundaries at the current viewpoint to determine the next viewpoint. In addition, we use image-based rendering, supported by conventional graphics hardware, to eliminate the need for surface reconstruction and thus accelerate the generate-and-test process.

1.3. Objectives

2.1 Our goal a NBV algorithm with the following properties:

1. *Generality* The algorithm should apply to objects of arbitrary topology and geometry.

2. *Efficiency* The computation of the NBV should not, on the average, take longer than the acquisition of the range data. Efficiency is the strong point of the proposed method, since the time complexity of an image-based method (as opposed to a volume-based one) is independent of object complexity; instead, it depends on the resolution of the range data and on the number of captured views.

3. *Accuracy* The sequence of views generated by the algorithm should cover the entire object surface, except for regions of a prespecified maximum size.

1.4. Paper outline

Section 2 discusses the geometrical representation of the known and unknown object information throughout the process of range data accumulation. Section 3 describes the proposed NBV algorithm. Section 4 presents results and discuses implementation issues. Finally, in Section 5, we draw conclusions and discuss future work.

2. REPRESENTING OBJECT SHAPE

Throughout the scanning process one must maintain a representation of what is known about the object shape—the scanned surface regions, and of what is still unknown—the yet-to-be-scanned regions.

2.1. Representing what is known: surfels

The determination of the NBV must be based on all available information regarding the shape of the object undergoing the scan. Thus, at first glance, it appears that to compute the NBV one must first construct a 3-D object representation (surface-based or volume-based) from the already acquired range data.

However, the range camera merely *samples* the object's surface, producing a discrete set of surface points that can easily be mis-interpreted in such a reconstruction, thus leading to an incorrect NBV.

Consider the 2D model in Figure 1. Two types of surface reconstruction errors can occur: erroneous joining of adjacent surface points (mis-connection) and erroneous separation of adjacent surface points (mis-disconnection). Mis-disconnection tends to occur for surfaces almost parallel to the viewing direction and results in spurious holes in the model surface. Mis-connection leads to erroneous occlusions – for example the central hole in Figure 1 is missed. These problems are especially severe for large and complex objects [6].

Furthermore, reconstruction is usually computationally expensive, slowing down NBV determination beyond the time needed for acquisition of range data, thus contradicting the second of our stated objectives (Section).

For these reasons we separate the reconstruction step from the NBV determination process. Instead of *interpreting* the (known) sampled range data in terms of a 3-D structure, we directly *represent* them by using **surfels** (surface elements) [11]. Each surfel is associated with a sampled surface point. Our method does not require the recovery of surfel normals. Surfel density depends on the resolution of the scan. Figure 2 zooms into surfels of various objects.

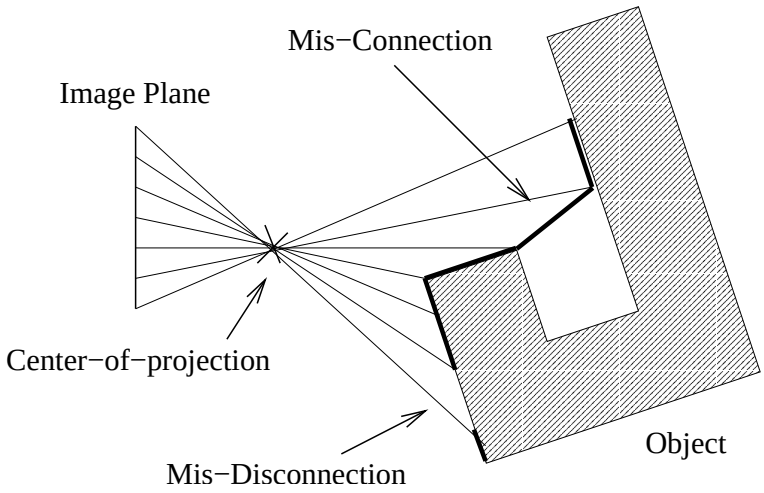
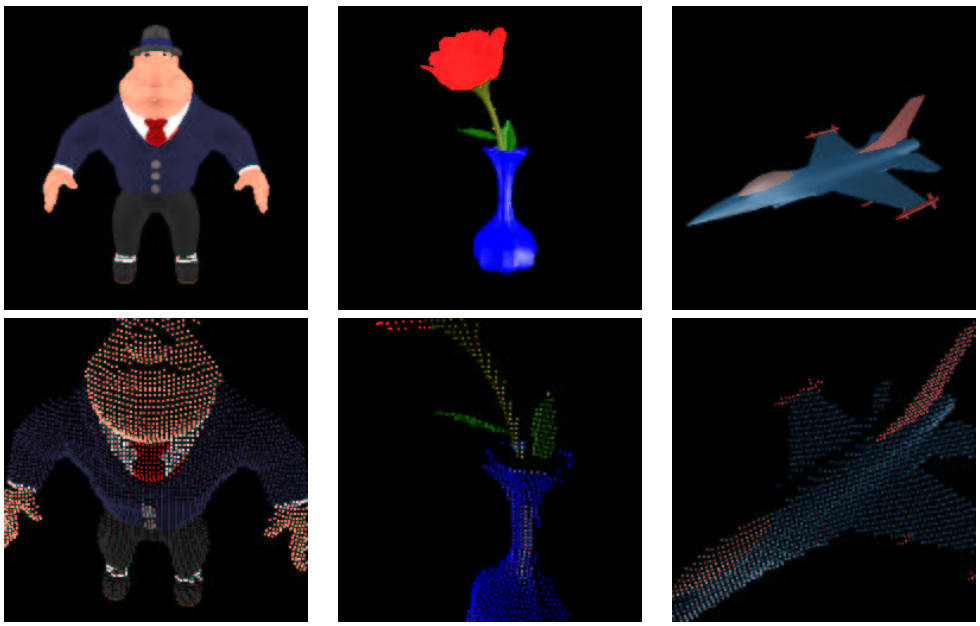


FIG. 1 Incorrect reconstruction: Erroneous joining of adjacent surfels (mis-connections) and erroneous separation of adjacent surfels (mis-disconnection). The thick line represents the surface reconstructed from the sample points.



(a) human model (b) rose model (c) F-16 model

FIG. 2 Zooming into the surfels on various objects.

2.2. Representing what is unknown: visual hulls

Our method employs image-based representations for both the acquired shape information and for the yet-unknown object regions, at any moment during the object scan. To represent the unknown object regions, we use the method of *image-based visual hulls* [9, 5].

As illustrated in Figure 3, the visual hull is a pyramid-shaped volume extending to infinity from the back side of the sampled surface, away from the center of projection.

Given two different views of the object, any point invisible from both views is located within the intersection of the two corresponding visual hulls. Visual hulls are usually computed using Constructive Solid Geometry (CSG) intersection techniques. However, these methods require additional reconstructions for volumizing the sampled data. A more efficient, ray-casting-based method for computing visual hull intersection is given in [9].

Thus, instead of using volume-based representations, we represent visual hulls in image-space. Following [7], we connect neighboring surfels into a rectangular mesh. In addition, for surfels on the object silhouette, we extend the mesh to infinity, away from the center of projection, along the line of sight. Note that we are not reconstructing the object surface, but representing the *unknown* object information only. Surfel mis-connection is not an issue, as it does not affect the representation of the unknown information. Note that no reconstruction step is needed in building the mesh, which is a direct representation of the sampled range data. Since the mesh is image-based and has the same resolution as the range data, it can be efficiently computed and rendered [7]. A typical view of the "Porsche" model (depth image resolution $= 64 \times 64$) is shown in Figure 4(a) with corresponding surfels and visual hull (from above) in Figure 4(b) and (c) respectively.

2.2.1. Computing visual hull intersection using the graphics hardware

Next, we need to *intersect the visual hulls* corresponding to different object views. Rather than using ray casting or volume-based methods, we compute the intersection of visual hulls by rendering the image-based visual hull representation using the depth buffer. Similar techniques can be found in [15, 22, 16].

Our algorithm is hardware-based provided hardware Z-buffering is available. At a potential viewing position each of the previously determined visual hulls are rendered one by one into the depth buffer, the final depth buffer representing the intersection of all visual hulls is the per-pixel maximum of depth values across all individual depth buffers. Figure 5 illustrates this rule. Figures 5(a) and (b) show depth buffers corresponding to the visual hulls of views 1 and 2 with thick lines. If we take the maximum of these two depth buffers, we can obtain a valid depth buffer (Figure 5(c)) for the visual hull intersection at the potential view. The hardware-based algorithm listed in Figure 6 is based on this rule and requires only general graphics hardware. The OpenGL function *glDepthFunc* is used to compute the maximum of all depth buffers.

glDepthFunc compares the input pixel depth value with the pixel depth value present in the depth buffer. For example, in case $GL\_GREATER$ is used, if the input pixel depth value is greater than the pixel depth value present in the depth buffer, the input pixel can be written to the frame buffer; otherwise the input pixel is ignored. Thus, when the temporary buffer back is written to the depth buffer

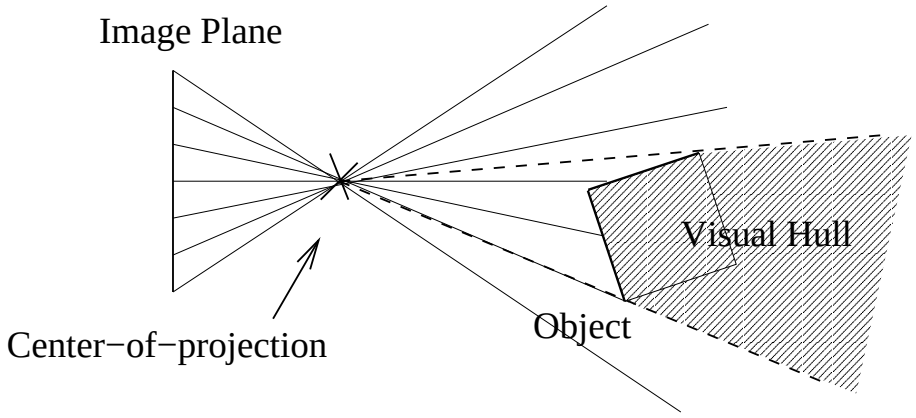
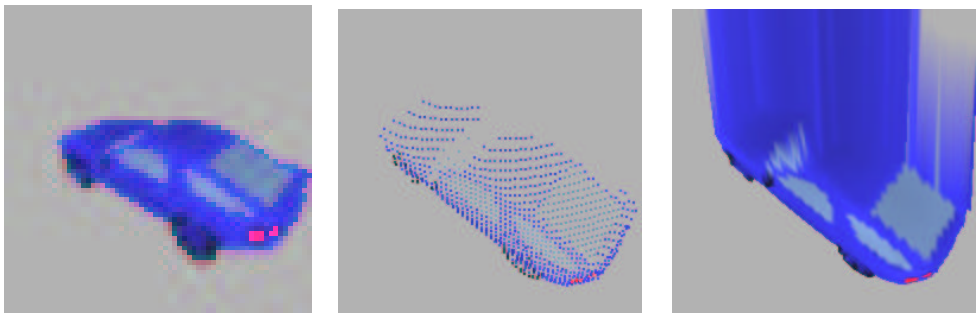


FIG. 3 Visual hull: representing the unknown object region.

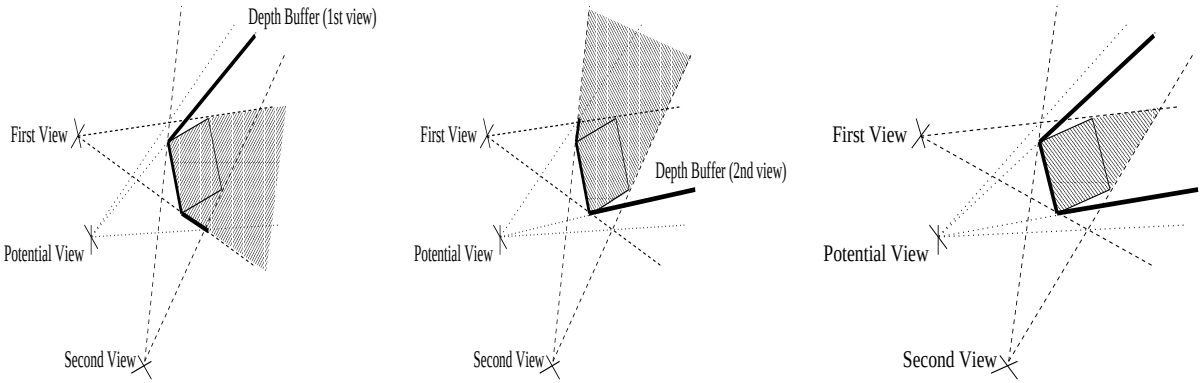


(a) Captured view (64×64 surfels)

(b) Corresponding surfels seen from above

(c) Corresponding visual hull seen from above

FIG. 4 Surfels (b) and visual hull (c) captured at the object view (a).



(a) Depth Buffer at 1st View

(b) Depth Buffer at 2nd View

(c) Max(Depth Buffers)

FIG. 5 Intersection of 2D Visual Hulls: maximum of two depth buffers. The thick lines show the depth buffer at a potential view.

using $GL\_GREATER$, the hardware can compute the maximum of the two depth buffers.

3. FINDING THE BEST NEXT VIEW

3.1. Definitions

The **viewing space**, V, is the set of all possible poses $\boldsymbol{v}$ of the range camera from which the object can be scanned. In practice, V is determined by the degrees of freedom of the scanning mechanism – robot arm holding the camera, rotational table on which the object is placed, path of the camera-bearing vehicle, etc.

The **goodness** of a view $\boldsymbol{v}$, $g(\boldsymbol{v}, V_0)$, $g \in [0,1]$, depends on the set of already captured views $V_0 \subseteq V$, and measures how much novel (not previously sampled) object surface area can possibly be revealed at a potential (i.e. not-yet-scanned) view $\boldsymbol{v}$. The amount of object information gained at a viewpoint depends on the number of novel surfels revealed by scanning at that viewpoint. To normalize $g(\boldsymbol{v}, V_0) \in [0,1]$, we divide by the total number of pixels in the image:

$$g(\boldsymbol{v}, V_0) = \frac{\text{amount of new information captured at } \boldsymbol{v}}{\text{window size}} \tag{3.1}$$

Note that if $\boldsymbol{v} \in V_0, g(\boldsymbol{v}, V_0) = 0$ as there is no new information to be gathered at any view in V_0.

The (local) **next best view**, $\boldsymbol{v}_{best}$, maximizes the goodness-of-view in a certain neighborhood $\mathcal{N}(V_0)$ of V_0:

$$\boldsymbol{v}_{best} = \underset{\boldsymbol{v} \in \mathcal{N}(V_0)}{\text{Arg Max}}\, g(\boldsymbol{v}, V_0) \tag{3.2}$$

where the neighborhood $\mathcal{N}(V_0)$ is defined as the set of views $v' \notin V_0$ that share object silhouette surfels with views in V_0. In other words, in the context of this paper the NBV is not the global maximizer of g, but the "local" maximizer of g among the views neighboring the views that have already been captured. This definition implies that, for example, after the very first view of an object is scanned, the NBV is not the diametrically opposite view (a choice that would globally maximize g), but rather a view neighboring the first view such that some silhouette surfels would be shared between the two views.

The rationale for this non-greedy strategy is that a greedy maximization of g would eventually result in substantial overlap (re-exploration of known surfels) among views taken at later stages of the scan.

3.2. Goodness of a potential view

To efficiently determine the NBV, one must be able to *predict* the goodness of a view (a *potential* view) – that is, to estimate $g(\boldsymbol{v})$ without actually scanning $\boldsymbol{v}$. Prediction schemes based on gradual object reconstruction and geometrical calculations are prohibitively expensive computationally. We circumvent such complications by treating the goodness of a potential view, $g(\boldsymbol{v}, V_0)$, as a *rendering function*, implicitly computed by the graphics hardware when rendering V_0 at $\boldsymbol{v}$. To this end, the two image-based representations presented in Sections 2.1 and 2.2 were used.

Figure 7 illustrates the algorithm used to estimate $g(\boldsymbol{v}, V_0)$ by rendering in image space. First, the procedure `BUILD_VISUAL_HULLS` is called to set up the depth buffer corresponding to the intersection of the visual hulls at the potential view.

```
PROCEDURE BUILD_VISUAL_HULLS
BEGIN
  RENDER the visual hull at the first view into the depth buffer
  WHILE more views
     SAVE the depth buffer content to a temporary buffer
     CLEAR the depth buffer content
     RENDER the visual hull at the next view into the depth buffer
     CALL glDepthFunc(GL_GREATER)
        COPY temporary buffer to depth buffer
     CALL glDepthFunc(GL_LESS)
  END
END
```

FIG. 6 Algorithm for visual hull intersection determination.

```
PROCEDURE GOODNESS_POTENTIAL_VIEW
BEGIN
   // Build the depth buffer
   BUILD_VISUAL_HULLS

   // Paint the intersection of visual hulls in blue
   CALL glDepthFunc(GL_GREATER)
   DRAW a blue plane behind the object
   CALL glDepthFunc(GL_LESS)

   //Overlay red surfels on intersection of visual hulls (blue)
   DRAW surfels in red with size SpraySize

   // Determine goodness of view
   goodness = number of blue pixels / total number of pixels
   return goodness
END
```

FIG. 7 Algorithm: Computing $g(\boldsymbol{v}, V_0)$.

Then, to mark the resulting intersection in the frame buffer, a blue plane is drawn behind the object with `glDepthFunc(GL_GREATER)`; given that `GL_GREATER` is used and that the plane is drawn behind the object, only the pixels corresponding to the intersection of visual hulls are painted blue. Finally, a standard warping technique [7] is used to overlay red surfels on the (blue) intersection of visual hulls to indicate the captured object information. Thus, the remaining blue pixels of the intersection of visual hulls mark the unknown object regions that are available for scanning at the potential view. The goodness of a potential view is the amount of new surface information that can possibly be gained by scanning at that view, and is given by the number of remaining blue pixels (normalized by the total number of pixels).

Figure 8 shows typical red and blue images rendered by this algorithm; the corresponding captured views (V_0) can be found in Figure 2.

3.3. Determining the initial NBV

It can be assumed that the first scanned view (object orientation) is chosen randomly. One the first view is scanned, the first NBV must be determined. Its computation involves the following steps:

1. Finding depth edges and their weights The known (visible) and the unknown regions of the object's surface at a given viewpoint are separated by depth discontinuities (*depth edges* or occlusion boundaries). Intuitively, one needs to rotate the object around a depth edge to reveal what the edge occludes. Therefore, to recover the occluded regions, we first identify depth edges by applying standard edge detection techniques to the captured range data. Figure 9 shows the range image and its corresponding depth edges.

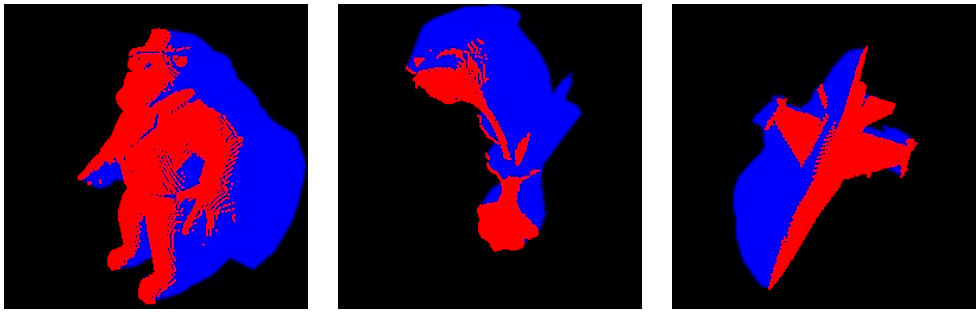
Next, depth edges are assigned weights, which are later used in a voting procedure for determining the axis of rotation to the NBV. The *weight* w of a depth edge is a measure of the unknown surface area the depth edge potentially delimits. Basically, the larger the depth "drop" across a depth edge is, the larger the area of the occluded surface region will be. The depth drop for a depth edge on the silhouette of the object (external boundary) is set to an arbitrary maximum value. The desired weight was defined as the z-differential δz across a depth edge normalized by the range $(z_{max} - z_{min})$ of all depth measurements: $w = \delta z/(z_{max} - z_{min})$.

2. Determining the axis of rotation to the NBV A depth edge connects two surfels at positions $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ on the visual rim [4] of the object. The visual rim is the set of object points on the object surface where the line of projection from the viewpoint (or camera) is tangential to the object surface; the depth drop is large along the line of sight in a direction perpendicular to the rim. We define the associated elementary rotation axis of the depth edge to be $\hat{r}$:

$$\hat{r} = (\boldsymbol{p}_1 - \boldsymbol{p}_2)/\|\boldsymbol{p}_1 - \boldsymbol{p}_2\| \tag{3.3}$$

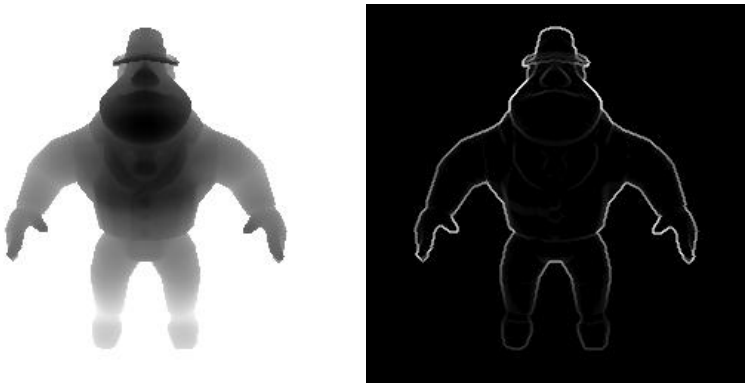
such that the depth to the right of the edge (which is directed along $\hat{r}$) is larger than the depth to the left. Thus, by rotating the object counterclockwise around $\hat{r}$, one can reveal the occluded region delimited by the edge.

Given the depth edge set, we determine the corresponding rotation axis to the NBV using the a voting scheme in which the edge weights are used. In the algorithm



(a) human model (b) rose model (c) F-16 model

FIG. 8 Red and blue-rendered images at potential views. Blue pixels indicate the novel information that can be gained at the potential views. The parts of the object surface that have already been scanned are rendered in red.



(a) depth image (b) depth edges

FIG. 9 Depth edges. The brighter the depth edge the larger its weight.

described in Figure 10, $\hat{r}_i$ is the rotation axis associated with the ith depth edge and E is the edge set.

```
# vote[i] expresses the importance of the i-th depth edge
For each depth edge, i
  vote[i] = 0
End
For each pair of depth edges, i and j, i ≠ j
  If angle(r_i,r_j) < (field of view)/2
    vote[i] = vote[i] + w_i + w_j;
    vote[j] = vote[j] + w_i + w_j;
  End
End
```

FIG. 10 Voting for the axis of rotation to the NBV

The rotation axis receiving the maximum number of votes, $\hat{r}_{max}$, is the axis of the rotation towards the NBV. Note that since this is a voting approach, we are not averaging candidate rotation axes, but finding an appropriate representative. Thus, a rotation around $\hat{r}_{max}$ will always reveal an occluded object surface. We note that the voting procedure can be accelerated by applying the Hough transform to the set of possible rotation axes.

3. Determining the angle of the rotation to the NBV After $\hat{r}_{max}$ is determined, a gradient descent search is initiated by gradually rotating the object around $\hat{r}_{max}$ to determine the optimal rotation angle and hence the NBV. As explained in Section 3.1, during gradient descent we maximize the goodness value g subject to the constraint that the NBV shares the silhouette surfels of the winning edge with the current view.

3.4. The NBV sequence

1. Sampling density and surfel registration After a new view is captured, it is necessary to determine whether any of its surfels has been previously scanned (is redundant). The quantity *SpraySize* (see Figure 7) plays an important role in this determination. As illustrated in Figure 11, each new surfel, s_{new}, captured at the novel view, $\boldsymbol{v}_{new}$, is first re-projected onto the image plane of each previously captured view by using the view interpolation technique described in [1]. Let $\boldsymbol{v}_0$ be a captured view, $\boldsymbol{v}_0 \in V_0$. The set S of surfels neighboring the surfel s_{new} re-projected onto $\boldsymbol{v}_0$ is then determined. To check whether s_{new} is redundant with respect to S, for each non-redundant surfel $s' \in S$ we calculate the maximum visual angle, α_{max}, between s' and s_{new}:

$$\alpha_{max} = \max(\angle(s_{new}, s') \text{ at } \boldsymbol{v}),\ \forall \boldsymbol{v} \in \text{viewing space } V. \tag{3.4}$$

Furthermore, note that the visual angle, α_{surfel}, of a surfel is dependent on the characteristics of the camera:

$$\alpha_{surfel} \approx \frac{\text{field of view of } (\boldsymbol{v})}{\max(w, h)} \tag{3.5}$$

where $w \times h$ is the resolution of the range camera.

Therefore, to check whether s_{new} is redundant, we compare α_{max} and α_{surfel} together with $SpraySize$:

$$s_{new} = \begin{cases} \text{redundant} & \text{if } \alpha_{max} \leq \alpha_{surfel} \times SpraySize \\ \text{non-redundant} & \text{otherwise.} \end{cases} \tag{3.6}$$

If the maximum visual angle between s_{new} and s' is smaller than a surfel's visual angle (subject to $SpraySize$ adjustment), the separation between s_{new} and s' is deemed too small to be perceptible and s_{new} is considered redundant. Thus, surfel size ($SpraySize$) can be used to adjust surfel density.

The final output of the NBV algorithm is a set of non-redundant surfels sampling the object surface. Then, point-based surface reconstruction techniques such as [17] can be applied for reconstructing the object.

2. From Depth Edges to Information Edges Depth edges delimit non-occluded from occluded surface regions at a view. However, it is not necessarily the case that all occluded surface regions at a view are unknown–some of these occluded surface region may have previously been scanned by the laser beam.

Thus, some depth edges at a new view v_{new} may delimit surface regions that have already been scanned at some previously captured views (the views in the set V_0).

On the other hand, some of the depth edges (surfel pairs) that were identified at previously captured views $v \in V_0$, are "resolved" at v_{new} when the object surface on both of their sides becomes visible. Thus, after capturing v_{new}, we need to determine if a depth edge at v_{new} or a surfel pair that had formed an edge at the previous views V_0 is indeed separating known and unknown surface regions.

This determination can be efficiently carried out using a method similar to the redundancy check (the reprojection method) for surfels. Given a depth edge at v_{new} (or a depth edge observed at a view in the captured views set V_0), we can reproject it to any of the scanned views (or to v_{new}). If it falls onto some non-edge surfels of a scanned view (or of v_{new}) within the $SpraySize$ limit, this depth edge is on a known surface region. Thus, it is not truly delimiting known and unknown surface regions. After this elimination, we can determine a subset of depth edges, that we term *information edges*, such that the set of all information edges represents the boundary of the set of surfels, that is, the red-blue boundary. The surfels constituting the information edges are flanked by other surfels on one side of the information edge only; on the other side there are no surfels, only the blue-painted visual hull.

Formally, we define E_k to be the set of depth edges at v_k and $\hat{E}_k \subseteq E_k$ to be the set of information edges corresponding to view v_k. I_k is the set of all information edges after the capture of k views, $v_0, \ldots, v_k$.

$\hat{I}_{k+1} \subseteq I_k$ is the subset of I_k consisting of the information edges that remain after eliminating those resolved at the next view v_{k+1}. Therefore, $I_{k+1} = \hat{I}_{k+1} \cup \hat{E}_{k+1}$.

Figure 12 illustrates this scheme. At the first view, v_0, we have $E_0 = \hat{E}_0 = I_0$ as there is no previous view. The NBV voting algorithm in Figure 10 is applied to $\hat{E}_0$ to generate the first best next view, v_1. The depth edge set of view v_k is E_k. Not all these depth edges are also information edges, and therefore, we apply the reprojection method to find $\hat{E}_k$ by checking E_k against previously scanned views.

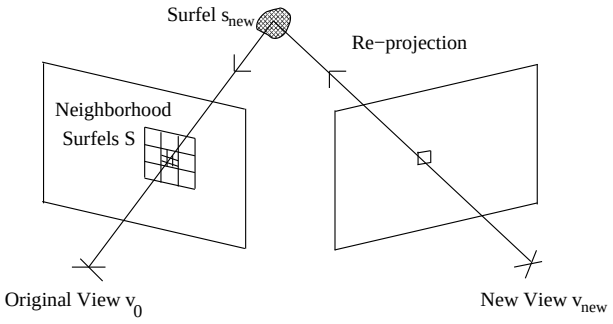


FIG. 11 Surfel registration (redundancy check) by reprojection.

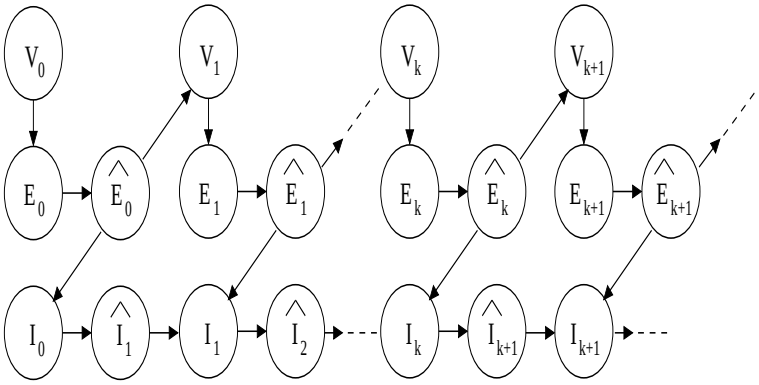


FIG. 12 The NBV sequence.

Similarly, we check I_k against v_{k+1} to find $\hat{I}_{k+1}$. $\hat{E}_k \cup \hat{I}_{k+1}$ is the set of information edges at this stage, after $k+1$ views.

We then use $\hat{E}_k$ to generate v_{k+1} based on the NBV voting algorithm. If $\hat{E}_k$ is empty, we backtrack and use $\hat{E}_{k-1} \cap I_k$ to generate v_{k+1}, etc. The algorithm stops when I_{k+1} is empty or when $I_k = I_{k+1}$.

4. IMPLEMENTATION AND RESULTS

4.1. Implementation details

We simulated object scanning experiments in **OpenGL**, using several 3-D models (see Figure 2) of different complexity and the depth buffer as the source of range data. In a real experiment, to reduce measurement noise, a range data cleanup step [17] should follow the data capture step in practice.

Our code was implemented in **OpenGL** and made use of the depth buffer support (e.g. `glDepthFunc`) for the visual hull intersection algorithm. The object of interest was enclosed in a unit sphere (object space); the camera was constrained to a spherical surface of radius 2 unit (viewing space) centered on the object center. The camera axis was always oriented towards the object center. To speed up data processing, run-length encoding was employed to represent scanlines of the captured range data.

In addition, to simulate real scanned objects, which contain noise, we added Gaussian noise to the sampled depth values. Figure 13 illustrates a noise-corrupted Porsche model.

Run length encoding along scanlines Roughly speaking, about 1/2 to 1/3 of the range/RGB-image data belong to the background. Storing the whole data set wastes memory and can slow down rendering. Instead, we apply run length encoding to store only the RGB/depth data of the object along each scanline. In practice, we can identify background range data by examining the RGB image using the blue screen technique. We assume the RGB and range data are rectified at the same view.

4.2. Results

The proposed NBV algorithm is image-based and the time complexity of the rendering function $g(\boldsymbol{v}, V_0)$ at the core of the algorithm depends on the resolution of the range image data (number of pixels), $n \times n$, and the number of captured views, $k = |V_0|$. The time complexity of $g(\boldsymbol{v}, V_0)$ is $O(kn^2)$.

On the contrary, for a volume-based method, the time complexity of $g(\boldsymbol{v}, V_0)$ is independent of $|V_0|$. However, given the same spatial resolution n along all the three dimensions of the object space, the time complexity of $g(\boldsymbol{v}, V_0)$ will be dominated by the spatial resolution of the data, $n \times n \times n$. Thus, in this case, $g(\boldsymbol{v}, V_0) \in O(n^3)$ (note that $k \ll n$).

We tested objects of different complexity: the *rose*, *F-16*, *human*, *Porsche*, and *flowers* models (in order of increasing shape complexity). Table 1 summarizes the $g(\boldsymbol{v}, V_0)$ computation time on a Pentium *IV* PC with a GeForce 3 (64MB RAM) video card and a 1400Hz CPU. The resolution of the range data was 256×256. The first column lists the names of the data sets and the timing results are shown in the next three columns. Rendering time increased quasi-linearly with the number

of captured views ($|V_0|$). Time variation among datasets indicates that rendering $g(\boldsymbol{v}, V_0)$ also depends on the number of captured surfels. Table 2 shows the NBV

Data set	$\|V_0\| = 1$	$\|V_0\| = 2$	$\|V_0\| = 3$
rose	3.354	6.037	8.374
F-16	3.174	5.620	9.124
human	4.073	7.283	10.140
Porsche	4.044	7.348	10.178
flowers	3.719	6.731	9.366

TABLE 1
Timing Result : computing $g(\boldsymbol{v}, V_0)$ (10^{-2}seconds).

computation time for the gradient descent approach. $|V_0| = k$ means that view $(k+1)$ is determined based on the previous k views. Since $g(\boldsymbol{v}, V_0)$ grows linearly with $|V_0|$, the time necessary for finding the NBV also grows with $|V_0|$. However, the time necessary for finding the NBV depends not only on $g(\boldsymbol{v}, V_0)$, but also on the geometry of the object. In any case, the NBV was always found in ~ 10 seconds, timing which roughly meets the efficiency requirement stated in the Introduction. The first four rows in Figure 14 show successive NBVs corresponding to these objects. The last two rows reveal sampled surfels from the four views. Sampled surfels in the last two rows are painted in red, green, blue, and yellow denoting the first, second, third, and the fourth views respectively.

Data set	$\|V_0\| = 1$	$\|V_0\| = 2$	$\|V_0\| = 3$
rose	0.720	5.122	10.01
F-16	0.901	4.037	15.34
human	0.708	10.78	13.57
Porsche	1.383	5.354	8.765
flowers	0.763	6.386	5.068

TABLE 2
Timing Result of finding NBVs (seconds).

Table 3 shows the number of valid and redundant surfels during capture. Obviously, there are no redundant surfels in the first view and the number of redundant surfels increases in successive views.

5. CONCLUSIONS

In summary, we have presented an image-based approach to the NBV determination problem. One major advantage of our method is that it does not require the reconstruction of the object. Moreover, the use of surfels automatically adjusts the NBV computation to the sampling density of the surface scan. The output of our algorithm is a dense set of surfels defining the scanned surface, making reconstruction trivial. By taking advantage of conventional graphics hardware, we can efficiently compute intersections of visual hulls and "render" the goodness of a potential view. Since our method is image-based, computing the goodness of a potential view depends on image data resolution n^2 and $|V_0|$ only. As a result, the

time complexity of the visibility computation is $O(|V_0|n^2)$, rather than the $O(n^3)$ complexity of volume-based methods.

In future work we will explore the inclusion of an "accuracy measure" (the third requirement for a NBV algorithm) in the NBV determination process. The accuracy measure will quantify the number of missing surfels as a function of viewing space location, thus measuring the quality of the NBV exploration. It is also desirable to generalize the viewing space beyond the viewing sphere to arbitrary surfaces and to include an automatic camera-object-collision detection scheme during NBV determination.

REFERENCES

[1] S. Eric Chen and L. Williams. View interpolation for image synthesis. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 279–288, August 1993.

[2] C. I. Connolly. The determination of next best views. In *IEEE International Conference on Robotics and Automation 1985*, pages 432–435, 1985.

[3] Banta J.E., Y. Zhien, X.Z. Wang, G. Zhang, M.T. Smith, and M. Abidi. A "best-next-view" algorithm for three-dimensional scene reconstruction using range images. In *Proc. SPIE*, volume 2588, pages 418–429, October 1995.

[4] J. J. Koenderink. *Solid Shape*. MIT Press, Cambridge, MA, 1990.

[5] A. Laurentini. The visual hull concept for silhouettebased image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, February 1994.

[6] 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. In *Proceedings of SIGGRAPH 2000*, pages 131–144, 2000.

[7] W. R. Mark, L. McMillan, and G. Bishop. Post-rendering 3d warping. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pages 7–16, April 1997.

[8] N. A. Massios and R. B. Fisher. A best next view selection algorithm incorporating a quality criterion. In *Proc. British Machine Vision Conference, BMVC97*, pages 780–789, September 1998.

[9] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan. Image-based visual hulls. In *SIGGRAPH 2000 Conference Proceedings*, Annual Conference Series, pages 369–374. ACM SIGGRAPH, Addison Wesley, July 2000.

[10] Maver and Bajcsy. Occlusions as a guide for planning the next view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):417–433, 1993.

[11] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *SIGGRAPH 2000 Conference Proceedings*, pages 335–342. ACM SIGGRAPH, July 2000.

[12] R. Pito. A sensor based solution to the next best view. In *International Conference on Pattern Recognition*, pages 941–945, 1996.

[13] R. Pito. A solution to the next best view problem for automated surface acquisition. *PAMI*, 21(10):1016–1030, October 1999.

[14] M.K. Reed and P.K. Allen. Constraint-based sensor planning for scene modeling. *PAMI*, 22(12):1460–1467, December 2000.

[15] J. Rossignac and A. Requicha. Depth-buffering display techniques for constructive solid geometry. *IEEE Computer Graphics and Applications*, 6(9):29–39, September 1986.

[16] N. Stewart, G. Leach, and S. John. An improved z-buffer csg rendering algorithm. In *Eurographics/Siggraph Workshop on Graphics Hardware*, pages 25–30, 1998.

[17] C.-K. Tang and G. Medioni. Integrated surface, curve and junction inference from sparse 3-d data sets. In *Proceedings 6th International Conference on Computer Vision (ICCV'98)*, pages 818–824, 1998.

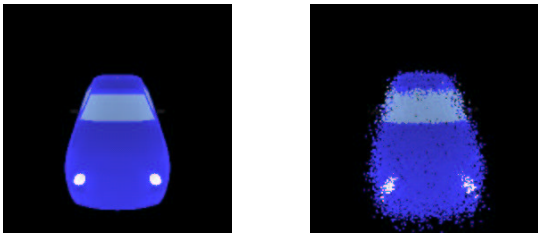
[18] K. Tarabanis, P. K. Allen, and R. Y. Tsai. A survey of sensor planning in computer vision. *IEEE Transactions on Robotics and Automation*, 11(1):86–104, February 1995.

[19] G.H. Tarbox and S.N. Gottschlich. Planning for complete sensor coverage in inspection. *Computer Vision and Image Understanding*, 61(1):84–111, January 1995.

[20] Whaite and Ferrie. Autonomous exploration: Driven by uncertainty. In *International Conference on Computer Vision and Pattern Recognition*, pages 339–346, 1994.

[21] P. Whaite and F.P. Ferrie. Uncertain views. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3–9, 1992.

[22] T. F. Wiegand. Interactive rendering of csg models. *Computer Graphics Forum*, 15(4):249–261, October 1996. Eurographics'98.



(a) Noise-free surfels (b) Noise-corrupted surfels

FIG. 13 Adding Gaussian noise to the sampled depth values.

Data set	1st view		2nd view		3rd view	
	valid surfel	redundant surfel	valid surfel	redundant surfel	valid surfel	redundant surfel
rose	6814	0	8111	304	4349	2804
F-16	5772	0	5681	1821	10937	3273
human	11847	0	11873	38	14737	2669
Porsche	11499	0	11562	27	17581	1821
flowers	7447	0	9433	426	4218	2304

TABLE 3
Number of Surfels (256×256).

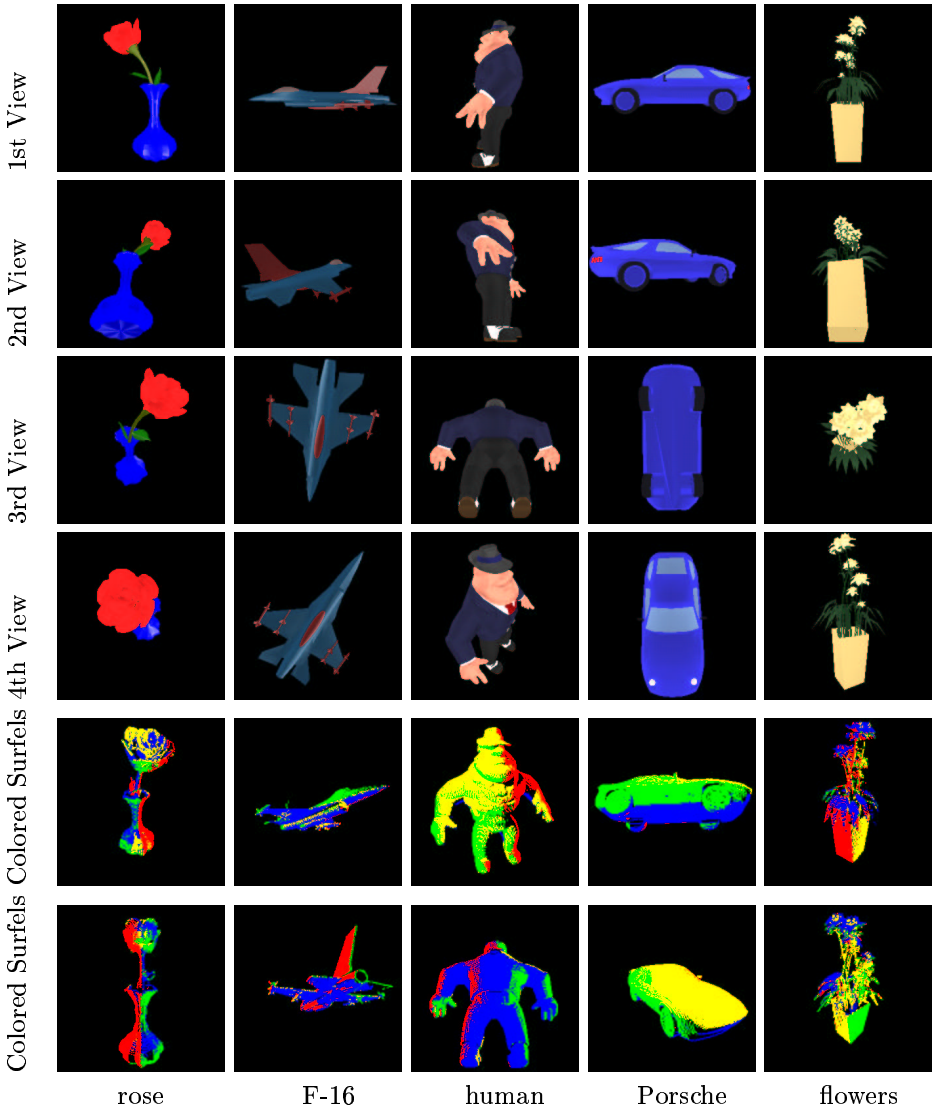


FIG. 14 The first four rows show successive NBVs of different objects and the last two rows reveal the sampled surfels at these NBV views with different colors. Surfels are painted in red, green, blue, and yellow for the first, second, third, and fourth views respectively.