Free-form Acquisition of Shape and Appearance

Pieter Peers
Vincent Masselus
Philip Dutré

Report CW 403, February 2005

Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Free-form Acquisition of Shape and Appearance

Pieter Peers
Vincent Masselus
Philip Dutré

Report CW 403, February 2005

Department of Computer Science, K.U.Leuven

Abstract

This report focuses on the acquisition of shape and appearance of real objects under static illumination without the need for an elaborate calibration process or an expensive setup. Emphasis is placed on ease of use and obtaining visually pleasing results. Using recent advances in matting techniques, an opacity hull of the object is computed from an un-calibrated sequence of photographs.

User interaction is limited to drawing trimaps for a small subset of the recorded photographs. These trimaps indicate which parts of the photographs represent the object, background or unknown. From these trimaps the mattes of the photographs can be extracted, and subsequently a coarse intermediate opacity hull can be computed. The opacity hull is iteratively refined by computing a trimap, and thus a matte, for the remaining photographs using the already computed mattes and the coarse intermediate opacity hull.

We illustrate our method on a wide range of objects, ranging from ordinary objects and human heads to large outdoor statues.

Keywords : Shape Acquisition, Appearance Acquisition, Opacity Hull.

1 Introduction

The acquisition of the shape and appearance of real objects has been widely researched in recent years. Most techniques require an elaborate calibration process or an expensive setup. In this report we start from the concept of capturing the shape and appearance of a real object by free-form camera movement. Additionally we would like to impose minimal restrictions on the kind of objects that can be captured, but still be able to re-render them from arbitrary new viewpoints. Emphasis is placed on ease of use, minimal calibration during acquisition and obtaining visually pleasing results.

During the acquisition phase an object is photographed from arbitrary positions. For this set of photographs, the intrinsic and extrinsic camera parameters are estimated. Each of these recorded images with their respective camera parameters, help to define the shape and appearance of the object.

For a selected subset of the recorded photographs, trimaps are manually drawn. A trimap defines which parts of the photograph represents the object, the background or a weighting of both (denoted as 'unknown'). From these trimaps and the respective photographs, an α-matte can be computed [RT00, CCSS01, SJTS04]. Based on the α-mattes and camera parameters a coarse intermediate opacity hull [MPN*02] is created.

For each photograph for which no trimap was hand-drawn, a trimap is automatically generated from the already constructed intermediate opacity hull and the hand-drawn trimaps. Next, an α-matte is computed and used to refine the opacity hull. Once all recorded images are processed, the full detail opacity hull can be viewed from an arbitrary viewpoint.

Decoupling the acquisition process from a fixed setup opens up a whole new range of possibilities for on-field 3D photography. Human faces (including hair) and large outdoor objects can now be captured and represented as opacity hulls. Additionally, the sampling density can be more easily adapted to the complexity of the objects.

2 Related Work

Numerous techniques exist to acquire the geometry of an object, without the need for an expensive calibrated setup (e.g. [Pol99, RHHL02]). Although important, shape alone is not sufficient to create realistically looking objects. Texture, micro-geometrical detail, self-shadowing, opacity and the view-dependency of reflections play an important role in providing a believable visualization. We denote these factors plainly as 'appearance'. Image-based rendering techniques try to capture an object solely by appearance. These techniques regard the object as a black box system, and capture enough information to let the system appear like the object ([GGSC96, LH96]). Image-based rendering techniques only acquire the global shape of the object implicitly, since the object can be viewed from any position. Due to the lack of an explicit global geometrical model, ghosting artefacts can occur during rendering.

Since exclusively capturing either shape or appearance fails to provide visually pleasing results, many techniques have been developed to capture both simultaneously. [Deb96] captures the geometry of buildings and employs view-dependent texturing to model fine details. Similarly, [YSK*02] requires a global geometrical model augmented with micro-billboards to represent fine geometrical details. Many Surface Light Field techniques have been proposed to represent the appearance of an object on a

given polygon model [WAA*00, CBCG02]. The unstructured Lumigraph rendering technique [BBM*01] generalized image-based rendering techniques which use a coarse geometrical model. Shum et al. [SSY*04] introduced "Pop-up Light Fields", which models a Light Field using a set of planar layers. These layers pop up depending on the scene complexity. In [MPN*02], objects are represented using opacity hulls. Unlike the previous techniques, opacity hulls do not use a polygon mesh to model the shape of the object, but a surfel representation. This method enables to capture and represent a wide variety of effects, including fuzzy edges and complex fine geometry. As a downside, most of these techniques require expensive setups, or massive amounts of photographs, or are only usable in lab conditions.

Recently, techniques have been developed to acquire the shape and appearance of geometrically ill-defined objects in a very practical manner. However, these methods are specifically geared towards one type of object. An example of such a technique is the capture and visualization of trees [RMMD04].

In this report we represent the shape and appearance of a real object by an opacity hull [MPN*02]. The advantage of using opacity hulls is that it is a flexible solution which can represent a wide range of object types. The opacity hull is a surfel model of the visual hull augmented with view-dependent opacity and color. More formally, an opacity hull is defined as the intersection of the view-cones defined by the α-mattes of each photograph and their respective camera parameters.

Matusik et al. [MPN*02] extract an α-matte by displaying multiple sine wave patterns on plasma screens placed behind and below the object. Although very effective, this requires an elaborate setup and multiple photographs to extract an α-matte for a single viewpoint. Recent advances in natural image matting allows to extract an α-matte from a single image using a user defined trimap [RT00, CCSS01, SJTS04]. We will employ these techniques to pull an α-matte for each photograph. Another limitation of the technique introduced by Matusik et al. is that the setup constrains the size of the objects which can be acquired. Furthermore, building such a setup requires elaborate calibration.

In this report we address these acquisition issues by using a free-form acquisition approach. We only focus on the acquisition of opacity hulls of real objects under static illumination.

3 Overview of the Technique

Our technique consists of four steps to acquire and compute an opacity hull of a real object:

1. **Acquisition:** The object is photographed from freely chosen camera positions. For each photograph the intrinsic and extrinsic camera parameters are estimated (section 4).

2. **Initial Coarse Hull**: Next, a coarse opacity hull is constructed from a sparse subset of photographs for which the user manually draws trimaps. For each photograph/trimap couple an α-matte is extracted. An iterative algorithm (section 5.1) is used to compute a coarse visual hull. This step is schematically depicted in figure 1.

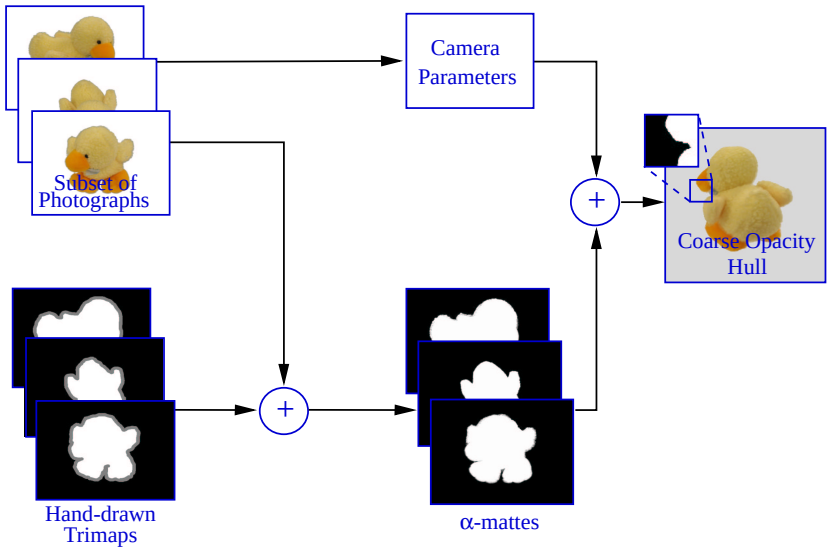

Figure 1: A schematic overview of the construction of an opacity hull. For each photograph an α-matte is extracted using a hand-drawn trimap (if available). These α-mattes, together with the corresponding camera parameters are used to define a coarse opacity hull.

3. **Refinement**: For each unprocessed photograph for which no trimap is hand-drawn, a novel trimap is computed from the already computed visual hull and trimaps (section 5.2). For each generated trimap an α-matte is computed, which is subsequently used to refine the visual hull (using the iterative visual hull algorithm of section 5.2). A schematic overview is shown in figure 2. This process is repeated until all α-mattes are computed and added to the visual hull.

4. **Adding Color and Opacity**: Finally, the computed visual hull is converted into an opacity hull by adding the view-dependent opacity values and color information (section 5.3).

The resulting opacity hull is a surfel model, which for each surfel has an associated appearance vector. This appearance vector contains view-dependent color and opacity information.

4 Acquisition and Camera Calibration

The acquisition process consists of recording photographs of the object from freely chosen positions, which is a simple and easy task. We record 50 to 300 photographs depending on the desired accuracy and the complexity of the shape or appearance of the object.

The intrinsic and extrinsic camera parameters are then automatically estimated [HZ04]:

- If the scene contains enough feature points (i.e. identifiable points between two

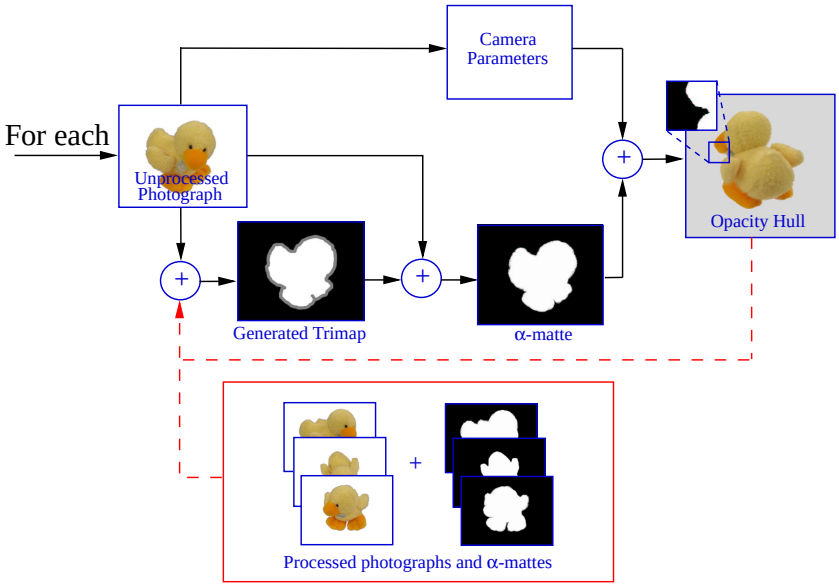

Figure 2: A schematic overview of the refinement of the opacity hull. For each unprocessed photograph a trimap is computed based on the coarse opacity hull and the already computed α-mattes and their respective photographs. From this generated trimap and the corresponding photograph an α-matte is extracted. This α-matte is then used to further refine the opacity hull. This process is repeated until all photographs are used.

or more images), then the camera parameters can be automatically computed from the raw images.

- If the scene does not contain enough feature points (i.e. when the automatic camera calibration fails) then we resort to placing marker objects in the scene during the photo-shoot. From these markers the extrinsic camera parameters are estimated. A similar approach was used in [RMMD04].

Once the camera parameters are estimated, the opacity hull can be computed.

5 Opacity Hull Construction

In this section we discuss the computation of the opacity hull, given the photographs and the user-defined trimaps.

In our technique we use Bayesian matting [CCSS01] to compute an α-matte for each photograph given a trimap (see appendix A for short overview of Bayesian Matting). Bayesian matting computes an α-matte directly from the trimap, without requiring additional user interaction. Recently Poisson matting [SJTS04] has been introduced. However it is not suited for our application because it requires significant user interaction to fine-tune each individual α-matte.

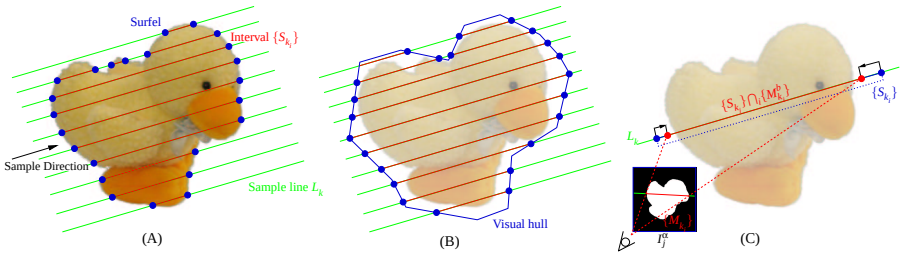

Figure 3: An illustration of the iterative construction of a visual hull. In sub-figure (A), the sampling of an object to a surfel model is depicted. The surfel model (blue dots) of the object can be equivalently represented by a set of intervals $\{S_{k_i}\}$ (red lines) along the sample lines L_k (green lines). Since the exact geometry of the object is unknown in our application, an interval representation of the visual hull is used (sub-figure (B)). This representation can be easily updated, given a novel α-matte I^α_j, by intersecting the intervals $\{S_{k_i}\}$ and the back-projected matte intervals $\{M^b_{k_i}\}$ (sub-figure (C)).

Since the user only draws trimaps for a subset of the photographs, an algorithm is required to generate trimaps for the remaining photographs. We present an algorithm that uses a coarse visual hull and the already computed trimaps to derive new trimaps (section 5.2).

An iterative algorithm is developed to avoid recomputation of the visual hull each time a new trimap (and consequently an α-matte) is available. The details of this iterative algorithm are discussed in section 5.1.

Finally, the computation of the opacity hull from the iteratively constructed visual hull is detailed in section 5.3.

5.1 Iterative Algorithm for Shape Reconstruction

In this section we develop an iterative algorithm to compute a visual hull from a set of α-mattes and corresponding camera parameters. To facilitate the iterative computation of the visual hull, a specialized representation is required. Additionally, this representation must be easy to convert into a surfel-sampled visual hull and thus an opacity hull.

Our representation is inspired on the sampling of a polygon geometry (in our case the visual hull) into a surfel model. In [PZvBG00] a polygon model is regularly sampled along the directions parallel to the three major axes. Each sample (surfel) is the intersection of the object and a half line with origin on a regular grid perpendicular to the sample direction. We call such a half line 'sample line' and denote it by L_k. The set of intersections associated with a sample line L_k can be seen as a set of intervals $\{S_{k_i}\}$ indicating the interior of the object along L_k. Each interval S_{k_i} is equivalent to two surfels (one at the begin point and one at the end point of the interval located on the sample line L_k). An overview of this is illustrated in figure 3A. We will show that keeping the intervals along a sample line is a suitable representation for iteratively constructing a surfel-sampled visual hull. This representation is equivalent to a great

Init: A set of samples lines $\{L_k\}$, each with an maximal interval $\{S_{k_i}\}$ associated.

for each photograph I_j
{
compute the α-matte I_j^α *(Bayesian matting)*
for each sample line L_k
{
project L_k on I_j^α and compute the set of intersections $\{M_{k_i}\}$
compute $\{M_{k_i}^b\}$, the back-projection of $\{M_{k_i}\}$ on L_k
update the set of intervals $\{S_{k_i}\}$ on L_k by $\{M_{k_i}^b\} \bigcap_i \{S_{k_i}\}$
}
}

Figure 4: A high level description of the iterative algorithm for constructing the shape.

extent to the LDC representation of [PZvBG00].

Suppose we have an interval representation of the coarse visual hull of an object (figure 3B). Given an α-matte, I_j^α, and the corresponding camera parameters, each sample line L_k is projected onto the α-matte I_j^α. The intersecting intervals $\{M_{k_i}\}$ of the projection of L_k with the α-matte I_j^α are determined and subsequently back-projected onto the sample line L_k (denoted as $\{M_{k_i}^b\}$). Each interval S_{k_i} needs to be adjusted so that it is contained in $\{M_{k_i}^b\}$ to keep the visual hull consistent with the new α-matte I_j^α. More formally we compute $\{M_{k_i}^b\} \bigcap_i \{S_{k_i}\}$. The refinement process is illustrated in figure 3C and a high level algorithm is shown in figure 4. We repeat this process for all sample lines.

To bootstrap the algorithm, the initial (very) coarse visual hull is set to the bounding box of the object and a sample line density is selected (e.g. 256×256) depending on the desired opacity hull quality.

5.2 Automatic Trimap Generation

To compute an α-matte from a photograph, a trimap is needed. Drawing a trimap for each photograph is time consuming. Therefore, we would like to minimize the number of trimaps that the user has to draw.

In video matting [CAC*02], a technique is presented to derive a trimap from other trimaps. This method is based on optical flow, which works optimally when the change between consecutive images is equally distributed (i.e. constant time steps) and small. However, in our free-form approach the change between photographs is not necessarily equally distributed or small, and thus another solution needs to be found.

Shum et al. [SSY*04] introduced coherence matting, a method similar to Bayesian matting, except that the background color is determined by reconstructing the background from multiple photographs. However, this method does not work well when

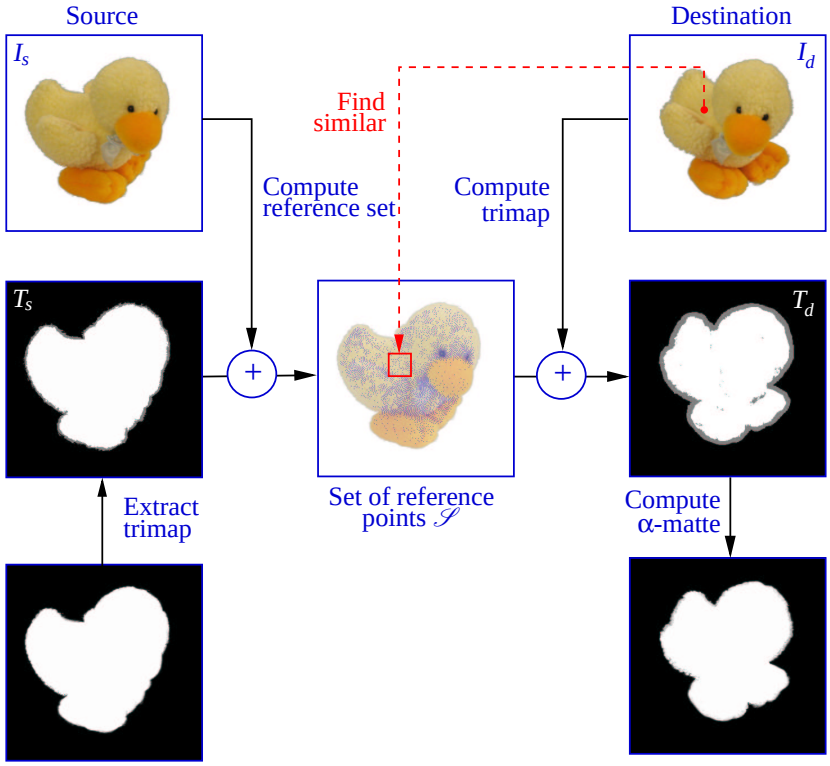

Figure 5: Given a source image I_s and respective trimap T_s, a trimap T_d can be computed for a destination image I_d. The key idea is to construct a set of reference points $\mathscr{S}$ of I_s that are representative for the foreground pixels in I_s. All pixels of I_d that match a reference point in $\mathscr{S}$, are marked as foreground in the destination trimap T_d.

the background contains non-lambertian surfaces, or surfaces with a high frequency texture. In our free-form approach, no assumptions can be made on the properties of the surfaces in the background. Furthermore the camera positions in [SSY*04] are positioned on a plane, which greatly eases the reconstruction of the background, as opposed to our method.

We can make four observations concerning trimaps and opacity hulls:

1. Matting techniques can usually handle small 'holes' (unknown regions) in the foreground (representing the object) or background markings of the trimap as long as there is enough information left in the fore- or background markings around the hole.

2. A visual hull encapsulates the object. It is always larger than the real object. The outside of a visual hull projected onto the image plane is guaranteed to contain only background pixels.

3. If two photographs (recorded at different viewpoints) of the same scene have pixels that are similar, then there is a large probability that they will have the same trimap classification. Furthermore a one-to-one mapping of similar pixels is not required, as long as all similar pixels have the same trimap designation.

4. The coarse visual hull gives a reasonable idea (neighborhood) where a pixel from one image maps into another image.

Using these four observations, we can now formulate our trimap computation algorithm. Assume we have two images, a source image I_s and a destination image I_d. We want to derive the trimap T_d of I_d, given a trimap T_s of I_s. We start by initializing T_d as background. Next, the intermediate visual hull is projected onto T_d and each covered pixel is marked as unknown (observation 2). Since the visual hull encapsulates the object, a number of these unknown marked pixels can be designated as foreground.

To find these pixels, we first create a set $\mathscr{S}$ of foreground marked points of I_s that are not similar to any unknown marked pixels in I_s. Next, we mark any unknown point in T_d as foreground, if its corresponding pixel in I_d is similar to a point in the set $\mathscr{S}$ (observations 1 and 3) (figure 5).

The search process of finding a similar pixel in $\mathscr{S}$ can be expedited in two ways:

- $\mathscr{S}$ can be pruned to only include points that are not similar to each other, which reduces the size of the search space (see also figure 5: the blue dots in $\mathscr{S}$).

- The coarse hull provides a reasonable guess in which area we can find a similar pixel. We used an area of 32×32 pixels around the projected position of the pixel from I_d to I_s (observation 4).

Two questions remain: how do we define 'similar' and which source trimaps T_s and photographs I_s do we use given a specific destination photograph I_d.

There are different ways to determine if two pixels are similar to each other. In our implementation we used the pixel consistency method proposed by Sand and Teller [ST04], but other methods such as cross-correlation can also be used.

For T_s and I_s we use the photographs recorded from the nearest N viewpoints for which a trimap is manually drawn by the user, since we have most confidence in the quality of α-matte of these photographs. To further enhance the quality of T_d we also use the nearest photograph for which a trimap is already computed.

We do not directly use the originally hand-drawn or generated trimaps for I_s, but extract a new trimap from the corresponding α-matte. This new trimap will have fewer pixels marked as unknown. To extract a trimap from a matte we mark a pixel as foreground if $\alpha > 95\%$, background when $\alpha < 5\%$ or otherwise as unknown.

A high level desciption of the algorithm is shown in figure 6.

5.3 Assigning Opacity and Color

Once all α-mattes are computed and added to the visual hull, the final opacity hull can be computed. First, a point cloud is created by converting each interval into two surfels (begin and end point of the interval). The normals of the surfels can be computed by taking the smallest principal component of the N closest distance weighted neighboring

Input: Source image I_s and α-matte I_s^α.
Destination image I_d.
Output: The trimap T_d and α-matte I_d^α associated with I_d.

(1) compute T_s from I_s^α: $\begin{cases} \textit{foreground} & \text{if } \alpha_s > 95\% \\ \textit{background} & \text{if } \alpha_s < 5\% \\ \textit{unknown} & \text{otherwise} \end{cases}$
(2) compute $\mathscr{S}$ as the set of pixels from I_s, marked foreground in T_s and **not similar** to a pixel in I_s marked unknown in T_s.
(3) prune $\mathscr{S}$ by removing all pixels which are **similar** to eachother.
(4) mark T_d completely *background*.
(5) project the visual hull on T_d and mark covered pixels as *unknown*.
(6) **for** each pixel p in I_d and marked *unknown* in T_d
{
(6a) compute p_s by projecting p to I_s using the visual hull.
(6b) mark p *foregournd* in T_d if p is **similar** to a pixel in the neighborhood of p_s in $\mathscr{S}$.
}
(7) compute I_d^α from T_d *(Bayesian Matting)*.

Figure 6: A high level description of the automatic trimap generation algorithm.

points ([ABCO*03, section 3.2]). Next, the appearance and opacity are found by projecting each surfel back into the foreground image and respective α-matte. These color and opacity values are stacked into a vector and stored per surfel. Thus each surfel contains an appearance vector defined by the view-dependent color and opacity values.

6 Compression and Visualization

To visualize the opacity hull, any suitable surfel rendering algorithm can be used, but instead of having a fixed color per surfel, an appearance function needs to be evaluated. A spherical Delaunay triangulation based on the sampled directions (i.e. viewpoints for which a photograph is recorded) is created. To evaluate an appearance function for a viewpoint, a barycentric weighting based on this triangulation is computed. This weighting is similar to [PCD*97]. To retain rendering speed, the spherical Delaunay triangulation is precomputed using the algorithm described in [PWH04]. This precomputation is possible since all appearance vectors have an identical Delaunay triangulation. Using a precomputed Delaunay triangulation is only correct for distant cameras.

Special care has to be taken when dealing with models that were acquired by an incomplete sampling of the set of camera positions (e.g. a single circular pass around the object). The renderer can restrict the user to view the object only from view directions that lie within a maximum range of the measured sample directions. Alternatively, if the distribution of sample directions is large enough, the renderer can switch to a constant color (the color of the closest sample direction or the average color of the appearance function) when the user deviates too much. Our visualization runs at

interactive speeds (3 to 12 fps) on a budget graphics card (NVidia Geforce FX 5200).

The storage requirements of the models obtained by our methods range from 250 to 1000 megabytes, depending on the number of surfels used (100.000 to 300.000), and the length of the appearance vectors (50 to 300). We would like to reduce these storage requirements without quality loss. Furthermore, we would also like to minimize the memory requirements during visualization. To retain fast rendering, a compression scheme is needed that can be decompressed on the fly during rendering without much overhead.

In [MPN*02] the appearance of each surfel is computed during visualization. Each surfel is back-projected in the 4 photographs recorded from the viewpoint closest to the current visualization viewpiont. The weighted sum of the 4 values corresponding to the back-projected pixels are used to splat the surfel. The photographs are compressed using a PCA scheme. There are two problems with this approach. First, each surfel needs to be back-projected during rendering, requiring additional overhead. Second, using PCA on a single image neglects inter-image relations to increase the compression ratios.

In "Opacity Light Fields" [VPM*03], hardware accelated techniques are presented for rendering the opacity hulls of [MPN*02]. Impressive compression ratios are obtained, but at the cost of reduced rendering quality and increased sensitivity with respect to inaccuracies in the geometry. The calibration condition in which our technique can be applied are less optimal then the conditions in [MPN*02]. Thus, the probability of having a less accurate calibration is larger, forcing us to use a more robust rendering and compression method.

Clustered principal component analysis (CPCA) [KL97], first introduced in the field of computer graphics by Sloan et al. [SHHS03], is a non-linear dimension reduction technique that builds a locally linear model of the data. CPCA is an ideal tool for compressing the appearance vectors of the captured data, because it can achieve a significant reduction in storage requirements and still is very fast to decompress. We cannot transform our appearance functions into a local coordinate system or use spherical harmonics as is done by Sloan et al., because of the irregular sampling. However, we can still use CPCA directly on the appearance vectors. A compression ratio of $1/20$ still gives good visual results. The average reduced modelsize is 20 to 50 megabytes.

Storing the appearance vectors explicitly at each surfel has a number of advantages. First, no back-projection is needed, allowing faster visualization. Second, an extra dimension is taken advantage of during compression by using CPCA directly on the appearance vectors. The cost of decompressing is about the same as in the approach of [MPN*02].

7 Results

In figure 7, a visualization of an opacity hull of a large statue is shown for two new viewpoints. Although it does not contain many opaque features, it illustrates that our method is not limited by the size of an acquisition setup. The statue itself is quite large and had to be captured outdoors. We only circled the statue once and recorded approximately 150 photographs. For only 20 of these photographs, a trimap was drawn manually. The constructed opacity hull contains 100.000 surfels.

Example	Figure	#Photographs	#Hand-drawn trimaps	#surfels
Statue	7	150	20	100.000
Kachina Doll	8	125	25	200.000
Duck	9	300	50	80.000
Head	10	30	30	300.000

Table 1: A summary of the results in figures 7, 8, 9 and 10

The necessity for appearance capture is demonstrated in figure 9. This duck model has a simple macro-geometry, but a complex micro-geometry (i.e. the bumps on its body). For capturing these micro-geometrical features exactly, a very accurate 3D scanner is required. The opacity hull represents the micro-geometry by means of the view-dependent appearance functions. We captured the complete model in approximately 300 photographs. For a small subset of 50 photographs, a trimap was drawn by hand. The opacity hull contains 80.000 surfels, which are sufficient to represent the simple macro-geometry.

The Kachina doll in figure 8 would have been difficult to capture using a geometry-based method. The model features many fine opaque details such as hair. Details of the opacity values are depicted on both sides. We captured this model by a single sweep around the objects in approximately 125 photographs. The resulting opacity hull contains 200.000 surfels.

In figure 10 different visualizations of the opacity hulls of two human heads are shown. Note that extruding strands of hair and the fine details of the glasses are captured faithfully. For each head we recorded a low number of photographs (approximately 30). We moved a single camera around to capture the heads. The recording process is short enough, since no calibration is required during the photoshoot, for the subjects not to move. Because we only recorded such a low number of photographs, all trimaps needed to be drawn manually. Both computed opacity hulls contain 300.000 surfels.

Table 1 summarizes the results.

8 Discussion and Comparison

A key component of our system is the automatic creation of trimaps. As a rule of thumb we draw trimaps for viewpoints 20 to 30 degrees apart. This results in acceptable trimaps and extracted α-mattes. We also found that the pixel consistency method of Sand et al. [ST04] is less effective when the rotation axis and the view axis are not orthogonal, i.e. when the rotation occurs in the view plane. Therefore more trimaps are drawn for photographs recorded from above the object. Automatic generation of the trimaps will most likely fail on objects which are hard to matte using hand-drawn trimaps (e.g. objects for which opaque parts have a different color than the solid parts). As a backup, it is always possible to draw more or even all trimaps when the automatic generation fails or does not deliver the desired quality. Note that this would only affect the amount of manual work but does not influence the acquisition process.

To capture fine geometrical details, exact camera parameter estimation is required. A slight inaccuracy in the camera calibration can result in small features being cut

away. A similar problem can occur when the object is not completely static (e.g. the wind blowing on a tree). Nevertheless, we found the method to be robust with respect to inaccurate calibration. Only very small features are cut away when the calibration is not perfect. Large details are only cut away when the miscalibration is significant (error of $+10\%$).

Opacity hulls were originally introduced in the seminal work of Matusik et al. [MPN*02]. We summarize the main differences between our method and theirs:

- The presented method requires only $1/6$ of the photographs to achieve the same sampling density. [MPN*02] need 6 photographs of the object backlit by different sine wave patterns to extract an α-matte. In our technique, we use Bayesian matting [CCSS01] which only requires a single photograph. The quality of the α-mattes extracted by Bayesian matting is good, but less compared to the quality achieved by the approach of Matusik et al..
- The appearance vectors are explicitly associated with each surfel. No back-projection is required during visualization, as opposed to [MPN*02], which requires a back-projection of each surfel during visualization.
- By using CPCA for compressing the opacity hull, interviewpoint relations are taking in account.
- Our method does not require the construction and calibration of a data acquisition setup, resulting in a more flexible acquisition process.
- Due to the free-form data acquisition, a wider range of objects can be captured. Outdoor or large objects can be captured on location. Additionally the sampling density can be locally adapted to the captured object.
- [MPN*02] are also able to capture opacity hulls which can be re-illuminated. Our method only works under static illumination.

9 Conclusion and Future Work

In this report we presented a free-form acquisition method for capturing the shape and appearance of real objects. We use opacity hulls in order to limit the required number of photographs and to maximize the kind of objects that can be captured. To minimize user input, we developed a novel technique for generating trimaps from a sparse set of hand-drawn trimaps.

A novel algorithm was introduced to automatically compute trimaps from a sparse set of handdrawn trimaps. Additionally, an iterative method for computing a surfel representation of the visual hull directly was presented. The main advantages of the iterative method are modest memory consumption, not all photographs need to be loaded into the memory at once, and the possibility to use the coarse intermediate visual to do additional computations. Finally, associating the appearance vectors explicitly to a surfel, allows faster visualizations and better compression.

In future work we would like to improve the computation of the trimaps by using cross-image information more efficiently. We also would like to use this cross-image information to create better α-mattes.

We would like to extend our method to include relighting. To keep the acquisition practical in such a case is a major challenge.

Acknowledgments

We would like to thank Frank Verbiest for his help with the camera calibration of some of the scenes. Furthermore, we thank Karl vom Berge and Saskia Mordijck for allowing us to create an opacity hull of their head.

References

[ABCO*03] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics 9*, 1 (2003), 3–15.

[BBM*01] BUEHLER C., BOSSE M., MCMILLAN L., GORTLER S., COHEN M.: Unstructured lumigraph rendering. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM Press, pp. 425–432.

[CAC*02] CHUANG Y.-Y., AGARWALA A., CURLESS B., SALESIN D. H., SZELISKI R.: Video matting of complex scenes. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 243–248.

[CBCG02] CHEN W.-C., BOUGUET J.-Y., CHU M. H., GRZESZCZUK R.: Light field mapping: efficient representation and hardware rendering of surface light fields. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 447–456.

[CCSS01] CHUANG Y.-Y., CURLESS B., SALESIN D. H., SZELISKI R.: A Bayesian approach to digital matting. In *Proceedings of IEEE CVPR 2001* (December 2001), vol. 2, IEEE Computer Society, pp. 264–271.

[Deb96] DEBEVEC P. E.: *Modeling and Rendering Architecture from Photographs*. PhD thesis, University of California at Berkeley, Computer Science Division, Berkeley CA, 1996.

[GGSC96] GORTLER S. J., GRZESZCZUK R., SZELISKI R., COHEN M. F.: The lumigraph. In *SIGGRAPH 96, Computer Graphics Proceedings* (Aug. 1996), Rushmeier H., (Ed.), Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 43–54.

[HZ04] HARTLEY R. I., ZISSERMAN A.: *Multiple View Geometry in Computer Vision*, second ed. Cambridge University Press, ISBN: 0521540518, 2004.

[KL97] KAMBHATLA N., LEEN T. K.: Dimension reduction by local principal component analysis. *Neural Comput. 9*, 7 (1997), 1493–1516.

[LH96] LEVOY M., HANRAHAN P.: Light field rendering. In *SIGGRAPH 96 Conference Proceedings* (Aug. 1996), Rushmeier H., (Ed.), Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 31–42.

[MPN*02] MATUSIK W., PFISTER H., NGAN A., BEARDSLEY P., ZIEGLER R., MCMILLAN L.: Image-based 3D photography using opacity hulls. In *SIGGRAPH 2002 Conference Proceedings* (2002), Hughes J., (Ed.), Annual Conference Series, SIGGRAPH, ACM Press/ACM SIGGRAPH, pp. 427–437.

[PCD*97] PULLI K., COHEN M., DUCHAMP T., HOPPE H., SHAPIRO L., STUETZLE W.: View-based rendering: Visualizing real objects from scanned range and color data. In *Proc. 8th Eurographics Workshop on Rendering* (June 1997).

[Pol99] POLLEFEYS M.: *Self-calibration and metric 3D reconstruction from uncalibrated image sequences*. PhD thesis, Katholieke Universiteit Leuven, ESAT-PSI, Leuven, Belgium, 1999.

[PWH04] PANG W.-M., WONG T.-T., HENG P.-A.: Estimating light vectors in real time. *IEEE Computer Graphics and Applications 24*, 3 (2004), 36–43.

[PZvBG00] PFISTER H., ZWICKER M., VAN BAAR J., GROSS M.: Surfels: surface elements as rendering primitives. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 335–342.

[RHHL02] RUSINKIEWICZ S., HALL-HOLT O., LEVOY M.: Real-time 3D model acquisition. *ACM Transactions on Graphics 21*, 3 (July 2002), 438–446.

[RMMD04] RECHE-MARTINEZ A., MARTIN I., DRETTAKIS G.: Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Trans. Graph. 23*, 3 (2004), 720–727.

[RT00] RUZON M., TOMASI C.: Alpha estimation in natural images. In *Proceedings of IEEE CVPR 2000* (June 2000), vol. 2, IEEE Computer Society, pp. 18–25.

[SHHS03] SLOAN P.-P., HALL J., HART J., SNYDER J.: Clustered principal components for precomputed radiance transfer. *ACM Trans. Graph. 22*, 3 (2003), 382–391.

[SJTS04] SUN J., JIA J., TANG C.-K., SHUM H.-Y.: Poisson matting. *ACM Trans. Graph. 23*, 3 (2004), 315–321.

[SSY*04] SHUM H.-Y., SUN J., YAMAZAKI S., LI Y., TANG C.-K.: Pop-up light field: An interactive image-based modeling and rendering system. *ACM Trans. Graph. 23*, 2 (2004), 143–162.

[ST04] SAND P., TELLER S.: Video matching. *ACM Trans. Graph. 23*, 3 (2004), 592–599.

[VPM*03] VLASIC D., PFISTER H., MOLINOV S., GRZESZCZUK R., MATUSIK W.: Opacity light fields: interactive rendering of surface light fields with view-dependent opacity. In *SI3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics* (New York, NY, USA, 2003), ACM Press, pp. 65–74.

[WAA*00] WOOD D. N., AZUMA D. I., ALDINGER K., CURLESS B., DUCHAMP T., SALESIN D. H., STUETZLE W.: Surface light fields for 3D photography. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 287–296.

[YSK*02] YAMAZAKI S., SAGAWA R., KAWASAKI H., IKEUCHI K., SAKAUCHI M.: Microfacet billboarding. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering* (2002), Eurographics Association, pp. 169–180.

A Overview of Bayesian Matting

In this section a short overview of Bayesian matting is given. For a detailed overview we refer to the original paper of Chuang et al. [CCSS01].

All matting methods try to solve the *compositing equation* for each pixel:

$$C = \alpha F + (1-\alpha)B,$$

where C, F and B are respectively the composited pixel, the foreground pixel and the background pixel. α is called the opacity. Each matting method differs in the way it solve this equation and what it require as input.

In Bayesian matting only C and a trimap is known. The trimap is a user specified map that indicates what parts of C are definitely foreground ($\alpha = 1$), definitely background ($\alpha = 0$), or unknown ($\alpha \in [0..1]$). α, F and B are computed using a *maximum a posteriori* technique, formulated using a Bayesian framework:

$$argmax_{F,B,\alpha} P(F,B,\alpha|C) \quad = \quad argmax_{F,B,\alpha} L(C|F,B,\alpha) + L(F) + L(B) + L(\alpha),$$

where $L(\cdot)$ is the log likelihood of the probability. The matting problem is now reduced to specifying the log likelihoods.

$$L(C|F,B,\alpha) = -||C - \alpha F - (1-\alpha)B||^2/\sigma_C^2,$$

where σ_C^2 is the standard deviation of the camera noise in C. The foreground likelihood is defined by:

$$L(F) = -0.5\ (F-\overline{F})^T\ \Sigma_F^{-1}\ (F-\overline{F}),$$

where $\overline{F}$ is the average estimated foreground color in the neighborhood of F, and Σ_F is the covariance matrix in the same neighborhood. Both are defined as:

$$\begin{aligned}\overline{F} &= W^{-1}\sum_i w_i F_i \\ \Sigma_F &= W^{-1}\sum_i w_i\ (F_i-\overline{F})(F_i-\overline{F})^T,\end{aligned}$$

where $W = \sum_i w_i$ and w_i is a weighting function in terms of the distance to F. [CCSS01] uses a Gaussian falloff with a standard deviation equal to 8 pixels.

$L(B)$ is computed similarly as $L(F)$. $L(\alpha)$ is assumed to be constant. In [CCSS01] an efficient two-step iterative algorithm is used to solve for F, B and α. We refer the interested reader to the original paper.

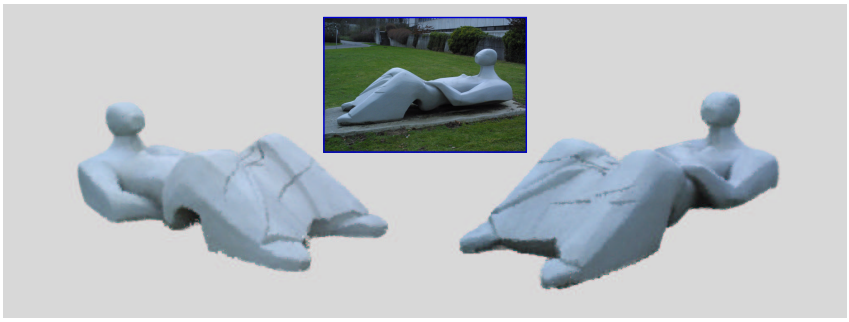

Figure 7: A large statue, captured outdoors. A reference photograph is shown in the middle.

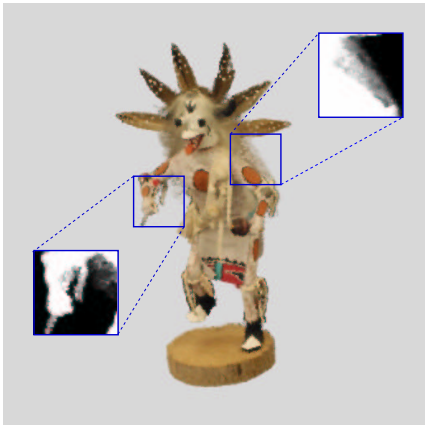

Figure 8: A Kachina doll with fine geometrical details and (opaque) hair, accompanied by two details of the opacity values.

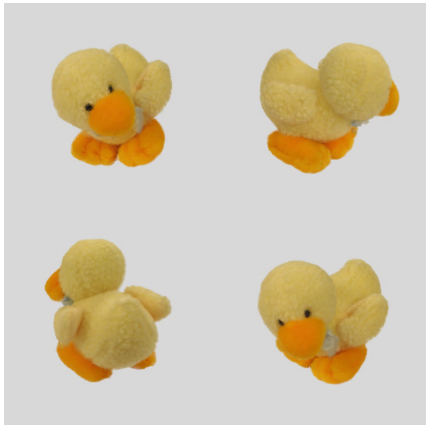

Figure 9: A duck with complex micro-geometry (bumps).

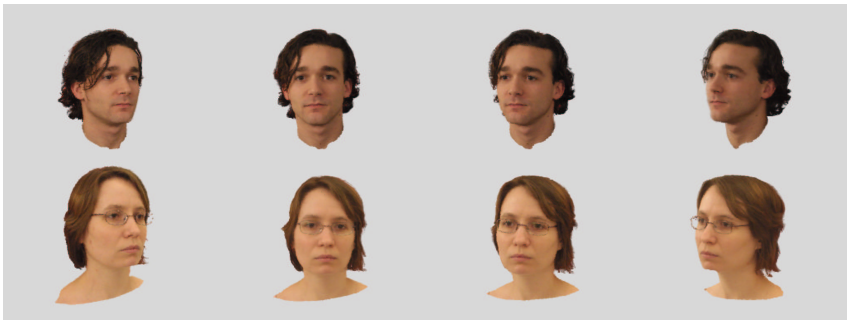

Figure 10: Two human heads visualized from different viewpoints. Note the fine detail in the hair and glasses.