Image-Based Rendering of Cars using Semantic Labels and Approximate Reflection Flow

Simon Rodriguez, Siddhant Prakash, Peter Hedman, George Drettakis

Image-Based Rendering of Cars using Semantic Labels and Approximate Reflection Flow

SIMON RODRIGUEZ, Université Côte d'Azur and Inria
SIDDHANT PRAKASH, Université Côte d'Azur and Inria
PETER HEDMAN, University College London
GEORGE DRETTAKIS, Université Côte d'Azur and Inria

Fig. 1. We propose a new solution for rendering captured cars, and in particular their reflective, semi-transparent windows. A textured mesh from multi-view stereo reconstruction (a) is missing the window geometry. Recent free-viewpoint Image-Based Rendering algorithms (b, c) are not designed to handle the rendering of both the reflection of blue sky, green leaves and the transmissive content (car interior). Our method (d) handles this by computing real-time reflection flows on an ellipsoid approximation of the curved window surface, based on our estimate of a smooth hull of the car that exploit *semantic labels* in the input images. The effect of reflection rendering is best seen in the supplemental videos.

Image-Based Rendering (IBR) has made impressive progress towards highly realistic, interactive 3D navigation for many scenes, including cityscapes. However, cars are ubiquitous in such scenes; multi-view stereo reconstruction provides proxy geometry for IBR, but has difficulty with shiny car bodies, and leaves holes in place of reflective, semi-transparent windows on cars. We present a new approach allowing free-viewpoint IBR of cars based on an approximate *analytic reflection flow* computation on curved windows. Our method has three components: a refinement step of reconstructed car geometry guided by *semantic labels*, that provides an initial approximation for missing window surfaces and a smooth completed car hull; an efficient reflection flow computation using an ellipsoid approximation of the curved car windows that runs in real-time in a shader and a reflection/background layer synthesis solution. These components allow plausible rendering of reflective, semi-transparent windows in free viewpoint navigation. We show results on several scenes casually captured with a single consumer-level camera, demonstrating plausible car renderings with significant improvement in visual quality over previous methods.

CCS Concepts: • **Computing methodologies** → **Image-based rendering**; *Image processing*; *Reconstruction*.

Additional Key Words and Phrases: image-based rendering, semantic labeling, reflection rendering

Authors' addresses: Simon Rodriguez, Université Côte d'Azur and Inria, simon.rodriguez@inria.fr; Siddhant Prakash, Université Côte d'Azur and Inria, siddhant.prakash@inria.fr; Peter Hedman, University College London, p.hedman.14@alumni.ucl.ac.uk; George Drettakis, Université Côte d'Azur and Inria, george.drettakis@inria.fr.

ACM Reference Format:
Simon Rodriguez, Siddhant Prakash, Peter Hedman, and George Drettakis. 2020. Image-Based Rendering of Cars using Semantic Labels and Approximate Reflection Flow. *Proc. ACM Comput. Graph. Interact. Tech.* 3, 1 (May 2020), 17 pages. https://doi.org/10.1145/3384535

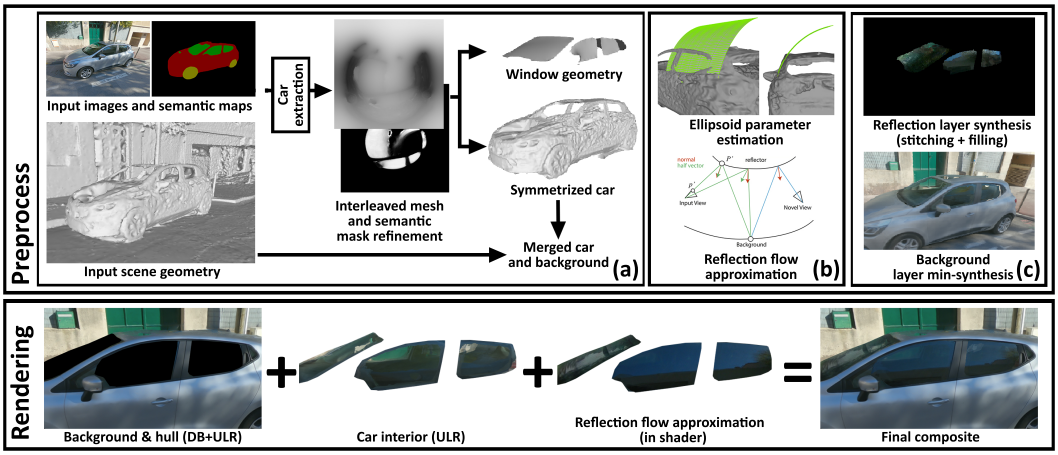


Fig. 2. Overview of our method. Top: (a) We isolate cars and refine the car mesh using a spherical projection, exploiting semantic labels to impose smoothing priors in window regions. Mesh smoothing is iteratively interleaved with semantic mask refinement, and approximate surfaces for windows are produced (Sec. 3). The second step (b) automatically fits an ellipsoid to the approximate window surface, which is used by our reflection flow computation (Sec. 4). The flow is also used in our final preprocessing step (c), together with the refined mesh to synthesize reflection and background layers (Sec. 5). Bottom: the layers and mesh are used during interactive navigation to synthesize novel viewpoints with plausible reflections, by computing reflection flow on the fly using the estimated parameters.

1 INTRODUCTION

Recent Image-Based Rendering (IBR) solutions only require simple capture using consumer cameras, and provide realistic free-viewpoint navigation at interactive rates in scenes such as cityscapes [Chaurasia et al. 2013; Hedman et al. 2018]. Car and car window rendering are arguably two of the main obstacles for using IBR for free-viewpoint street navigation. Existing solutions have difficulty with the poor reconstruction of shiny car bodies and the depth estimation ambiguity of reflections moving across curved semi-transparent windows. In this work, we provide a first *plausible* solution, by improving the overall rendered appearance of car bodies thanks to our estimation of smooth and filled car geometry, the believable motion of reflections on car windows and our synthesized reflection layers. We target a *lightweight* capture process with a single commodity camera (e.g., a GoPro), typically in a "street-side" fashion. In this context, recent IBR methods build on efficient Multi-View Stereo (MVS) algorithms [Reality 2018; Schönberger et al. 2016], that produce acceptable quality geometry for non-reflective/transmissive surfaces. The reconstructed geometry is used to reproject input images [Buehler et al. 2001; Hedman et al. 2018] in the novel view, allowing interactive, high-quality free viewpoint navigation in these cases.

However, for reflective car bodies these MVS algorithms produce inaccurate geometry due to strongly view-dependent and textureless appearance, and window surfaces are most often missing. Most IBR algorithms (e.g., [Chaurasia et al. 2013; Hedman et al. 2018]) are not designed to handle

the flow of reflections on a window which is different from that of the interior or background visible through that window. Previous specific solutions for IBR with reflections [Kopf et al. 2013; Sinha et al. 2012] have difficulty with curved surfaces of car windows, and with the novel views we target that are quite far from the input views, but quite common for street-level navigation.

IBR for reflections on cars raises three challenges outlined in Fig 2(a)-(c). We first need to provide car geometry that is as complete and smooth as possible, including a window surface. We then need to efficiently compute flow for reflections on windows taking their curved nature into account. Finally, we need to separate layers for reflections and background so we can flow them separately during free-viewpoint navigation. We successively address each of these challenges in our method; each step produces the input necessary to provide a solution for the next challenge.

For the first challenge we use semantic labels to identify cars and car windows in input images, using powerful modern machine learning-based segmentation. Unfortunately, these semantic labels are inaccurate and not multi-view consistent. Our key ideas are to use a spherical projection of the car and perform interleaved mesh smoothing and multi-view consistent label refinement in this spherical space. This space fits well with multi-view consistency operations, and facilitates the use of powerful image-processing methods. This step produces a smoothed and complete car body, including a first estimation of the window surface, see Fig. 2(a) and Sec. 3.

For the second challenge we introduce an efficient reflection flow computation based on analytic approximation of curved windows. We fit ellipsoids to each window by exploiting the previously estimated window geometry and provide efficient flow computation by gradient descent in the shader; Fig. 2(b) and Sec. 4.

For the third challenge, traditional layer separation algorithms (e.g., [Szeliski et al. 2000; Xue et al. 2015]) are not designed for the curved window surfaces and the low quality of the lightweight capture data we acquire. Instead, we introduce a plausible reflection layer *synthesis* algorithm. We use our approximate reflection flows as an initialization, and then use image stitching to complete the synthesis; Fig. 2(c) and Sec. 5.

In summary, we present three main contributions, providing a fully automatic solution for IBR of cars:

- A new algorithm to provide a complete and smooth car body reconstruction suitable for rendering in a casual capture context, as well as initial window surfaces.
- A reflection flow approximation for plausible interactive reflection rendering and an automatic ellipsoid fitting algorithm that uses the initial window surfaces.
- A reflection and background layer synthesis method building on our reflection flow.

Our solution allows interactive rendering of plausible motion of reflections in car windows, and diminishes visual artifacts due to missing and incorrect car geometry. This plausible motion greatly improves perceived visual quality compared to previous methods (Fig. 1, 10, 11 and supplemental videos) especially when moving around the scene.

2 RELATED WORK

Our approach focuses on Image-Based Rendering, layering and reconstruction of semi-transparent objects and the use of semantic information for 3D reconstruction. We review the most closely related work.

2.1 Image-based rendering

Image-based rendering generates novel views of a scene acquired as a series of photographs. Initial methods directly resampled light rays [Gortler et al. 1996; Levoy and Hanrahan 1996]; for sparser acquisition, a 3D mesh [Buehler et al. 2001; Debevec et al. 1998], can be used to allow smooth

blending of input views. Modern implementations of these techniques rely on Multi-View Stereo (MVS) reconstruction [Goesele et al. 2007], but suffer from their limitations. Despite recent progress [Reality 2018; Schönberger et al. 2016], these methods still have difficulty with scene coverage and are not designed to handle highly reflective surface nor the depth ambiguity of transparency.

Recent unstructured methods alleviate reliance on global geometry, using view-dependent segmentation [Chaurasia et al. 2013] or meshes [Hedman et al. 2016], recently using learning to improve blending on refined meshes [Hedman et al. 2018]. We build on such ideas, and introduce a solution for semi-transparent, reflective surfaces (e.g., car windows) that were previously problematic.

Specific techniques exist for scenes with reflective and transparent surfaces. Sinha et al. [2012] estimate up to two depths per pixel, producing two geometry proxies for separate image-based rendering of background and reflections. Kopf et al. [2013] perform image warping and blending in the space of color gradients, and integrate to get the final result. These approaches are restricted to planar reflectors and limited motion, and are unsuitable in our context of curved windows.

Penner and Zhang [2017] proposed the volumetric Soft3D approach to IBR. Reconstructed volumetric depth and color are interpolated and accumulated to generate the novel view, allowing multiple surfaces to be present at a given pixel; however, artifacts can appear in non-transparent regions, especially far from the input cameras poses. In contrast, we provide a larger space for free-viewpoint navigation, thanks to our approximate reflection rendering and layer synthesis.

Recent work on IBR [Mildenhall et al. 2019; Thies et al. 2019, 2020] exploits neural networks to improve rendering quality. Thies et al. [2020] address the problem of moving highlights, and focus on rendering of isolated objects rather than the full scenes we consider [Thies et al. 2019] but do not handle semi-transparent, reflective objects; Mildenhall et al [2019] focus on much smaller camera baselines than ours. Nonetheless, these methods present many exciting ideas on using the power of deep learning to improve IBR, and we consider these very promising avenues for future work.

2.2 Layering, Reconstruction, Rendering of Semi-Transparent and Reflective Objects

Layering. Separating reflective layers from the background is widely studied in computer vision and graphics. Single-image solutions can require user assistance [Levin and Weiss 2007], or specific acquisition hardware such as polarization [Wieschollek et al. 2018]. Multi-view methods [Szeliski et al. 2000; Xue et al. 2015] often jointly estimate the *flow* between the layers, as well as the layer decomposition itself. These methods require a sufficiently accurate initial flow estimate to be successful; in our context of uncertain input data, we opt to *synthesize plausible* reflection layers instead. Nonetheless, our synthesis method is inspired by the work of [Szeliski et al. 2000], and their idea of min-composite.

Reconstruction. Several methods address the challenge of reconstructing transparent and reflective objects, but typically require the use of custom acquisition devices [Whelan et al. 2018; Wu et al. 2018]. Godard et al. [2015] focus on specular objects and reconstruct them by refining the normals of an initial guess, guided by regenerated reflections. Numerous volumetric approaches can be found in the literature [Ihrke et al. 2010] to handle the multiple depths per pixel present in images of non-opaque objects. These often focus on specific object categories, or involve complex acquisition hardware setups [Ihrke et al. 2008]. We focus on a casual capture setup, using a single consumer-level camera.

Reflection Rendering. Reflections are often rendered with ray-tracing [Whitted 1980], but approximate approaches for rasterization pipelines (e.g., [Estalella et al. 2005; Ofek and Rappoport 1998]) search for the objects reflected from a curved surface so they can be rendered. Our IBR context is

different, since we do not have access to the full geometry, but our analytical surface approximation to curved windows allows us to build on such approaches to compute reflection flow.

2.3 Semantic labelling

Semantic labeling has been used to detect class-specific properties, allowing specific reconstruction processing. One common approach is the use of 3D models of a given object class (e.g., cars) to train priors for reconstruction, often resulting in good quality voxel reconstructions [Hane et al. 2014] or meshes [Yingze Bao et al. 2013]. While our object-class-specific smoothing priors have similarities in spirit with e.g., the class-based normal distributions of Häne et al. [2014], we focus on the use of a standard Structure-from-Motion (SfM)/MVS reconstruction pipeline, without the need for training based on 3D models.

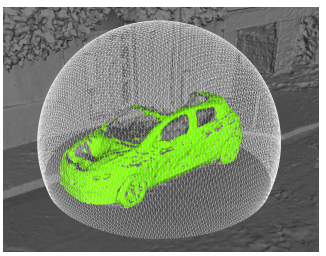
3 CAR GEOMETRY EXTRACTION AND REFINEMENT

To perform high quality rendering for cars including window reflections, we need to isolate the cars in the scene, refine their geometry, and estimate supporting geometry for the windows.

3.1 Isolating Cars with Semantic Labels

We will use semantic labels to identify and isolate cars in the scene; the labels will also be used to refine car geometry. We obtain 2D label maps for each input image, using DeepLab-v2 [Chen et al. 2017], trained on a subset of the ADE20K dataset [Zhou et al. 2017, 2019], to recognize car objects, but also *parts*, namely car body, car wheel and car window. Details of the training procedure are given in supplemental. An example training image and result obtained on one of our input images are shown in Fig. 3 (c,d).

We project the segmentation labels onto the geometry; vertices where at least 40% of the input semantic maps agree on the car label are considered as belonging to a car. We group these vertices in connected components using mesh and mask connectivity information. This extraction is robust due to the overlap between input images and the fact that the cars are the focus. For each remaining component we estimate a bounding box aligned with the main axes of the car using PCA, and initialize the bounding sphere used for the spherical projection (see figure on the right).

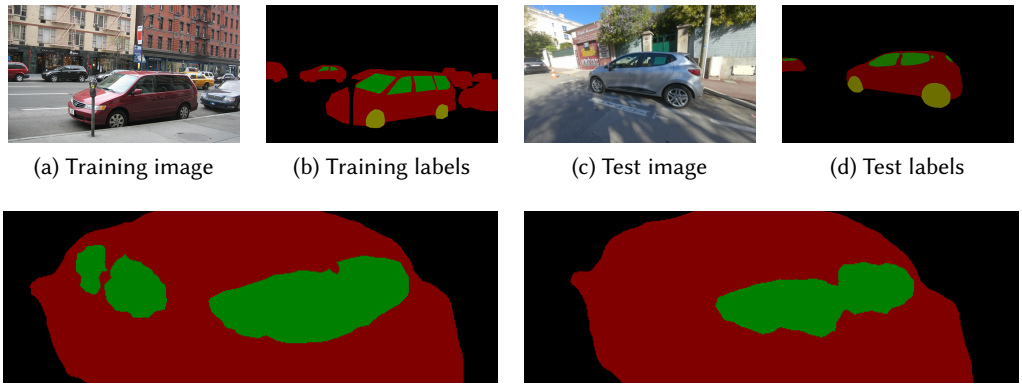


3.2 Smooth
Car Hull Extraction and Semantic Mesh Refinement

We want to estimate a smooth version of the car body with filled holes, obtain an initial approximation of the car window surfaces and improve overall car reconstruction. Unfortunately, car bodies are badly reconstructed by state-of-the-art MVS reconstruction algorithms, and car windows are often completely missing.

Semantic labels provide an indication of the location of the windows, and could serve as a guide to refine geometry. The segmentations sometimes contain errors, possibly because our viewpoints are very different from the training set images, e.g., close-ups where only a small region of the car is visible. Another case concerns objects lacking features, e.g. a black car in shadow, or with contradictory features caused by reflections, lead to missing regions in some predicted maps. In addition, labels are not always multi-view consistent, see Fig. 3(e), (f).

A precise model of the car and windows could help improve multi-view consistency, but this model is precisely what we are trying to obtain. To solve this dependency problem, we iteratively



(a) Training image (b) Training labels (c) Test image (d) Test labels

(e) Mask from first view (closeup) (f) Mask from second view (closeup)

Fig. 3. We train a deep-learning based segmentation classifier. Example of a training image (a) and associated hand-labeled segmentation map (b). Segmentation (d) obtained for one of our input images (c). The masks of two neighboring views (e) and (f) are not consistent.

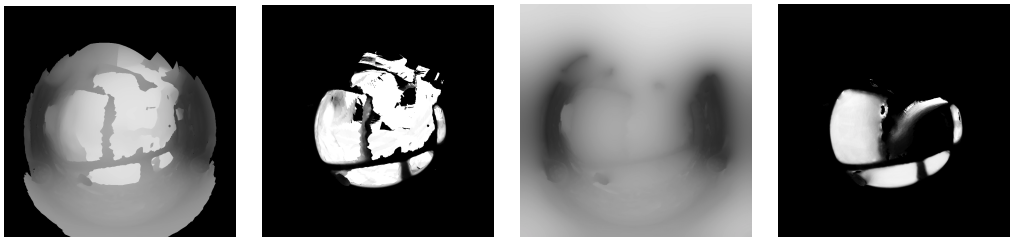
estimate geometry, interleaving mesh smoothing with updates of *semantic mask probabilities*, i.e., the probability that a pixel has a given label.

Smoothing the mesh in object space is difficult, since semantic labels from input views reproject incorrectly on the mesh through the window holes. However, we know that cars have a spherical topology, and it is natural to use a spherical projection of each car for multi-view consistent window segmentation. We choose to enforce spherical topology as a prior in "spherical" image space, which allows the use of efficient image-processing algorithms and facilitates multi-view coherence.

We start by projecting the geometry assigned to the car onto the bounding sphere using a spherical projection and create a depth map (Fig. 4(a)). We reproject the semantic "car window" labels into the same space, estimating a semantic label probability map using the reconstructed geometry (Fig. 4(b)). We next use the semantic labels to estimate a smooth car hull, fill the window holes and repair inaccurately reconstructed regions as much as possible. The semantic map is then refined by reprojecting the input view maps using the updated geometry. Akin to an Expectation-Minimization approach, we iterate in an interleaved fashion those two steps. Finally, we refine the semantic masks in a multi-view coherent manner.

Mesh Refinement Step. To refine depth, regions of the depth map where reprojected labels agree as "car window" are considered with low confidence, while regions that are seen by a high number of cameras without this label have a strong weight. Evidently, the missing window surfaces result in incorrect label reprojection; we prefer and smooth regions that are more likely to be correctly reconstructed car body, and fill the other regions with smooth propagation of the depth. A smoothness prior is thus applied to the entire hull of the car, and a data term is added on the Laplacian of the depth to encourage planar regions and counteract the tendency of the solver to pull the surface towards the sphere. We use a conjugate gradient method to solve this constrained optimization; details are provided in Appendix A. At the end of this step, we obtain a depth map with smoothed reflective surfaces and progressively filled-in window surfaces, Fig. 4(c).

Semantic Mask Probability Update Step. We can now use this smooth car hull geometry to reproject the label maps with updated visibility. This new label probability map (Fig. 4(d)) is of much better



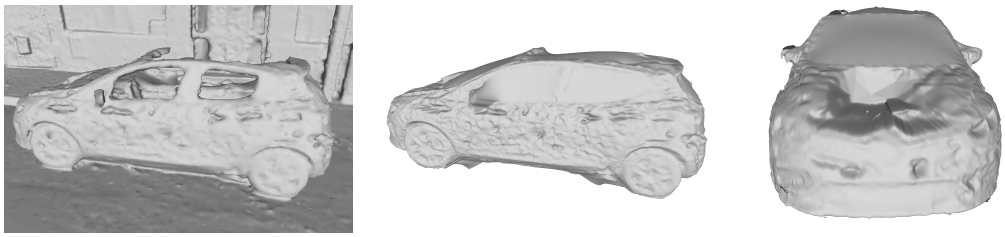
(a) Input depth map (b) Reprojected window label map (initial) (c) Smoothed depth map (d) Reprojected window map (refined)

Fig. 4. From the input disparity map (a), we refine a smoothed and filled disparity map (c), using the reprojected semantic information as a constraint (b). Semantic labels are then reprojected again (d).

quality than the reprojection using the original reconstructed geometry (Fig. 4(b)). The updated map is then used for the next mesh refinement step.

Final semantic mask refinement. After three interleaved iterations, we refine the label probability map in spherical space, using a Markov Random Field (MRF) guided by color similarity of input views reprojected onto the mesh and confidence, based on label probabilities and visibility.We also discourage well-reconstructed pixels to be labeled as windows. Details of the MRF are provided in Appendix B.

Smooth Mesh Extraction & Symmetrization. The final smooth depth map is used to regenerate a car mesh with filled windows and smoothed surfaces, Fig. 5(b). Regions that were previously holes and deformations caused by reflections and transparent surfaces (see Fig. 5 (a)) now have much smoother supporting geometry. In all the examples presented, we have used "street side"



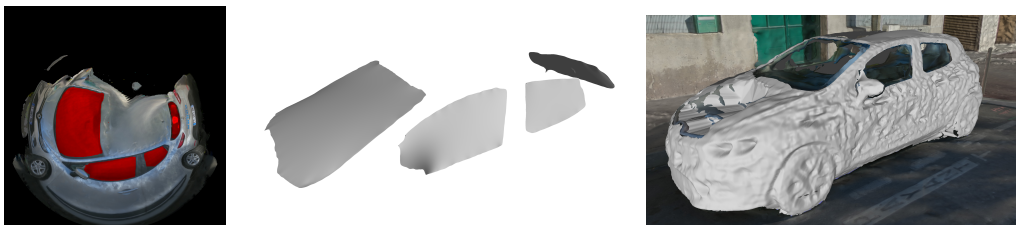
(a) Input geometry (b) Refined geometry (c) Symmetrized

Fig. 5. Left to right. (a) Original input geometry from the MVS reconstruction. (b) Refined geometry after the iterative mesh smoothing step. (c) Result of symmetrization. The far side of the car not seen by input cameras in our "street-side" casual capture is reconstructed by symmetry.

capture, with no photographs on the side of the car facing away from the street. We complete the missing information by generating a symmetrized version of the refined geometry. A copy of the car geometry is reflected along its principal vertical plane, and automatically re-aligned with the initial car mesh using an Iterative Closest Point approach. Both geometries are merged based on visibility, i.e., in regions where the initial car is visible in less than 20% of the cameras, we instead

use the mirrored version. The resulting refined and symmetrically completed mesh is shown in Fig. 5(c).

The final label map (Fig. 6 (a)) is then used to cut windows out of the smoothed mesh. Specifically, the parts of the mesh that are labeled as windows are extracted separately (Fig. 6 (b)), and the remaining hull of the car is merged back with the initial scene geometry (Fig. 6 (c)). Vertices of the initial geometry too close to the smoothed mesh are discarded, to avoid double surfaces. We refer to this windowless smoothed mesh (Fig. 6 (c)) as the *refined mesh* or geometry from now on.



(a) Final refined map (b) Extracted windows (c) Merged background and car hull

Fig. 6. At the end of the MRF step we obtain the final window mask (a, in red, drawn over the reprojected texture). This map is used to separate the window meshes (b) from the merged background/interior and car hull meshes (c).

4 ELLIPSOID APPROXIMATION FOR REFLECTION FLOW COMPUTATION

At a given pixel p in the novel view (Fig. 7(a)), we see a point P_r that is reflected from the background onto the reflecting window at P. We need to find the pixel in a reference input view that contains the reflection of P_r. We propose an efficient algorithm to explicitly compute reflection flow between views. This technique will be used both during preprocess and during rendering. We achieve this using two simplifying assumptions: that scene geometry is distant and that windows can be approximated by ellipsoid geometry.

4.1 Reflection flow computation

We assume that the scene geometry reflected in the windows can be approximated by the bounding sphere of the scene. Far-away geometry creates very small parallax when the camera moves, and convex reflectors further decrease the parallax. High quality geometry of the reflected objects is thus not required. Furthermore, each window is approximated by an ellipsoid. This representation will be key to making the problem tractable and achieving real-time performance. At pixel p, we see a point P on the surface of a window. We know the position, surface normal and window parameters at P. Knowing the novel view position, we can compute the reflected ray under a perfect mirror assumption, $\mathbf{r}$. The intersection of this ray with the background at P_r (see Fig. 7(a)), can also be derived analytically using our spherical world assumption.

We then search for the point P' on the ellipsoid reflector surface such that P_r falls inside the input view after being reflected at this point. This point has the property that the normal at the surface and the half vector between the incoming and outgoing reflected rays coincide [Estalella et al. 2005] (see Fig. 7(b)). We find this point using an approximate gradient descent. At a given candidate point P_c, we compute the half-vector between P_cP_r and P_cP_i (where P_i is the location of the input view). If $P_c = P'$ this vector is equal to the normal to the ellipsoid at P_c. Else we update the normal by shifting it toward the half-vector. Thanks to the bijection between ellipsoid positions

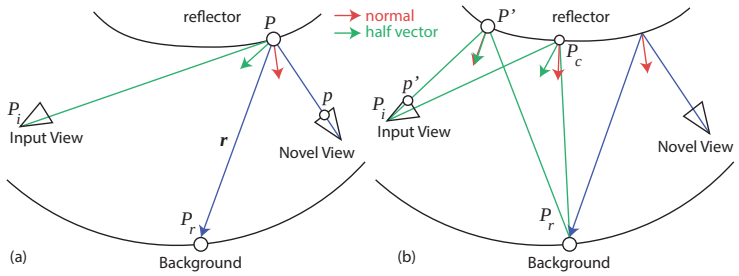


Fig. 7. (a) The initial configuration for the reflection flow computation. (b) Two steps of the gradient descent. When the half-vector (green) is aligned with the normal (red), we have found the point corresponding to the reflection in the input view. The value of pixel p' can be used to flow the reflection to the novel view.

and their normals, we can easily convert this updated normal to the corresponding new P_c. We can iterate this process until we find P' ; following this update procedure guarantees that we reach the correct solution [Estalella et al. 2005]. In practice, the algorithm requires fewer than 30 steps to convergence, and can be performed very efficiently in a shader (Algo. 1).

Algorithm 1 Iteratively compute P', assuming an axis-aligned ellipsoid for brevity

Input: P_i, P, P_r, ellipsoid center C and radii R_e
Output: P'
  $P_c \leftarrow P$
  $n \leftarrow normalize((P_c - C)/R_e)$
  for $i = 1$ to 30 **do**
    $h \leftarrow normalize(normalize(P_r - P_c) + normalize(P_i - P_c))$
    $n \leftarrow normalize(n + 0.2(h - n))$
    $P_c \leftarrow C + R_e * n$
  end for
  $P' \leftarrow P_c$

Once the point P' is found, it can be reprojected in the input view. If it falls inside the window label mask, the corresponding pixel p' can be used as a source for reflection (Fig. 7). By doing this computation for all pixels of the novel view covering a window, we can compute the reflection flow to the input view.

This reflection flow computation will be used during the window ellipsoid parameter estimation as described in the next subsection. At runtime we estimate the flow of reflections of each window for a set of input views using the same algorithm.

4.2 Ellipsoid Fitting for Car Windows

Each window is approximated by an ellipsoid with longitudinal and vertical radii. Due to shape and physical constraints, the range of admissible curvatures for car windows is quite limited. We start from the "cut-out" window surface (Fig. 8 (a)), use the average normal of the window as the third ellipsoid axis, and the projection of the scene up vector onto the window surface as a vertical axis. The longitudinal axis is the cross product of the two previous directions.

Our method is based on feature matching and estimates the radii from the motion of reflections between close-by views of the same window. We fall back to dense image matching when such features are missing – reflections such as the sky or a uniform wall cause only a few moving

lines to appear across the window. In both cases, a range of parameters is swept and the set of radii that best explain the reflection motion is extracted. We use the same range of radii for all windows ($[1m, 40m]$) and sweep it using quadratic steps to sample small values more densely, as small changes to the radii only create noticeably different reflection motion if the radii are small (see supplemental for an illustration).

Every reflection is mixed with transmitted colors from the car interior or the scene background, making matching more complicated – yet we only need to estimate two parameters per window. The motion only has to be correctly estimated for a few pixels ; this low dimensionality combined with the range prior make the problem tractable. A detailed description of the method is provided in Appendix C.

While the ellipsoid fit is an approximation – since car windows can have small imperfections and normal variations [Jacquet et al. 2013] – the resulting flow is sufficiently plausible in our test scenes, and allows real-time performance. See Fig. 8 (b,c) for an example of fitted ellipsoid. We experimented with a planar reflector, but the field-of-view for the reflector is incorrect and results were significantly worse (see supplemental).

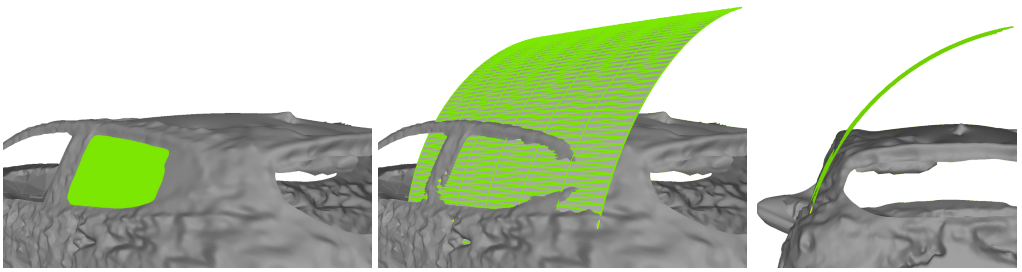


Fig. 8. (a) The extracted window surface (in green); (b) and (c) The result of the fitting process (in green).

5 SYNTHESIZING REFLECTION LAYERS

To render car windows, we need to combine a reflection layer, and the *background or transmitted layer* which corresponds to the car interior and the rest of the scene visible through the window. This layer is reprojected using the reconstructed geometry in the scene. The geometry is generally of good quality for the background, and very approximate for the car interior. We use the term *background flow* to refer to the reprojection of pixels using the refined geometry without windows. The *reflection layer* corresponds to the reflections on the windows, that move according to the flow we compute using our ellipsoid approximation.

Our data is insufficient to achieve accurate reflection layer decomposition; we thus choose to synthesize a *plausible* reflection layer that will be used at runtime. Inspired by min/max compositing [Szeliski et al. 2000], we use the min-composite of the reflection flows as a first estimate of the reflection layer, and synthesize a plausible layer by using a variant of image stitching techniques. Consider $l = 1..L$ layers over images I_i with $i = 1...N$. For a given image I_k, the min-composite $M_{l,k}$ for layer l is given as: $\mathbf{min}\mathcal{W}^l_{i\to k}(I_i),\ \forall i$, where $\mathcal{W}^l_{i\to k}$ is the warp or flow of layer l from image i to k. Since light is additive, $M_{l,k}$ is an upper bound on the value of layer l in image I_k.

For the background layer, we use the min-composite of the background flow directly, since this tends to reduce artifacts by preferring darker pixels, Fig. 9(a). Reliably reconstructing car interiors would significantly complicate the capture process: many images near the car are needed, and the far side must most often be captured, breaking the "street-side" capture context we target.

5.1 Reflection Layer Synthesis

For a given input view, we use the ellipsoid approximation for each window to compute reflection flow, reprojecting pixels from 25 neighboring views into it, to create the min-composite of the reflection layer, Fig. 9(b). This min composite has many visible artifacts: misalignment errors due to inaccurate reflection flow, the presence of moving objects (e.g., the photographer), artifacts due to errors in the mask reprojection and color harmonization discontinuities. The flow of the background layer can be even more approximate, since car interiors have very little reconstructed geometry. We thus see that the standard layer separation approach [Szeliski et al. 2000] cannot directly be applied in our context.

Instead, we synthesize a plausible reflection layer by applying standard image stitching techniques [Agarwala et al. 2004] on the min-composite, using a MRF formulation [Kwatra et al. 2003]. We use the seam-hiding pairwise term described by Kwatra et al. and a custom per-pixel data cost for each source image s:

$$w_s(p) = L_s(p) + min_{s' \neq s} |C_s(p) - C_{s'}(p)| + 2 * \min(T, D(p)). \quad (1)$$

where $L_s(p)$ is the luminance of pixel p in image s, $C_s(p)$ its color, $D(p)$ its distance to the border and T is set to 1% of the image diagonal. The first term encourages the solver to prefer the minimum value and to remove outliers, the second measures photoconsistency as the smallest L_1 distance to colors fetched from other images (s'), and the third discourages using pixels close to image edges. The resulting stitch is shown in Fig. 9(c). Sky reflections often have color differences between the various input views, resulting in remaining color harmonization boundaries that we remove with a final automatic Poisson editing step (Fig. 9(d)). The stitched image is used as a data term for both colors and gradients ($w_{\text{gradient}} = 1, w_{\text{data}} = 0.01$). However, seams are discouraged by choosing the gradient closer to zero from the source images at both sides of the boundary. The final harmonized reflection layers are then saved to be used for rendering.

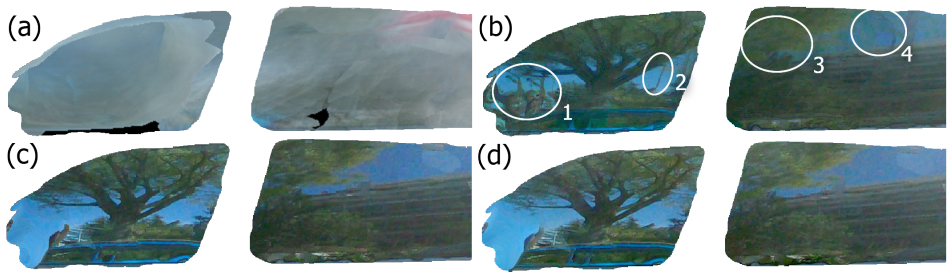


Fig. 9. (a) The min-composite for the background. (b) The min-composite of initial reflection flows from the 25 neighboring images to a given input view. Notice artifacts due to (1) presence of the photographer, (2) errors due to incorrect masks, (3) imprecise flows, and (4) color harmonization edges. (c) Our MRF stitching reduces alignment, motion and mask artifacts. (d) An additional Poisson image editing step reduces remaining composite and harmonization artifacts.

6 RENDERING, RESULTS & COMPARISONS

6.1 Rendering and Implementation

Rendering of a novel view proceeds in two steps. First we render the background of the scene, then we composite in the reflections computed using the reflection flow computation described above.

To render the background of the surrounding scene, we use Deep Blending [Hedman et al. 2018], and a per-pixel implementation of the ULR with a standard weighting scheme [Buehler et al. 2001],

Fig. 10. Our scenes, top to bottom: *Vine*, *Carpenter*, *Narrow-Street* and *Corner-Street*. Left to right: input images and semantic masks, scene geometry with the input viewpoints (in blue) and output viewpoints (pink and green), two renderings of each scene using our method. Note the distance from each novel view to the closest input viewpoint (displayed in overlay)).

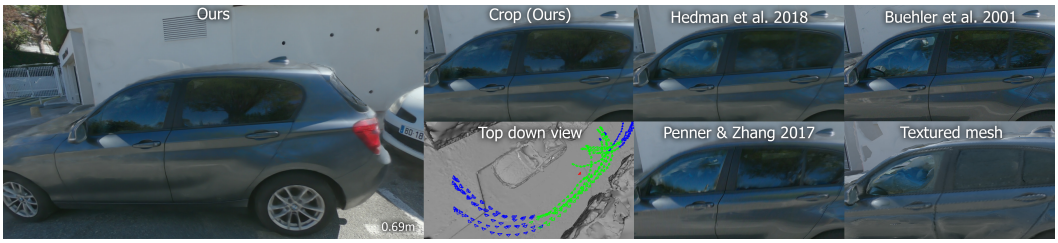


Fig. 11. Comparison with recent approaches. Note how our method maintains sharp and complete reflections. Please refer to supplemental materials and the accompanying video to see these results in motion.

reprojecting input images on the refined mesh. Since our scenes are large, we restrict the per-view mesh voxel grid for Deep Blending to encompass the cars, and use the per-pixel ULR for the rest of the scene.

We then render the interiors and background visible through the window regions, using per-pixel ULR to project the *transmission* layers onto the interior geometry, overwriting any previously rendered pixels

Finally, we render reflections by warping colors from the *reflection* layers using reflection flow. We only warp a subset of the input views, selecting the 50% views closest to the novel view, as

reflection layers further away won't contribute significantly to the final reflection. For each selected view, reflection flow is computed on the fly during rendering in a shader for each novel view pixel where the window supporting surface is visible. The background intersection and gradient descent on the ellipsoid are used to find the corresponding pixel in the input view. We additionally check that the warped pixel falls inside the same window region in the input view (using the supporting surface again). The warped reflection information of the selected views is blended, and the result is composited with the background using an alpha value of 0.75, which we found experimentally to work well with all scenes. We apply a small blending falloff at boundaries between the two regions.

We show results of our method with our unoptimized C++ and OpenGL implementation [1] All tests reported here were run on an Intel Xeon 5118 (48 logical cores) with 96GB of RAM and a NVIDIA GeForce RTX 2080 Ti.

6.2 Results

We present results on four scenes *Vine*, *Carpenter*, *Narrow-Street*, and the *Corner-Street* (Fig. 10), with the latter two containing two processed cars. The number of photos for each scene is respectively 177, 200, 360, and 330, captured using a GoPro (Hero 6) in burst more, giving 2 photos per second, while the user walks around the car 4 times at different heights. Capturing a car takes about 5 minutes. The input resolution (after camera calibration) is respectively 2720x1607, 2768x1639, 2864x1695 and 2200x1305. Car geometry extraction and ellipsoid estimation are performed using images resized to 1920px width.

Results are shown in Fig. 10, 11, in the supplemental material and the supplemental video. The effect of reflections is best perceived when moving the viewpoint; we strongly encourage the reader to view the videos. In the paths shown in the supplemental videos, our novel camera is on average at 0.75m from the closest input camera, with a maximum at around 1.65m.

Our interactive renderer shown in the video reaches 8Hz (120.0ms) at 1280x720 resolution. Rendering time is distributed as follows on average: background rendering using DeepBlending and ULR: 108.0ms, car interior rendering using ULR on the interior min-images: 3.0ms, reflection rendering and compositing: 8.5ms. Preprocessing for our method on a scene with 200 images takes 10min for mesh and segmentation refinement, 20min for the ellipsoid parameter estimation, 1h10min for reflection layer stitching and 10min for Poisson editing, in addition to standard off-the-shelf SfM/MVS (Colmap) and preprocessing for DeepBlending.

6.3 Comparisons

We performed comparisons with the following alternative methods (see Fig. 1 and Fig. 11): A textured mesh, generated by COLMAP [Schönberger and Frahm 2016; Schönberger et al. 2016] and textured with RealityCapture [Reality 2018], a per-pixel ULR method using the same mesh, Soft3D [Penner and Zhang 2017] and the DeepBlending [Hedman et al. 2018] method. For fairness, we retrained DeepBlending with our scenes to achieve the best possible results. We have included Soft3D for the scenes *Carpenter*, *Corner-Street* and *Vine*, while the other methods are provided for all scenes. We provide a comprehensive supplemental material page, containing paths from each scene with the different algorithms for easy comparisons. We also provide semantic maps, foreground/background layers and other intermediate data.

For the vast majority of cases, our method provides a much cleaner and plausible result compared to previous methods. The fact that windows are filled and that reflections move in a plausible manner are key elements of realism for navigation in these scenes. Soft3D performs well when we are close to the input cameras (see sequences in videos), but degrades rapidly as we move away

[1] Code for the main steps of the algorithm will be released, see https://repo-sam.inria.fr/fungraph/ibr-cars-semantic/.

from the input views. DeepBlending improves the overall result compared to ULR, but still cannot completely recover from the lack of window geometry, and cannot infer reflection flow.

7 LIMITATIONS AND CONCLUSIONS

Our method only produces *plausible* renderings of reflections and transmission for car windows, especially for the car interior.

7.1 Limitations

The inaccuracy of the interior geometry is a limiting factor for the quality of rendering we can achieve (please refer to supplemental for an example). The feature-based and dense ellipsoid estimations both require at least some reflection information; windows in scenes under very strong sunlight might not contain enough reflections for this step. In the specific configuration where there is a strong discontinuity in a highly transmissive area (typically dark car interior over a bright background with some reflections over the dark areas) our reflective stitching method is not very successful, resulting in rendering artifacts. This is due to the ambiguity of the min composite in this configuration. These limitations are shown in our supplemental material and video.

The robustness of some steps (geometry refinement, ellipsoid fitting) could be improved through learning-based approaches that could extract automatic features more resilient to variability in the input data. Our method also inherits limitations from the rendering algorithms used for the background and car bodies. Those parts could benefit from future work on geometric and material priors to render broader specular effects with high fidelity. Despite these shortcomings, compared to all previous methods our plausible rendering, and the reduction of the most visible artifacts is a major step forward to allowing useable IBR in a cityscape navigation context.

7.2 Summary and Conclusion

We have presented a new method that allows plausible rendering of cars, in a casual capture context using a single consumer camera. Our method is based on the introduction of an efficient reflection flow computation that can be computed in real time in a shader, using an analytical approximation of curved car window surfaces. We create a smooth car hull, filling the windows that are missing in the MVS reconstruction, efficiently enforcing spherical topology using image processing operations. The first approximation of the window surfaces is used to support the ellipsoid fit for the car windows, enabling the efficient reflection flow computation. The final component is the use of the reflection flow for a reflection layer synthesis algorithm, based on image stitching operations.

In conclusion, we have presented a first solution allowing plausible IBR of cars and in particular car window reflections. Our method makes a significant step forward in allowing applications requiring realistic free-viewpoint navigation in cityscapes to use IBR.

ACKNOWLEDGMENTS

The authors would like to thank S. P. Bangaru for initial tests, G. Brostow for fruitful discussions, J. Philip and S. Morgenthaler for helping with the supplemental material. This research was funded by the ERC Advanced Grant FUNGRAPH No 788065 (http://project.inria.fr/fungraph) and by the Rabin Ezra scholarship fund. The authors are grateful to Adobe for software donations and to Inria Sophia Antipolis - Mediterranée "Nef" computation cluster for providing resources and support.

A MESH REFINEMENT MINIMIZATION

For mesh refinement (Sec. 3.2), we use a conjugate gradient solver to minimize the following penalty function:

$$E(\mathbf{d}) = \sum_p \left(w_u(p) \Big(\mathbf{d}(p) - \mathbf{d}'(p) \Big)^2 + w_b \sum_{q \in N(p)} \Big(\mathbf{d}(p) - \mathbf{d}(q) \Big)^2 + w_l \Big(\mathcal{L}_d(p) - \alpha_l \Big)^2 \right) \quad (2)$$

where $\mathbf{d}(p)$ is the estimated depth at pixel p, $\mathbf{d}'(p)$ the initial depth at p (both expressed in $[0, 1]$), $N(p)$ is the set of pixels around p, $\mathcal{L}_d$ is the discrete Laplace operator on the depth. We set the Laplacian prior $\alpha_l = cos(3^\circ)$, and $w_b = w_l = 0.33$. We initialize the unary weight with $w_u(p) = max(0, \mathcal{P}_{car}(p) - \mathcal{P}_{win}(p))$, where the probabilities $\mathcal{P}$ for car and window respectively are extracted from the segmentation map. In subsequent iterations we set $w_u = 0$ for outlier pixels, defined as pixels that have moved from their initial 3D position more than 2% of the sphere radius (i.e. a few centimeters).

B SEMANTIC LABELING MRF

For the final semantic mask refinement, (end of Sec. 3.2), we solve a labeling problem with graph cut, using the constant data cost 0.5 for "car", and an adaptive data cost for "window". Pixels inside the window regions and outliers get cost 0.0 (i.e., likely windows), while pixels outside dilated windows or pixels with high photoconsistency are given cost 1.0. All remaining pixels are given cost 0.52 to encourage smaller window regions. The smoothness term is a color-gradient weighted Potts term, for neighboring pixels with different labels.

C ELLIPSOID FITTING ALGORITHM

In this appendix, we present the details of the ellipsoid fitting algorithm for windows (Sec. 4.2). For each window, we choose 10 reference images where the window is visible and as fronto-parallel as possible. We also ensure that each reference image has neighboring images on all 4 sides. We compare each reference image to its 10 closest neighbors. We compute ORB features [Rublee et al. 2011] for those images ; we use best buddy matching [Vaish et al. 2006] and Lowe thresholding [Lowe 2004] ($th = 0.95$) to establish correspondences. We also discard correspondences whose motion can be explained by the reconstructed window geometry, such as feature points on stickers or scratches on the windows. If a feature point in the reference image is reprojected in a neighbor view such that the distance to the matched point is smaller than 1.5 % of the image dimensions (10px for our typical 4Mpixel images), the correspondence is discarded. We fall back to dense image matching when the average number of successful matches is smaller than 0.1% of the window area (in pixels).

Feature point matching. We use a RANSAC inspired algorithm which computes the total number of inlier feature point correspondences as our score. That is, for each pair of radii (r_x, r_y), we count how many matches can be explained by the predicted reflection flow. A match between the reference and a neighbor image is an inlier if the flow predicts the location of the feature point in the neighbor view with no more than a 10px error for our typical 4Mpixel input images. We observed that most side windows are very anisotropic, with a much smaller curvature – nearly planar shape – around the vertical axis. We encode this with a prior on the ratio between r_y and r_x:

$$p(r_x, r_y) = \mathcal{N}_{\mu=8,\sigma=3.5}(\frac{r_y}{r_x}) \quad (3)$$

For windshields and rear windows, there is less anisotropic behavior and we use the constant prior $p(r_x, r_y) = 1$. We pick the pair of radii which maximises $s(r_x, r_y) = p(r_x, r_y)\#\text{inliers}(r_x, r_y)$.

Dense image matching. We only use the top ranked reference image to compute our score as this matching is slower. Images are also downscaled at 720p for speed and robustness to outliers. For each (r_x, r_y), we compute the reflection flows and warp 25 neighboring images into the reference view. The number of neighbors is increased as median consistency is sensitive to noise. We use the median-based photoconsistency from [Vaish et al. 2006] to compute an error map for each warped image. A per-pixel median error is extracted, and its mean value is computed over all pixels in the window mask. The parameter pair with the lowest error is selected.

REFERENCES

Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. 2004. Interactive digital photomontage. In *ACM Transactions on Graphics (TOG)*, Vol. 23. ACM, 294–302.

Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. 2001. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer Graphics and Interactive Techniques*. ACM, 425–432.

Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. 2013. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics (TOG)* 32, 3 (2013), 30.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 2017. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 4 (2017), 834–848.

Paul Debevec, Yizhou Yu, and George Borshukov. 1998. Efficient view-dependent image-based rendering with projective texture-mapping. In *Rendering Techniques' 98*. Springer, 105–116.

Pau Estalella, Ignacio Martin, George Drettakis, Dani Tost, Olivier Devillers, and Frédéric Cazals. 2005. Accurate interactive specular reflections on curved objects. In *Proceedings of Vision, Modeling and Visualization*. Eurographics Association. http://www-sop.inria.fr/reves/Basilic/2005/EMDTDC05

Clement Godard, Peter Hedman, Wenbin Li, and Gabriel J. Brostow. 2015. Multi-view reconstruction of highly specular surfaces in uncontrolled environments. In *International Conference on 3D Vision*. IEEE, 19–27.

Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M. Seitz. 2007. Multi-view stereo for community photo collections. In *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, 1–8.

Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. 1996. The lumigraph. In *Proceedings of the 23rd annual conference on Computer Graphics and Interactive Techniques*. ACM, 43–54.

Christian Hane, Nikolay Savinov, and Marc Pollefeys. 2014. Class specific 3d object shape priors using surface normals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 652–659.

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (TOG)* 37, 6 (November 2018). http://www-sop.inria.fr/reves/Basilic/2018/HPPFDB18

Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. 2016. Scalable inside-out image-based rendering. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 231.

Ivo Ihrke, Kiriakos N. Kutulakos, Hendrik P.A. Lensch, Marcus Magnor, and Wolfgang Heidrich. 2008. State of the art in transparent and specular object reconstruction. In *Eurographics State Of The Art Report (STAR)*. Eurographics Association.

Ivo Ihrke, Kiriakos N. Kutulakos, Hendrik P.A. Lensch, Marcus Magnor, and Wolfgang Heidrich. 2010. Transparent and specular object reconstruction. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 2400–2426.

Bastien Jacquet, Christian Hane, Kevin Koser, and Marc Pollefeys. 2013. Real-world normal map capture for nearly flat reflective surfaces. In *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, 713–720.

Johannes Kopf, Fabian Langguth, Daniel Scharstein, Richard Szeliski, and Michael Goesele. 2013. Image-based rendering in the gradient domain. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 199.

Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. 2003. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 277–286.

Anat Levin and Yair Weiss. 2007. User assisted separation of reflections from a single image using a sparsity prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29, 9 (2007), 1647–1654.

Marc Levoy and Pat Hanrahan. 1996. Light field rendering. In *Proceedings of the 23rd annual conference on Computer Graphics and Interactive Techniques*. ACM, 31–42.

David G. Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60, 2 (2004), 91–110.

Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. 2019. Local light field fusion: practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)* 34, 4 (2019).

Eyal Ofek and Ari Rappoport. 1998. Interactive reflections on curved objects. In *Proceedings of the 25th annual conference on Computer Graphics and Interactive Techniques*. ACM, 333–342.

Eric Penner and Li Zhang. 2017. Soft 3D reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 235.

Capturing Reality. 2018. RealityCapture reconstruction software. https://www.capturingreality.com/Product.

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 2011. ORB: an efficient alternative to SIFT or SURF. In *Proceedings of the IEEE International Conference on Computer Vision*, Vol. 11. IEEE, 2564–2571.

Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. 2016. Pixelwise view selection for unstructured multi-view stereo. In *Proceedings of the European Conference on Computer Vision*. Springer.

Sudipta N. Sinha, Johannes Kopf, Michael Goesele, Daniel Scharstein, and Richard Szeliski. 2012. Image-based rendering for scenes with reflections. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 100–1.

Richard Szeliski, Shai Avidan, and Padmanabhan Anandan. 2000. Layer extraction from multiple images containing reflections and transparency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1. IEEE, 246–253.

Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2019. Deferred neural rendering: image synthesis using neural textures. *ACM Transactions on Graphics (TOG)* (2019).

Justus Thies, Michael Zollhöfer, Christian Theobalt, Marc Stamminger, and Matthias Nießner. 2020. Image-guided neural object rendering. In *International Conference on Learning Representations*. ICLR. https://openreview.net/forum?id=Hyg9anEFPS

Vaibhav Vaish, Marc Levoy, Richard Szeliski, C. Lawrence Zitnick, and Sing Bing Kang. 2006. Reconstructing occluded surfaces using synthetic apertures: stereo, focus and robust measures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2. IEEE, 2331–2338.

Thomas Whelan, Michael Goesele, Steven J. Lovegrove, Julian Straub, Simon Green, Richard Szeliski, Steven Butterfield, Shobhit Verma, and Richard Newcombe. 2018. Reconstructing scenes with mirror and glass surfaces. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 102.

Turner Whitted. 1980. An improved illumination model for shaded display. *Commun. ACM* 23, 6 (June 1980), 343–349.

Patrick Wieschollek, Orazio Gallo, Jinwei Gu, and Jan Kautz. 2018. Separating reflection and transmission images in the wild. In *Proceedings of the European Conference on Computer Vision*. Springer, 89–104.

Bojian Wu, Yang Zhou, Yiming Qian, Minglun Cong, and Hui Huang. 2018. Full 3D reconstruction of transparent objects. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–11.

Tianfan Xue, Michael Rubinstein, Ce Liu, and William T. Freeman. 2015. A computational approach for obstruction-free photography. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 79.

Sid Yingze Bao, Manmohan Chandraker, Yuanqing Lin, and Silvio Savarese. 2013. Dense object reconstruction with semantic priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1264–1271.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. 2017. Scene parsing through ADE20K dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 633–641.

Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. 2019. Semantic understanding of scenes through the ADE20K dataset. *International Journal of Computer Vision* 127, 3 (2019), 302–321.