3D-aware Image Editing for Out of Bounds Photography

Amit Shesh[*] Antonio Criminisi[†] Carsten Rother[‡] Gavin Smyth[§]

Illinois State University Microsoft Research, Cambridge United Kingdom

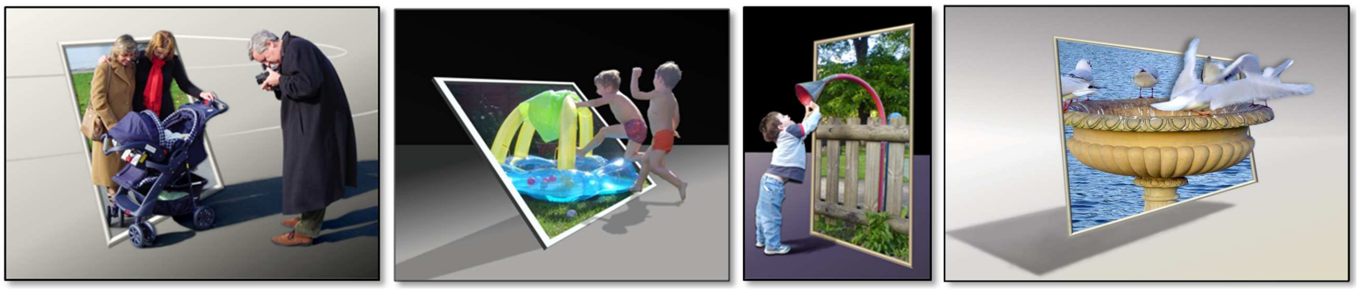


Figure 1: **Example "Out of Bounds images" obtained with our software tool.** The algorithms presented in this paper allow a user to easily create such depth-rich images starting from *single*, conventional two-dimensional photographs and paintings. The careful placement of occlusions and cast shadows is responsible for the strong "pop-out" effect in these images.

Abstract

In this paper, we propose algorithms to manipulate 2D images in a way that is consistent with the 3D geometry of the scene that they capture. We present these algorithms in the context of creating "Out of Bounds"(OOB) images - compelling, depth-rich images generated from single, conventional 2D photographs (fig. 1). Starting from a single image our tool enables rapid OOB prototyping; i.e. the ability to quickly create and experiment with many different variants of the OOB effect before deciding which one best expresses the users' artistic intentions. We achieve this with a flexible work-flow driven by an intuitive user interface.

The rich 3D perception of the final composition is achieved by exploiting two strong cues – occlusions and shadows. A realistic-looking 3D frame is interactively inserted in the scene between segmented foreground objects and the background to generate novel occlusions and enhance the scene's perception of depth. This perception is further enhanced by adding new, realistic cast shadows. The key contributions of this paper are: (i) new algorithms for inserting simple 3D objects like frames in 2D images requiring minimal camera calibration, and (ii) new techniques for the realistic synthesis of cast shadows, even for complex 3D objects. These algorithms, although presented for OOB photography, may be directly used in general image composition tasks.

With our tool, untrained users can turn ordinary photos into compelling OOB images in seconds. In contrast with existing workflows, at any time the artist can modify any aspect of the composition while avoiding time-consuming pixel painting operations. Such a tool has important commercial applications, and is much more suitable for OOB prototyping than existing image editors.

Index Terms: I.3.3 [Computer Graphics]: Picture/Image Generation— [I.4.9]: Image Processing and Computer Vision—Applications

[*]e-mail: ashesh@ilstu.edu
[†]e-mail:antcrim@microsoft.com
[‡]e-mail:carrot@microsoft.com
[§]e-mail:Gavin.Smyth@microsoft.com

1 Introduction

Many advanced image editing operations require the knowledge of the 3D geometry of the scene being visualized. For instance, inserting a 3D object into an image so that it looks consistent with the scene requires knowledge of the depth of all relevant surfaces. The positions of light sources are also needed to add shadows. For the task of re-lighting, i.e. moving a light source, the normals of all surfaces are also necessary. Unfortunately, estimating all such 3D information accurately from a single image is a severely under-constrained task and is far from being solved. This paper shows how minimal 3D information extracted from an image can be used to augment it with new objects, or enhance its depth perception.

The general algorithms described in this paper are presented through their application to "Out-of-Bounds" (OOB) photography. OOB photography is a recent technique which produces compelling, depth rich images from single, conventional 2D photographs (see exemplar results in fig. 1). By placing a 3D graphics frame in between different depth-layers in the image new occlusions are generated. This, in combination with the additional cast shadows, increases the sense of depth in conventional 2D images. The technical challenges in this application are: i) interactively inserting a 3D frame into an image in a way which conforms with the geometry of the scene, and ii) casting realistic shadows for any 3D object illuminated by an arbitrary light source. This paper describes algorithms to create such compositions quickly and effectively.

The general OOB concept is not new as a famous drawing *Drawing Hands* by Escher demonstrates[1]. The boundary of the paper creates T-junctions [17] which help us perceive the hands as coming out of the plane of the paper. This perception is strengthened by internal shading and by the shadows cast on the desk surface. These are the visual cues exploited in OOB.

Existing work-flows. Various techniques to create OOB images already exist. Fig. 2 illustrates a typical work-flow. Starting from a photograph, the user places a frame between the selected foreground layer and the background. By replacing the original background with a clean graphics one, distracting elements are removed and the foreground brought to life. Shadows cast by the frame and foreground object(s) further enhance the 3D illusion. Existing algorithms follow this work-flow *sequentially*, using general purpose editing tools such as Photoshop and Gimp (see an example tutorial at http://aczafra.com/2006/09/01/out-of-bounds-photography-using-photoshop/ and example images at http://www.flickr.com/groups/oob/).

[1]http://www.mcescher.com : Picture gallery "Back in Holland 1941-1954"

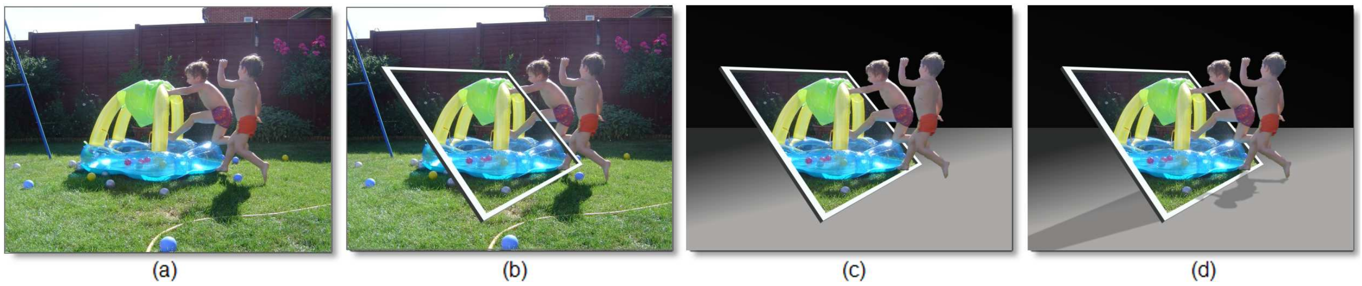


Figure 2: **A Typical Work Flow.** **(a)** The original photograph. **(b)** A frame is positioned in the image consistent with the scene's geometry. **(c)** Objects of interest are segmented out and placed against a clean graphics background. **(d)** Shadows are cast from the inserted frame and foreground objects to enhance the scene's 3D geometry and produce the final OOB composition.

Such a process, while effective has two critical drawbacks. First, it relies on the artistic ability of the user to draw perspectively correct frames and realistic-looking shadows. For example, drawing the frame in fig. 3(left) so that it appears to lie on the ground is not an easy task for an amateur user. Secondly, the OOB creation process is inherently experimental in nature. Creating the perfect composition in a single attempt is extremely difficult even for the most seasoned artist, and hence the ability to modify components (*e.g.* different frames, shadows in different positions) and rapidly explore the vast space of possible OOB results is highly desirable (see fig. 3). However, editing OOB compositions using general purpose editing tools involves slow and painstaking pixel-painting and often forces the user to redo many expensive operations from scratch.

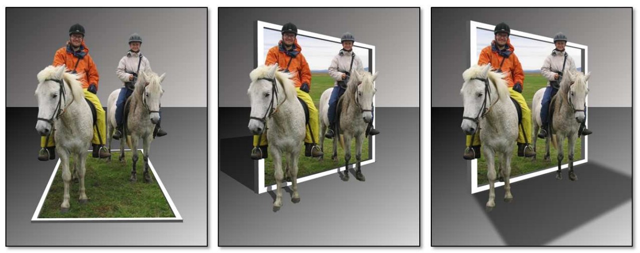


Figure 3: **Various OOB effects** created from the same input photograph (see fig. 13). Being able to interactively modify frame and cast shadows enables quickly exploring all possible OOB compositions.

Our approach. We propose a new, *non-sequential* (non-modal) approach for creating OOB images. Our tool permits the user to modify any aspect of the composition at any time while automatically maintaining geometric consistency. For instance when repositioning the frame, both the frame's shadow and the object shadows cast on the frame are automatically updated, thus avoiding further manual intervention. We achieve this with our new geometry-driven frame positioning and shadow casting techniques.

After an overview of related work, we enunciate the problem and the focus of this paper. Algorithmic details are presented in Sections 2.1-2.5, followed by comparisons with existing methodologies in Section 2.6. Results are shown throughout the paper and in the accompanying video.

Related Work. A vast body of literature in computer vision and graphics have dealt with the problem of reconstructing 3D geometry from single or multiple images [6, 7, 9–12, 15, 19, 20] and so have many commercial products like Boujou (http://www.2d3.com), PhotoModeler (http://www.photomodeler.com), and Canoma (http://www.canoma.com). However, the problem we face in this paper is new: we wish to edit a given image in a way that obeys the scene's underlying geometry without having to acquire or use any 3D information about the scene itself. Specifically, we aim to interactively extract the minimum 3D information needed to manipulate inserted objects and cast convincing shadows. We achieve this by borrowing from existing computer vision techniques and developing new ones.

Description	*Votes*		
	Like	*OK*	*Dislike*
2D (flat) frame, no shadows	27%	53%	20%
3D frame, with shadows	**66%**	21%	13%

Table 1: **Results of our survey.** "Like", "OK" and "Dislike" votes accumulated over 539 participants and 5 different image sets. A majority expressed a marked preference for the most complete OOB composition; i.e. with 3D frames and cast shadows for both the frame and objects (2nd row).

The work on spidery mesh [11] and automatic photo pop up [10] are related since they produce depth perception by enabling limited view point change without explicitly reconstructing the full 3D geometry. Also, Chuang *et al.* [5] invoke 3D perception by using apparent movement in images. In contrast, our technique enhances 3D perception within single, static images, rendered with conventional display or printing technology.

Problem Statement and Goals. OOB photography comprises the synergistic effect of 3D perception from a variety of individual effects: depth discontinuity, flat, 2D/3D and planar/curved frames, strategically placed shadows, etc. We conducted a preliminary survey to ascertain the relative importance of some of these effects in an OOB image and to investigate how an OOB tool could be used for commercial applications. 539 individuals participated in this web-based survey over two weeks. They were asked i) whether they liked the OOB effect and whether they would be interested in creating such images themselves, ii) to compare and vote for the best among two OOB variants and iii) to brainstorm the use of such images. Participants also voluntarily submitted verbose comments at every stage. Table 1 summarizes the result.

Participants clearly preferred the OOB variant with both 3D frames and cast shadows for both frame and foreground objects[2]. Their comments also revealed an overall interest in the OOB framework[3], although a few were not very impressed[4]. Many interesting applications were suggested: advertising campaigns, card making, web design, business presentations, etc. Thus, in this work we focus on facilitating the two effects that were voted to contribute considerably to the OOB effect: geometrically consistent 3D frames and cast shadows.

2 From Photographs to OOB Images

This section outlines the user interface for our OOB tool as well as technical details of the general object insertion and shadow synthesis algorithms.

[2]"I like the third [image] because it doesn't seem like it's cut and paste, the shadow makes it realistic"

[3]"Being able to turn some of the pictures into 3D works of art like these would be great. I can only imagine the amount of time it would take in a photo editor to perform this manually"

[4]"I don't like very much edited photos with real humans"

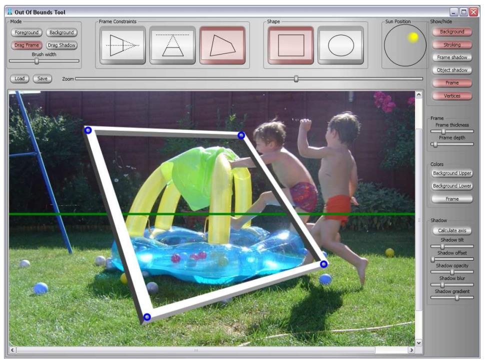


Figure 4: **A snapshot of our tool.** Creating OOB images involves quick and intuitive 2D manipulations. See also fig. 2.

2.1 The tool's user interface

A snapshot of our OOB software tool is shown in fig. 4.

Frame placement. The user places a 3D frame in the image by simply dragging its four corners; its apparent thickness and depth are automatically determined to make it look consistent with the 3D geometry of the existing scene. The actual color, thickness, depth and shape of the frame can be modified at any time. This topic is discussed further in Section 2.2.

Foreground selection. Foreground objects are extracted in the form of an alpha matte by a brush-based interface using existing techniques. See Section 2.3 for details.

Shadow generation and manipulation. Cast shadows are generated for the frame and the selected foreground objects. An intuitive user interface allows the user to select the position of the illuminant (*e.g.* the sun), the shadow sharpness, etc. Shadows may be directly dragged with the mouse. Section 2.5 discusses the details.

Background editing. The original background is replaced by a new image, gradient effects or some transformation of the original image itself. Details are discussed in Section 2.4.

Non-sequential interaction. An important aspect of our work is that the user is free to edit *any* aspect of the composition in *any* order. Corresponding geometric entities are automatically updated.

2.2 Computer assisted frame placement

The inserted frame provides important 3D cues as it produces new occlusions between itself and the foreground objects, and makes them stand out. Our survey showed that additional frame depth, i.e. a 3D instead of 2D frame, was one of the factors that improved the 3D feel of the final composition. Thus, correct perspective rendering of all sides of the frame is critical to produce a convincing illusion. We assume the frame to be a convex quadrilateral. More general shapes can be handled by parameterizing them as inscribed in such a quadrilateral (currently the tool also allows elliptical frames).

Our goal is to design a simple, interactive interface to place a 3D frame within an existing image, consistent with the scene geometry. One option could be to provide a conventional 3D track-ball interface. However this method offers too many unnecessary degrees of freedom (6 DOFs for 3D pose and also camera parameters). Also, 3D interface metaphors do not blend well with the otherwise 2D operations in OOB creation. Alternatively, a simple 2D polygon-drawing interface could be adopted in which the user may freely move all vertices of the polygon (16 for a 3D frame). This interface places the burden of achieving correct perspective on the user.

We propose an easy-to-use 2D interface which exposes only the minimum, necessary parameters to generate convincing 3D frames. The user only needs to position the four corners of the frame's front face and adapt the frame's thickness and depth. The positions of the remaining 12 corners in the image are automatically computed via the "calibrated" geometric approach described next. Fig. 5 illustrates an example and motivates the importance of our approach.

2.2.1 Camera calibration for frame rendering

Camera calibration is a common task in computer vision. Its goal is to compute the camera *intrinsic* and *extrinsic* parameters. A conventional approach exploits cues in the image like parallelism and orthogonality, e.g. of a building, to perform this task [9]. Since such cues might not be present in a generic image, the user has to provide them. In our work we only use the drawn frame's front face (*i.e.* only four corners) to estimate all camera parameters. Unfortunately, the camera has more DOFs than the constraints provided by the imaged frame. Therefore, reasonable prior values must be assigned to a number of geometric unknowns such as the pixel's aspect ratio and skew. Our contribution is a closed-form, real-time solution for the remaining unknown camera parameters.

Assuming a conventional pinhole camera model, a world 3D point $\mathbf{X}$ is projected into the corresponding image point $\mathbf{x}$ as

$$\mathbf{x} \sim K[R|\mathbf{t}]\mathbf{X} \qquad (1)$$

where $\sim$ indicates equality up to scale. The points $\mathbf{x}$ and $\mathbf{X}$ are represented by 3- and 4-vectors, respectively, in homogeneous coordinates. The 3×3 matrix R represents the camera rotation (3 DOFs). The 3-vector $\mathbf{t}$ represents the camera center, and the 3×3 matrix K represents the camera intrinsics.

Intrinsics. We start by estimating the intrinsics matrix K. A commonly used model for K is:

$$K = \begin{pmatrix} f_x & \sigma & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{pmatrix}. \qquad (2)$$

The principal point $(p_x, p_y)^\top$ is chosen to be the center of the image by default. Also, the camera CCD pixels are safely assumed to be square, *i.e.* $f_x = f_y = f$ with skew $\sigma = 0$. This leaves us with the focal length f as the only remaining unknown. The assumption that the four corners of the frame's front face form a rectangle provides a quadratic constraint on f from which a closed form solution is computed (Details in appendix A[5]).

Extrinsics. To determine the extrinsic parameters (rotation and translation) we assume that the frame: i) lies on the $Z = 0$ plane in the world coordinate system, and ii) forms a parallelogram with unknown height h, length l and skew s. Ideally we would like to enforce its skew $s = 0$ (*i.e.* the frame being a proper rectangle instead of a parallelogram); however, that would negate guarantees of a solution to the problem in all cases (selecting s as one additional unknown is the best choice to guarantee a closed-form, consistent solution in 3D and yet give freedom to the user to drag the 4 points independently). Formally, the four frame corner points are:

$$\mathbf{X}_1 = (0,0,0,1)^\top, \quad \mathbf{X}_2 = (l,0,0,1)^\top,$$
$$\mathbf{X}_3 = (l+s,h,0,1)^\top, \quad \mathbf{X}_4 = (s,h,0,1)^\top$$

Consequently, the four frame sides are $(\mathbf{X}_1, \mathbf{X}_2)$, $(\mathbf{X}_2, \mathbf{X}_3)$, $(\mathbf{X}_3, \mathbf{X}_4)$, $(\mathbf{X}_4, \mathbf{X}_1)$. This provides a total of 9 unknowns: l, s, h, R (3 DOFs) and $\mathbf{t}$ (3 DOFs). Let $\mathbf{x}_{1\ldots4}$ be the corresponding four image points dragged by the user. The 8 linearly independent equations provided by eqn. (1) suffice to solve for the 9 unknowns, since the

[5]Supplementary material can be found at http://research.microsoft.com/en-us/projects/i3l/i3l_oob.aspx

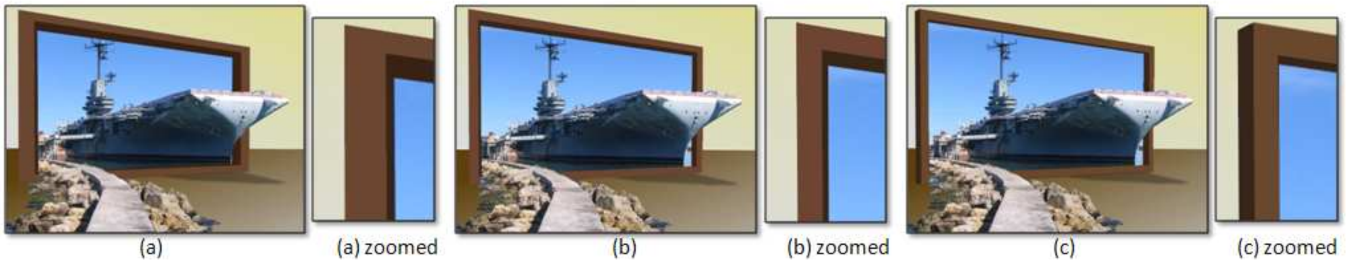


Figure 5: **Effect of camera calibration on the geometry of the inserted 3D frame.** **(a)** The frame rendered with incorrect camera parameters. The borders do not have uniform width and the internal sides do not look right. **(b)** The frame using our calibration technique and a *fixed* focal length of the camera, corresponding to a standard $50mm$ lens ($f = 750$ pixels). The borders' widths are correct but the internal sides still look unrealistic. **(c)** Our final result. Correct estimation of the camera intrinsic parameters yields a correct looking 3D frame. See Section 2.2.

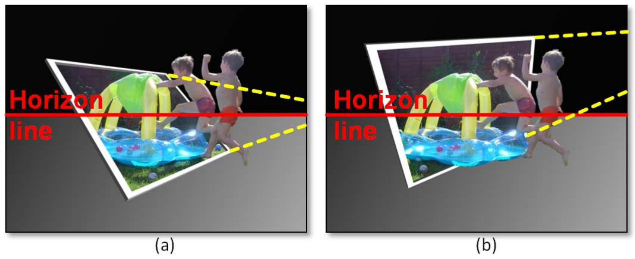


Figure 6: **Constrained frame manipulation.** **(a)** The frame is constrained to have two sides intersect at a point on the horizon line. This ensures geometric consistency as it implies that in the world these two sides of the 3D frame are parallel to the ground. **(b)** Allowing unconstrained placement may lead to inconsistent-looking frames.

overall scale can be set arbitrarily (*e.g.* by imposing the extra constraint $||\mathbf{t}||^2 = 1$). Appendix B[6] shows how a closed form solution for all camera parameters is obtained for any given $\mathbf{x}_{1\ldots4}$. This is true even for extreme cases like when two frame sides intersect each other in the image; in which case, one or more of the 3D corners lie behind the camera. Finally, to add realism the frame faces are shaded based on the surface normals.

Note that our method of computing camera parameters from the drawn frame is not guaranteed to acquire the parameters of the actual camera which captured the picture. This is largely because the user places a frame in the image without any knowledge of actual camera parameters, solely so that it looks consistent *visually*. This possible discrepancy is not a problem, and may even be exploited to our advantage. For instance, the selected frame may yield a considerably shorter focal length than the true one and hence may be used to deliberately enhance perspective distortion to strengthen the "pop-out" effect.

2.2.2 Constraining the frame geometry

So far we have allowed the user to position the frame freely in the image. However, there are many scenarios where strong geometric cues (*e.g.* the ground plane or vertical structures) are visible in the image. In such cases the inserted frame should look "consistent" with the scene geometry.

Our tool implements the most common scenario, *i.e.* that of a visible ground plane. As shown in fig. 6 if the vanishing line of the ground plane (the horizon line) is known (automatically computed or manually entered) then the frame can be constrained to have two of its sides intersect at a point on the vanishing line (fig. 6(a)). This corresponds to imposing that two sides of the frame are parallel to the ground and helps ensure scene consistency. We can also force the bottom side of the frame to lie on the ground (as in fig. 2(d)). To achieve this we compute the normal to the ground plane as $n = K^T \mathbf{l}$, given the horizon line $\mathbf{l}$ and the camera intrinsics K [9].

[6]Supplementary material can be found at http://research.microsoft.com/en-us/projects/i3l/i3l_oob.aspx

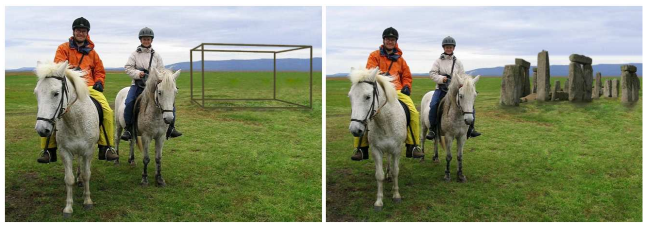


Figure 7: **Camera calibration for 3D object insertion.** Once the camera and the ground plane have been calibrated, additional 3D objects can be inserted into an existing image. (Left) a wire-frame box was added in a perspectively correct manner to the picture in fig. 13. (Right) As a proof of concept, we also inserted Stonehenge into this picture. Light position and color were adjusted manually.

2.2.3 Camera calibration for 3D object insertion

The approximate camera calibration allows us to insert generic 3D objects into the scene easily. Specifically our calibration procedure provides us with all the terms in eqn. 1, as well as the 3D position of the ground plane. With this, it is possible to project *any* 3D point in the world coordinate system into the image. Fig. 7 shows an example.

2.3 Foreground extraction

GrabCut [21] is used to interactively select the foreground regions and extract their alpha matte. Foreground objects are those we wish to have "popping-out". Note that the user only needs to extract accurate boundaries in those areas outside the frame; other boundaries can be coarse. This reduces the amount of manual interaction and increases speed of execution. The ability to iterate back and forth between the different algorithmic steps means that only minimal segmentation effort is required to achieve the desired result. For high resolution images, we use 2-scale multi-resolution energy minimization [16] for efficiency.

2.4 Background editing

Strong 3D effects are achieved by replacing the original background with a simple, uncluttered one. This focuses the viewers' attention onto the foreground object and removes distracting background elements, as dictated by basic photographic principles. This strategy is supported by the many OOB examples on Flickr. Alternatives include modifying the original background (*e.g.* frosted glass effect) or replacing it with an entirely new image (example in fig 14).

2.5 Shadow generation and editing

Most subjects in our survey considerably preferred OOB images with cast shadows. In fact, shadows provide important visual cues

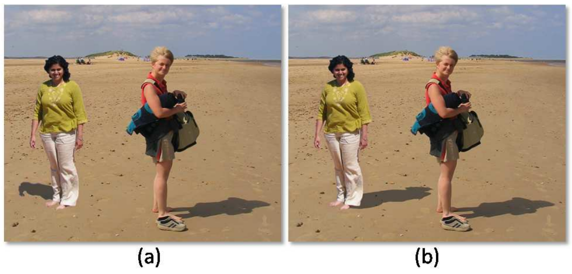


Figure 8: **Consistency of directional cast shadows.** (a) A composite image where the left person (and her shadow) has been added to an existing image. It is visually disturbing as the two shadows are inconsistent. This could be a potential result of [14] assuming a perfectly extracted and color-corrected shadow. (b) A geometrically consistent composition created with our tool. Consistent shadows improve the realism of the final image. Incorrect internal shading and attached shadows are less noticeable as shown in [3].

that "anchor" the object onto the ground plane and enhance a composition's overall sense of realism.

There has been some debate about how realistic synthesized shadows have to be in order to deceive the human eye. For instance, Cavanaugh [3] observed that incorrect *attached* shadows and *shading* may sometimes go unnoticed. However OOB images often contain strong *cast* shadows of nearby objects against a clean graphics background, making inconsistencies more prominent and the composition unrealistic (see fig. 8). Therefore we believe that a principled approach to generating realistic shadows is essential for a convincing illusion. We propose simple algorithms to generate and manipulate cast shadows effectively. These algorithms transcend OOB photography; they can be used directly in generic image composition. Fig. 8 shows an example.

If strong cast shadows for the foreground object already exist in the original image, we can simply "import" them by means of existing brush-based techniques [22] (*e.g.* fig. 1 leftmost image). However, shadow matting is often difficult (*e.g.* the original ground plane is highly textured or bumpy - see original images in fig. 13), or even impossible (*e.g.* the shadow is very faint or not visible at all). Also, shadow matting techniques are unsuitable if shadows have to be projected onto the inserted frame. Thus, synthesizing novel, realistic-looking shadows is desirable.

To this purpose one could think of first reconstructing the complete 3D scene and then using ray-tracing to generate the shadow of interest. However, despite recent advances, existing techniques for 3D reconstruction from *single* images [6, 10, 12, 20] are still complex and thus not suitable for our purposes. Moreover, they are often an "overkill". In fact, as shown later, knowing the complete scene geometry is often unnecessary. Since the end product is another 2D image we keep all operations in 2D by means of planar projective transformations. Note that as we do not need explicit 3D reconstruction we can also bypass the task of accurately calibrating the camera which captured the original picture. Instead, we use the approximate calibration described in Section 2.2.1 to initialize the unknown degrees of freedom of the necessary projective transformations. Next we describe the details of our algorithm.

2.5.1 Casting the frame's shadow

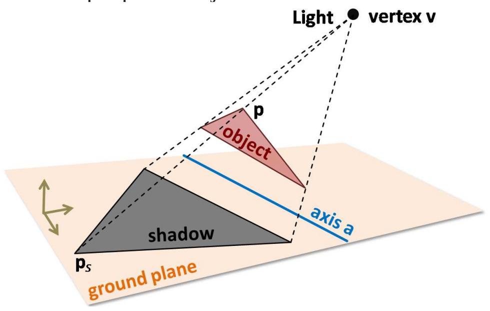
By construction the frame's 3D position with respect to the ground is known (Section 2.2). Given the position of the light source, the shadow cast by the frame onto the ground is promptly computed by ray-casting (this can also be formulated as a projective matrix transformation [1]). The frame's internal self-shadows are computed in the same way.

The position of the frame shadow is easily manipulated via a conventional click-and-drag interface. This corresponds to changing the position of the illuminant, assumed to lie on a hemisphere with infinite radius.

2.5.2 Casting the object's shadow

In contrast to the frame, the 3D coordinates of the segmented foreground object(s) are not known. Hence a more elaborate approach is necessary.

The following schematic figure illustrates the geometry of shadow casting for a simple planar object.



In the image plane, the mapping between points on the object and the corresponding shadow points can be compactly described by a *planar homology* [6]. More formally, given the 2D image of an object point $\mathbf{p}$ (in homogeneous coordinates), the corresponding shadow point $\mathbf{p}_s$ is given by $\mathbf{p}_s = H\mathbf{p}$, with H a 3×3 matrix representing the homology transformation. A homology matrix H has 5 DOFs and can be parameterized by a vertex $\mathbf{v}$, an axis $\mathbf{a}$ and a characteristic cross-ratio μ as:

$$H = I + \mu \frac{\mathbf{v}\mathbf{a}^\top}{\mathbf{v}^\top \cdot \mathbf{a}} \tag{3}$$

where I is the identity matrix. $\mathbf{v}$ is the image of the light source and $\mathbf{a}$ is the image of the line of intersection of between the object's plane and the ground plane. The scalar parameter μ encapsulates all remaining 3D DOFs such as camera intrinsics, distance of the light source, etc. In fact, there are many camera/object geometric configurations which all produce the same 2D image. A homology represents the *minimal* model that provides the user with sufficient control to generate and manipulate the shadow of *any* planar object.

In theory we can use the above approach to cast shadows of objects approximated as triangle meshes, where each triangle has its own homology. However, having 5 DOFs per triangle would overwhelm the user. Therefore, we attempt to model each object with as few planes as possible (an approximating plane may pass through the middle of the 3D object and not correspond to any existing surface.). More precisely, we split the extracted 2D object mask into a series of vertical strips in the image, each with its own homology, as shown in fig. 9. The bottom sides of these strips are their intersections with the ground, and thus define the series of axes of the respective homology matrices. In practice, manipulating these axes, together with the 2D light position is a sufficiently flexible, yet simple interface.

This approach works well for a wide variety of complex objects (see fig. 9). The exceptions are objects where i) the depth variation within a single strip is very large and ii) such depth variation shows up in the cast shadow. Fig. 9(c,d) provides an example in which the front legs of the spider are poorly approximated by a single plane. Using multiple planes suffice to generate realistic shadows for such objects (fig. 9(e,f)). (This is related to the observation in [4] that for some objects, morphological details of their shadows are less important.) More examples with a single axis are shown in fig. 1(2 middle images), fig. 3, and fig. 14(dinosaur and playground).

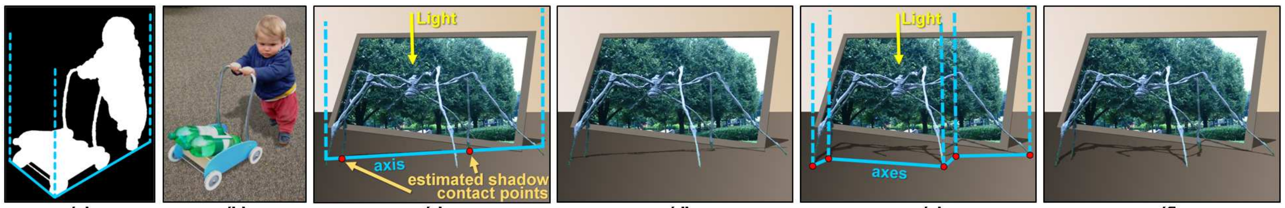


Figure 9: **Casting shadows via homologies.** To cast shadows, an object is approximated by a series of planes. Such planes are vertical strips which subdivide the object mask. Results are shown for two planes (a,b), a single plane (c,d) and four planes (e,f), where the respective left image is for illustration purpose only. The single axis in (c,d) was computed automatically. However, it approximates the underlying 3D geometry poorly (note the lack of shadow for the spider's front leg). Using more planes (e,f) leads to a better 3D proxy and thus more accurate results, at a cost of more complex interaction. Since the shadow contact points on the ground (tips of legs) are most important to be correctly detected, we advise the user to place an axis through them.

User interaction. The following series of operations take place from the user's point of view: i) an object is segmented; ii) by default a the cast shadow is computed automatically by using a single plane whose axis and μ parameter are computed automatically. This step requires no user input; iii) if the shadow looks unsatisfactory, the user can move the position of the axis or add more axes. The user can also move the light source, change μ or locally edit the shadow with a brush interface. To our knowledge *only* the single plane approach with manually set parameters has been described before [2]. Next, we describe our approach for automatically predicting the homology's axis.

Estimating the homology axis. Given an unknown 3D object we wish to automatically estimate the optimal, single approximating plane; *i.e.* the plane which cuts through the object and minimizes the depth error. This is a challenging task. However we observe that a 3D plane which is parallel to the image plane is often a good choice, yielding reasonable shadows. A more important observation is the following: consider all 3D points of the object's surface which touch the ground. We call them *ground contact points*. A subset of these points are *shadow contact points*, where the light source transitions from being "visible" to "invisible" (the light source is visible from a 3D point if the 3D segment connecting the light and the point does not intersect the object, see fig. 10). The shadow must "start" at these points to have the object appear "anchored" to the ground. This is the key difference between our solution and the naive one in fig. 11. The importance of these shadow contact points is related to the study in [13] which showed (to some extent) how the presence of such contact points provides approximate information about the object's height. We detect the most likely shadow contact points and fit an axis through them. The light is assumed in a default position.

We detect ground contact points by: i) projecting the object mask onto the ground, using our "roughly" calibrated geometry; ii) orthographically projecting the light and the camera positions onto the ground (see fig. 10), and iii) assigning to each point p on the silhouette S of the projected object mask a belief g for it being a ground contact point, as follows:

$$g(p) = e^{-\left(\frac{h(p)-1}{0.2}\right)^2} \quad \text{with } h(p) = (p_y - y_{min}) \,/\, (y_{max} - y_{min})$$

where y_{min}, y_{max} are the vertical coordinates of the bounding box of S. The formula for g is based on the observation that as the viewing direction of the camera is typically not from the top, pixels towards the bottom are more likely on the ground (potentially the *object-class-specific* training approach of Lalonde *et al.* [14] could improve this step further). Next, we estimate $v(p)$ which is the belief for a point $p \in S$ being visible from the light source, assuming it to be on the ground plane. We use $v(p) = 1 - g(q)$, where q is the first intersection point with S of the ray joining the light source and p (and $v(p) = 1$ if $p = q$). Finally, we compute a function $t(p)$ that has a high value (1.0) if the light ray at point p is tangential to S, and a low value (*e.g.* 0.1) otherwise. (We first smooth the object mask

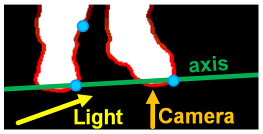


Figure 10: **Predicting the homology axis.** Explanation of the results presented in fig. 11(c). The object mask projected onto the ground plane is visualized in white. The red curve visualizes the quantity $g(p)$ (brighter red indicates points which are more likely to be ground contact points). The cyan dots indicate points where the light direction is tangent to the silhouette (where $t(p) = 1$); i.e. potential shadow contact points. Our estimated axis (green line) passes through the true shadow contact points (both $v(p) = 1$). A third tangent point (above axis) is correctly discarded by our method, due to a low visibility cost ($v(p) = 0.17$ in that case).

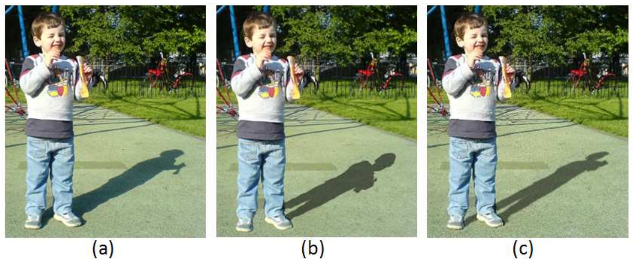


Figure 11: **Comparison of shadow synthesis algorithms.** (a) The original image with a cast shadow. (b) Naive shadow obtained by simply scaling and rotating the object mask. The shadow looks incorrect, especially near the feet. (c) Our perspectively correct shadow generated using an automatically estimated axis and scale factor μ.

with a Gaussian ($\sigma = 3$) to condition the tangent point detection algorithm.) We now determine the pair of points p_i ($i \in 1,2$) that maximizes $\prod_i g(p_i)v(p_i)t(p_i)$ (see fig. 10). The axis we seek is the line passing through this pair. For stability we constrain the point pair to have a minimum distance (50% of the maximum distance of any two tangent points) and the axis to have a minimum angle (10^o) with respect to the camera direction (in case no pair satisfies these constraints, an axis parallel to the image plane and through the optimal point p is chosen).
The algorithm described above has been found to work well on many examples of up-right objects (*e.g.* fig. 11, fig. 2, and dinosaur in fig. 14). Our algorithm could be extended to predicting multiple axes; however we found it better to leave this choice to the user.

Initialising μ. In order to complete the homology estimation we also need to compute the scalar μ. In the single plane (axis) case μ is initialized by assuming the plane to be perpendicular to the ground. Exploiting our initial 3D set-up, we first project the axis onto the ground plane. Then a new 3D point is computed by moving an arbitrary 3D point lying on the axis perpendicular to the ground. This new point and its shadow gives two linear constraints on H,

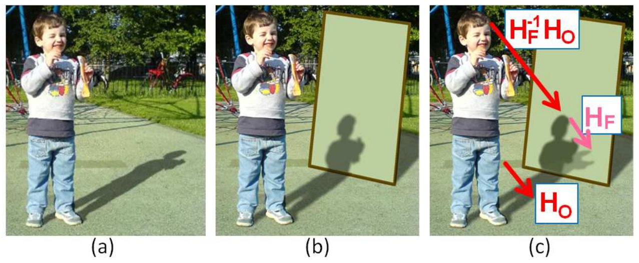


Figure 12: **Shadow transfer** from the ground plane (a) onto an artificially added plane (b). (c) Let H_O and H_F be the homologies of the shadows of two objects O and F onto the ground respectively. Then the homology for the shadow of O onto F is given by $H_F^{-1}H_O$.

sufficient to derive μ. If the result is unsatisfactory, the user can change the value of μ interactively.

Multiple planes. Although the axes for multiple planes are interactively selected, the corresponding μ's are computed automatically. The value of μ for the first plane is given as described earlier. Now, let $\mathbf{p}$ be the image of a 3D point which lies at the intersection of the first plane and its neighboring plane. As above, $\mathbf{p}$ and its shadow provide two linear constraints which suffice to obtain the value of μ of the neighboring plane. By repeating this operation for pairs of neighboring planes all values of μ are derived.

2.5.3 Shadow manipulation and transfer

As with frame shadows, object shadows can be moved simply by a "click-and-drag" interface. If the light is dragged to lie on or near the ground plane, the cast shadow degenerates to a line. While this is artistically undesirable, it can easily be avoided. Furthermore, the appearance of the shadow may be edited using a 2-brush interface (pen/erase). Typically, this step is used to remove minor inaccuracies in the shadow mask. Such edits are projected back onto (a copy) of the object mask (pre-homology warping). This allows us to move the shadow consistently after its mask has been manually edited. Such an operation would be very difficult to achieve with an explicit 3D reconstruction since the user may provide conflicting edits after moving the shadow. Our homology-based approach also allows us to cast the object shadow onto the frame (see fig. 12 for an example).

Other parameters like darkness, opacity and gradient of the shadow can be interactively controlled to improve its appearance. These DOFs correspond to changing the area of the light source, strength of ambient (*e.g.* reflection from the sky) and indirect lighting. Similar to [14], the shadow gradient is computed as a function of the distance from the object mask, modulated by the belief of an object mask point to be a ground contact point. Similar parameters and effects have been used to alter existing shadows in images [18] or to produce customized non-photorealistic shadows [8].

2.6 Comparison with Existing Approaches

Our techniques can be seen as building upon existing features of conventional photo editors, and adding new features to focus on rapid OOB prototyping. Specifically we impose critical dependencies between image layers to make them geometrically and mutually consistent, and facilitate 3D-aware operations like shadow creation and movement which greatly reduce the time required to create and switch between OOB variants. Without these dependencies, a user would have to resort to independently manipulating each layer (shadows, frames) to move from one OOB variant to another. Not only is this redundant, but also expensive because moving a frame in existing tools involves redrawing it with the correct perspective, and moving a shadow involves actually repainting it from scratch. Also, the burden of making them look realistic rests on the artistic and pixel-painting abilities of the user.

Comparison for changing one OOB variant into another				
Change from	Change to	Actual task	Gimp (sec)	Our tool (sec)
fig. 3(middle)	fig. 3(right)	**Move shadow**	105	5
fig. 3(middle)	fig. 3(left)	**Move frame**	115	11

Comparison for creating OOB images from scratch		
Individual task	Gimp (sec)	Our tool (sec)
Frame editing	66	11
Shadow editing	112	15
Foreground selection	136	22
Remaining tasks	37	7
Total time	352	55

Table 2: **Comparing speed with existing tools.**

To demonstrate this, we asked an experienced user to create new OOB images and change one OOB variant to another using our tool and the popular photo editor Gimp. Table 2 shows the results of this rudimentary experiment. The speed-ups in the upper table are mainly because the shadow had to be repainted from scratch and the frame had to be redrawn with the correct perspective. The lower table shows timings averaged over creating fig. 1(third), fig. 3(middle) and fig. 15(left). It shows how the two OOB tasks that we focus on can be completed significantly faster in our tool than in Gimp. It should be noted that the time for frame manipulation in Gimp is without any "trial-and-error", and the faster timings for foreground selection in our tool are because of Grabcut [21] which is not a contribution of this paper.

Thus we believe many ordinary users who would not otherwise find the OOB creation process worthwhile due to the above limitations of current work-flows and tools would now be willing to devote the minimal effort required by our tool to create them.

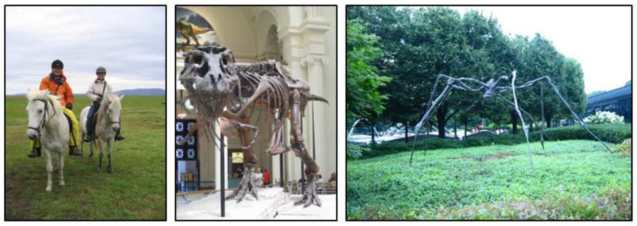


Figure 13: **Some original images** used to create OOB images in fig. 3, 9, and 14. Note that the objects' shadows are often not strong, or cast on non-flat ground.

3 Further Results, Limitations and Future Work

Resulting OOB pictures of different kinds are shown in figs. 1, 3, 5, 9, 14, 15 and in the accompanying video. In contrast, the original photographs (examples in fig. 13) look rather flat. The time to create these results varied between 15 seconds and about a minute.

A current limitation of our tool is its ability to handle only planar frames. An example with a non-planar frame is shown in fig. 14(rightmost), which is created using a conventional photo editor. Adding this feature would only involve implementing a more complete curve drawing interface.

An important outstanding question is: "For which pictures does the OOB effect work?" In general, landscapes and pictures with no prominent foreground objects are bad OOB candidates. Fig. 16 illustrates such an example. Firstly, the oranges do not form an interesting, prominent foreground. Secondly, the purpose of the frame is to emphasize existing large depth discontinuities (foreground-to-

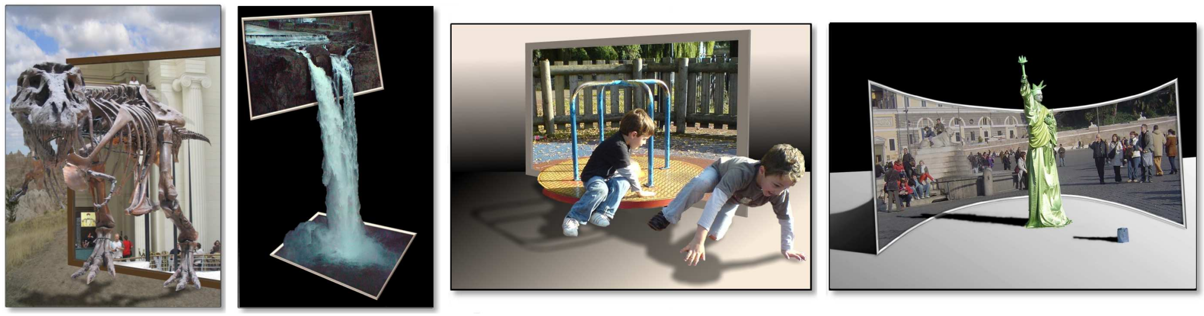


Figure 14: **Out of Bounds pictures** from photographs of very different nature. The rightmost image with a curved frame was not created with our existing tool, but a conventional photo editor, and motivates future development.

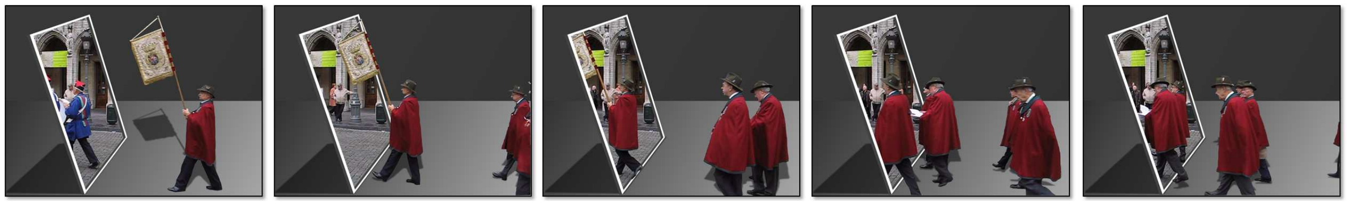


Figure 15: **Transforming a video sequence into an Out of Bounds slide show.** (please see accompanying video)

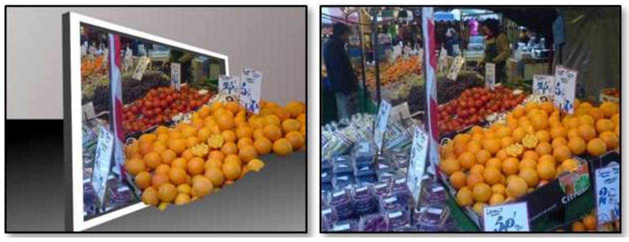


Figure 16: **Limitations.** An attempted OOB picture (left) from an image with no prominent foreground object (right).

background), which are not present in this case (the frame passes straight through the oranges' box in 3D).

These insights pose interesting research questions: "How can we automatically detect good OOB candidate pictures?" Ideally, the OOB tool should be able to browse through users' photo collections and select OOB candidate images automatically. Another new line of research could be to simplify a video into a static OOB scene, or to create a new OOB video with animated frames. Handling video is in many ways simpler since depth cues, such as occlusion, can be extracted more easily due to frame coherence, thus possibly enabling fully automatic frame placement.

4 Acknowledgements

We are grateful to Rick Szeliski for his useful inputs and John Winn for lending his voice in the accompanying video. We also thank the anonymous reviewers for their helpful comments and critique.

References

[1] J. Blinn. Me and my (fake) shadow. *IEEE Comput. Graph. Appl.*, 8(1):82–86, 1988.

[2] X. Cao, Y. Shen, M. Shah, and H. Foroosh. Single view compositing with shadows. *The Visual Computer*, 21(8-10):639–648, 2005.

[3] P. Cavanaugh. The artist as neuroscientist. *Nature*, 434:301–307, 2005.

[4] P. Cavanaugh and Y. Leclerc. Shape from shadows. *J. Experimental Psychology*, 15:3–27, 1989.

[5] Y.-Y. Chuang, D. B. Goldman, K. C. Zheng, B. Curless, D. H. Salesin, and R. Szeliski. Animation pictures with stochastic motion textures. In *SIGGRAPH*, pages 853–860, 2005.

[6] A. Criminisi. *Accurate Visual Metrology from Single and Multiple Uncalibrated Images*. Springer Verlag, 2001.

[7] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *SIGGRAPH*, pages 11–20, 1996.

[8] C. DeCoro, F. Cole, A. Finkelstein, and S. Rusinkiewicz. Stylized shadows. In *Proc. NPAR*, pages 77–83, 2007.

[9] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[10] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. pages 577–584, 2005.

[11] Y. Horry, K. ichi Anjyo, and K. Arai. Tour into the picture: using a spidery mesh interface to make animation from a single image. In *SIGGRAPH*, pages 225–232, 1997.

[12] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: a sketching interface for 3d freeform design. In *SIGGRAPH*, pages 409–416, 1999.

[13] D. Kersten, D. Knill, P. Mamassian, and Bulthoff. Illusory motion from shadows. *Nature*, 379(6560):31, 1996.

[14] J.-F. Lalonde, D. Hoiem, A. A. Efros, C. Rother, J. Winn, and A. Criminisi. Photo clip art. In *SIGGRAPH*, 2007.

[15] D. Liebowitz, A. Criminisi, and A. Zisserman. Creating architectural models from images. In *Proc. Eurographics*, pages 39–50, 1999.

[16] H. Lombaert, Y. Sun, L. Grady, and C. Xu. A multilevel banded graph cuts method for fast image segmentation. In *Proc. ICCV*, pages 259–265, 2005.

[17] D. Marr. *Vision: a computational investigation into the human representation and processing of visual information*. W. H. Freeman, San Francisco, 1982.

[18] A. Mohan, J. Tumblin, and P. Choudhury. Editing soft shadows in a digital photograph. *IEEE Comput. Graph. Appl.*, 27(2):23–31, March 2007.

[19] B. M. Oh, M. Chen, J. Dorsey, and F. Durand. Image-based modeling and photo editing. In *SIGGRAPH*, pages 433–442, 2001.

[20] M. Prasad and A. Fitzgibbon. Single view reconstruction of curved surfaces. In *Proc. CVPR*, pages 1345–1354, 2006.

[21] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, pages 309–314, 2004.

[22] T.-P. Wu, C.-K. Tang, M. S. Brown, and H.-Y. Shum. Natural shadow matting. *ACM Trans. Graph.*, 26(2):8, 2007.