Photo Clip Art

Jean-François Lalonde Derek Hoiem Alexei A. Efros Carsten Rother John Winn Antonio Criminisi
Carnegie Mellon University Microsoft Research Cambridge

Figure 1: Starting with a present day photograph of the famous Abbey Road in London (left), a person using our system was easily able to make the scene much more lively. There are 4 extra objects in the middle image, and 17 extra in the right image. Can you spot them all?

Abstract

We present a system for inserting new objects into existing photographs by querying a vast image-based object library, precomputed using a publicly available Internet object database. The central goal is to shield the user from all of the arduous tasks typically involved in image compositing. The user is only asked to do two simple things: 1) pick a 3D location in the scene to place a new object; 2) select an object to insert using a hierarchical menu. We pose the problem of object insertion as a data-driven, 3D-based, context-sensitive object retrieval task. Instead of trying to manipulate the object to change its orientation, color distribution, etc. to fit the new image, we simply retrieve an object of a specified class that has all the required properties (camera pose, lighting, resolution, etc) from our large object library. We present new automatic algorithms for improving object segmentation and blending, estimating true 3D object size and orientation, and estimating scene lighting conditions. We also present an intuitive user interface that makes object insertion fast and simple even for the artistically challenged.

Keywords: image databases, object insertion, blending and compositing, 3D scene reasoning, computational photography

1 Introduction

One of the biggest achievements of photography has been in bringing the joy of visual expression to the general public. The ability to depict the world, once the exclusive domain of artists, is now available to anyone with a camera. The sheer number of photographs being taken every day, as evidenced by the explosive growth of websites like Flickr, attests to people's need to express themselves visually. However, the creation of *novel* visual content is still the monopoly of a small select group: painters, computer graphics professionals, and Photoshop artists. One of the "Grand Goals" of computer graphics is to make the creation and manipulation of novel photo-realistic imagery as simple and effortless as using a word-processor to create beautifully typeset text is today.

However, there are a number of formidable challenges on the way toward this ambitious goal. For "traditional" computer graphics, the main problem is not the lack of good algorithms – recent advances in global illumination, material modeling, and lighting have allowed for synthesis of beautifully realistic and detailed imagery. Instead, the biggest obstacle is the sheer richness and complexity of our visual world. Every object to be rendered requires painstaking work by a skilled artist to specify detailed geometry and surface properties. And while there are a few databases of pre-made object models, typically the number of objects is quite small and the quality is far from photo-realistic. Various image-based approaches can also be used to acquire object models, but the data acquisition process is not straightforward and is typically not suitable for relatively large, outdoor, potentially moving objects (cars, pedestrians, etc). In this paper, we are particularly interested in being able to insert new objects into existing images (e.g. creating a movie storyboard or adding street life to a newly built city-block). While it is possible to add synthetic objects into real scenes [Debevec 1998], this requires estimating camera orientation and capturing environment maps – tasks too difficult for a casual user.

In this work, we advocate an alternative approach for creating novel visual content – by leveraging the enormous amount of photographs that has *already been captured*. A number of recent papers, such as Interactive Digital Photomontage [Agarwala et al. 2004], have demonstrated the power of using stacks of registered photographs to create novel images combining information from the entire stack. Last year's Photo Tourism work [Snavely et al. 2006] showed how hundreds of photographs acquired from the Internet can be used as a novel way to explore famous architectural landmarks in space as well as in time. However, these are all examples of using images taken at the same physical location – a necessarily limited set. What about exploiting all the available visual data in the world, no matter where it was captured, to build a universal *photo clip art* library that can be used for manipulating visual

Project Web Page: http://graphics.cs.cmu.edu/projects/photoclipart/

From the ACM SIGGRAPH 2007 conference proceedings.

ACM SIGGRAPH 2007, San Diego, CA

content "on the fly"? While this is a very ambitious goal, the recent emergence of large, peer-labeled object datasets [Russell et al. 2005; von Ahn et al. 2006], as well as advances in geometric scene understanding [Hoiem et al. 2005; Hoiem et al. 2006] suggest that efforts in this direction are very timely.

2 Overview

In this paper we propose a system for inserting new objects into existing photographs by querying a vast image-based object library, pre-computed using publicly available Internet object datasets. The central goal is to shield the user from all of the arduous tasks typically involved in image compositing – searching for a good object to cut out, manual cropping and resizing, color adjustment, edge blending, etc. The user is only asked to do two simple things: 1) pick a 3D location in the scene to place a new object; 2) select an object to insert using a hierarchical menu.

Two critical elements form the cornerstone of our approach to solving this challenging task:

Data-driven Object Placement. The difficulty in placing an image of an object (object sprite) into another image is that the camera orientation with respect to the object as well as the lighting conditions must match between the two images. One simple but key intuition is that while placing a *particular* object (e.g. my brown Volvo seen from the side) into a given image is a very difficult task, finding *some* instance of an object class (e.g. a car) that fits well is much easier. The idea, then, is to pose the problem of object insertion as a data-driven, context-sensitive object retrieval task. Instead of attempting to manipulate the object to change its orientation and/or color distribution, we simply retrieve an object of a specified class that has all the required properties (camera pose, lighting, resolution, etc) from our large object library. This turns an impossibly hard problem into a solvable one, with the results steadily improving as more and more data becomes available.

3D Scene-based Representation: One of the major weaknesses of most current image compositing approaches is that they treat it as a 2D Photoshop-like problem. We believe that any image manipulation must be done in the 3D space of the scene, not in the 2D space of the image. This does not mean that a complete 3D depth representation of a scene is required – only some rough qualitative information about surfaces and their orientations with respect to the camera, plus basic depth ordering. First, this allows the user some intuitive 3D control of the object placement process – placing the car further down the road will automatically make it smaller. More importantly, the 3D information is needed to build the photo clip art library. Each object in the library must be annotated with relative camera pose, whether it's occluded by other objects, whether it's on the ground plane, etc. In the current version, due to the limitations of the object dataset, all of our objects are assumed to reside on the ground plane. Finally, a coarse 3D layout of a scene appears sufficient to characterize its main illumination properties, which we use for matching the object lighting conditions.

2.1 Prior work

A number of papers have dealt with issues involved in compositing an object into an image in a visually pleasing way [Porter and Duff 1984; Perez et al. 2003; Jia et al. 2006; Wang and Cohen 2006]. However, they all assume that the user has already chosen the object that would be a good fit (geometrically and photometrically) for the target image. For instance, most blending techniques [Perez et al. 2003; Jia et al. 2006] rely on the assumption that the area surrounding the object in the source and target images are similar. Playing with these methods, one quickly realizes how very important the selection of a good source object is to the quality of the result and how difficult it is to get it right if one is not artistically gifted.

The work closest to our own and one of the main inspirations for this project is Semantic Photo Synthesis [Johnson et al. 2006] (which itself has been inspired by [Diakopoulos et al. 2004]). This paper develops a very ambitious approach for letting the user "design a new photograph" using textual and sketch queries into a database of real, automatically annotated images. Graphcut optimization [Boykov et al. 2001] is used to stitch together parts of different retrieved photographs to get the final resulting image. While this work is very sophisticated technically and loaded with good, novel ideas, unfortunately, the visual results are not very compelling. We believe that the reasons for this are three-fold: 1) the synthesis process is done purely in the 2D image plane, without regard to scene geometry; 2) the image parts being stitched together are often not well defined, i.e. not corresponding to physical objects; 3) there is no attempt to match illumination conditions between image parts. Our present work addresses all of the above issues. We employ a more incremental approach to image synthesis. Instead of starting from scratch and hoping to get a photo-realistic picture in the end, we begin with an image that is already real and try to alter it in ways that change the content without breaking this "realness".

2.2 Key Challenges

The main challenge of our proposed system is to make the resulting images look as real as possible. Here we highlight the issues that are critical for the success of this endeavor and summarize our approach for addressing them.

- **Rich Object Library:** The data-driven nature of our approach requires a library with very large number of labeled objects. After examining a number of available on-line object datasets, we have chosen LabelMe [Russell et al. 2005] as providing a good mix of object classes, large numbers of objects in each class, and reasonably good initial object segmentations. After post-processing and filtering the data, we end up with an object library currently containing 18 categories of objects with over 13,000 object instances.
- **Object Segmentation:** Getting clean object boundaries is critical for seamless object insertion. While the object dataset we use provides rough polygonal object outlines, this is rarely good enough for compositing. To address this problem, we have developed an automatic segmentation and blending approach. We extend the popular graph-cut/Poisson blending framework by introducing a new shape-sensitive prior, adding a crucial local context-matching term for blending, detecting shadows, and modifying the blending process to prevent severe discoloration.
- **Estimating true Object Size and Orientation:** A 2D object boundary tells us nothing about the size of the object in the world or its orientation to the viewer. For example, given an outline of a person, we cannot tell if it's a tall man standing far away, or a small boy standing nearby; or whether we are viewing him from street level or from a third-story balcony. But without this information one cannot hope for realistic object insertion. We have developed an automatic algorithm that uses all the objects labeled in the dataset to estimate the camera height and pose with respect to the ground plane in each of the images. For objects resting on the ground (our focus in this paper), these two parameters are enough to compute object size and orientation.
- **Estimating Lighting Conditions:** Consistency in lighting is another important cue for realistic compositing. A picture of a car taken at sunset will most likely not look right when placed into a midday scene. Unfortunately, we do not have enough information to compute a full environment map for each object. However, such accuracy might not be necessary. It is well-known that painters can often get away with substantially inconsistent lighting [Cavanagh 2005], and in graphics applications simple

Figure 2: Automatic object height estimation. Objects taken from a typical image in LabelMe dataset (left) are first shown in their original pixel size (center), and after being resized according to their automatically estimated 3D heights (right).

approximations have been shown to perform well [Khan et al. 2006]. In this paper, we propose a simple illumination model based on automatically computing color distributions of major surfaces (e.g. ground, vertical planes, sky) in the scene.

- **Intuitive User Interface:** While not the central focus of this paper, a good UI is critical for achieving the goal of making our system simple and intuitive. We have designed and implemented a full-functioning GUI that allows for easy browsing of the object library and lets users insert objects with a single click.

The remainder of the paper is devoted to describing each of these components in more detail. We first discuss our approach for creating the photo clip art object library (Section 3). We then present our system for inserting objects into existing user photographs (Section 4). Finally, we demonstrate some results (Section 5).

3 Creating the Photo Clip Art Library

The first issue in creating a photographic clip art library is finding a large dataset of labeled images of objects. Fortunately, the growth of the Internet and the need for large object recognition testbeds for computer vision research have produced a large number of freely available object datasets that we can choose from.

Most datasets (such as Caltech-101 [Fei-Fei et al. 2004], MSRC [Shotton et al. 2006], and PASCAL [Everingham et al. 2006]) have been collected and annotated by small groups of researchers and are therefore limited in size and/or annotation detail (most lack segmentations). Recently, there appeared a new breed of peer-labeled datasets. The most famous example is Flickr, where people can submit photographs and annotate them with semantic image and region tags. A creative approach is presented in an on-line game called Peekaboom [von Ahn et al. 2006], where humans communicate with each other by segmenting objects in images. Finally, LabelMe [Russell et al. 2005] provides as on-line tool for vision researchers to submit, share, and label images.

After experimenting with all these datasets, we decided that LabelMe is the best-suited for our purposes. Its advantages include a large number of images (over 30,000 at present), containing many different objects (over 2,000 object classes) as well as reasonably good segmentations. Also important for us is that most photographs are not close-ups of one individual object but depict scenes containing multiple objects, which enables us to estimate camera orientation. Finally, since this dataset is very popular in the vision community, it is constantly growing in size. At the same time, vision researchers are developing automatic methods for supervised recognition (e.g. [Shotton et al. 2006]) and object discovery (e.g. [Russell et al. 2006]) that should, some day, take over the labeling chore. Success has already been reported on specialized tasks (e.g. an automatically collected database of celebrity faces [Berg et al. 2004]), but further research is needed for more general domains.

The main downside of LabelMe is substantial variation between individual labeling styles as well as semantic tags. Some people include cast shadows, others don't; some people include occluded parts of an object, others don't. Since a semantic tag can be any text string, the same object could be labeled as: "person", "human", "pedestrian", "man smiling", etc. Therefore, one of our tasks in processing the dataset was to deal with such inconsistencies. Additionally, for every object in our library, we need to estimate its size and orientation in the real world as well as its illumination. We will now describe each of these steps in detail.

3.1 Estimating object size and orientation

To match and scale objects in our database to the background scene provided by the user, we need to know the true size of the objects in the world and the pose of the camera. As shown by [Criminisi et al. 2000], if we know the vanishing line of the ground plane, we can determine the relative heights of objects that rest on the ground plane. To annotate our database using this method, however, would require painstaking (and often inaccurate) annotation of vanishing lines in thousands of images. Instead, we provide a simple method to gradually *infer* camera pose and object heights across our database when provided with only the height distribution for *one* object class.

Let y_i be the 3D height of an object with a 2D height of h_i and a vertical position of v_i (measured from the bottom of the image). The camera pose is denoted by y_c and v_0, which correspond roughly to the camera height and the horizon position. Following Hoiem et al. [2006], our inference is based on the relationship $y_i = \frac{h_i y_c}{v_0 - v_i}$, assuming that objects stand on the ground and that ground is not tilted from side to side and roughly orthogonal to the image plane. Thus, given the position and height of two objects of known type in the image, we can estimate the camera pose without heavily relying on a prior distribution and use it to compute the 3D height of other objects in the scene. If we know the prior distribution over camera pose, we can provide a good estimate of it, even if only one instance of a known object is present [Hoiem et al. 2006].

Our procedure is as follows. We first compute the most likely camera pose for images that contain at least two known objects (instances of an object class with a known height distribution). From these, we estimate a prior distribution over camera pose. Using this distribution, we infer the likely camera pose for images that contain only one known object. Finally, we compute heights of objects in the images for which camera pose is known and estimate the height distributions of new classes of objects. We then iterate this process. As the height distributions of new object classes are discovered, they can be used to estimate the camera pose in additional images, which, in turn, allow the discovery of new objects.

We initialize by specifying that the height of people is normally distributed with a mean of 1.7 meters and a standard deviation of 0.085. We also initialize a loose prior of camera pose with

(a) Input image (b) with well matched objects (c) with poorly matched objects (d) Car source images

Figure 3: Lighting plays crucial role in object insertion. Given an input image (a), objects that are similarly illuminated fit seamlessly into the scene (b), while these with substantially different illumination appear out of place (c). By considering the source image from which an object was taken (d) in relation to the input image (a), our algorithm can predict which objects will likely make a realistic composite.

mean horizon of mid-image and mean camera height of 1.6 meters (roughly eye level), with a diagonal covariance of 1. Once we estimate the camera pose for images containing at least two people, we use those poses to model the prior with a Gaussian mixture model (with three full-covariance components). When we can infer the height of at least fifteen instances of an object class, we estimate the normal distribution of heights for that object. To avoid being confused by poorly segmented objects and other outliers, we ignore objects that are measured as more than twice as large or twice as small as the median height for their classes.

After several iterations, we infer the camera pose of about 5,000 images containing over 13,000 object instances, for which the 3D height can now be computed. As by-products, we also have a good estimate of the camera pose prior, as well as the height distributions of roughly fifty object classes ranging from cars to fire hydrants to horses. The accuracy of our estimates are confirmed by manual measurements of camera pose on a subset of images, as well as by the inferred mean object heights (e.g., 1.67m for "women", 1.80m for "men", 1.37m for "parkingmeter", which are all within two cm of their true values). Figure 2 shows an example of automatically estimate 3D heights for various objects from the same image.

3.2 Estimating lighting conditions

The appearance of an object is determined by the combination of its reflectance properties and the illumination from the surrounding environment. Therefore, an object taken from one image and inserted into another will look "wrong" if the two scenes have large differences in illumination (see Figure 3). The standard solution of measuring the incident illumination by capturing the object environment map (typically using a mirrored sphere at the object location [Debevec 1998]), is not possible here since we only have access to a single photograph of the object. However, it has been shown by Khan et al. [2006] that a usable environment map approximation can be obtained from the image itself, in their case, by extracting a disk around the object, making it into a hemisphere and mirroring.

We propose to use scene understanding techniques to compute a very coarsely sampled environment map given a single image as input. The basic idea is to come up with a rough 3D structure of the depicted scene and use it to collect lighting information from the three major zenith angle directions: from above, from below, and from the sides (azimuth direction is not used since there is not enough data in a single image). Our implementation uses the geometric context of [Hoiem et al. 2005] to automatically estimate 3 major surface types: ground plane, vertical planes, and the sky. The distribution of illuminations within each surface type is computed as a joint 3D histogram of pixel colors in the *CIE* $L^\star a^\star b^\star$ space, for a total of 3 histograms to form our scene *illumination context*. It is important to note that while [Khan et al. 2006] use high dynamic range images, we only have access to regular low dynamic range photographs so cannot capture true luminance. But since we don't use the illumination context to relight a new object, only to compare lighting conditions between images, this has not been a problem.

The illumination context is too global and cannot encode location-specific lighting within the scene, i.e. whether a given object is in shadow. Therefore, for each object we also compute a *local appearance context* – a simple color template of its outline (30% of object diameter outward from the boundary, and 5% inward). The local context is used as an additional cue for object placement.

3.3 Filtering and grouping objects

While the LabelMe dataset has a rich collection of objects and object classes, many are simply not useful for our purposes. A lot of the labeled objects are occluded or are actually object parts (e.g. a wheel, a car window, a head), while others are amorphous (e.g. sky, forest, pavement). The types of objects best suited for a clip art library are semantically whole, spatially compact, with well-defined contact points (i.e. sitting on the ground plane). We attempt to filter out the unsuitable objects from the dataset in a semi-automatic way.

As a first step, we try to remove all non-whole objects by searching the tag strings for words "part", "occlude", "region", and "crop". We also remove any objects that have a boundary right at the image border (likely occluded). Second, we estimate the 3D heights of all objects using the iterative procedure outlined in Section 3.1. As a by-product, this gives us a list of object classes that have been found useful for camera pose estimation (i.e. having relatively tight height distributions): "bench", "bicycle", "bus", "car" (various orientations), "chair", "cone", "fence", "fire hydrant", "man", "motorcycle", "parking meter", "person", "sofa", "table", "trash can", "truck", "van", "wheel", "woman". Of these, we discard small or unsuitable classes (e.g. "wheel"), and merge others (e.g. "man", "woman", and "person") into consistent groups. We also manually add a few fun classes that were either too small or not very consistent: "plant", "flowers", "tree", and "manhole". Note that most of the recovered classes are of outdoor objects. This mainly reflects the bias in the dataset toward outdoor scenes, but also due to ground-based objects mostly appearing outdoors.

In most cases the variance between objects within the same object class is still too high to make it a useful way to browse our clip art collection. While a few of the object classes have hand-annotated subclass labels (e.g. cars are labeled with their orientation), most do not. In these cases, clustering is used to automatically find visually similar subclasses for each object class. First, all objects are rescaled to their world 3D height (as calculated earlier), so that, for instance, a tree and a bush don't get clustered together. Next, k-means clustering using $L2$ norm is performed on the binary object outlines. This is done so that object will cluster solely based on shape, and not on their (lighting dependent) appearance. Results show successful subclasses being formed for most of the objects (e.g. see Figure 4).

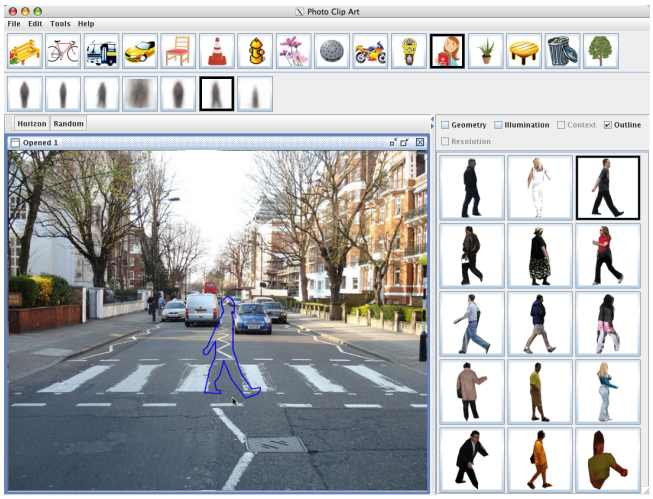

Figure 4: User Interface for our system. The two-level menu on the top panel allows users to navigate between different classes and subclasses of the photo clip art library. The side panel on the right shows the top matched objects that can be inserted into the scene.

4 Object Insertion

Now that we have built our photo clip art object library, we can describe how to insert objects into user photographs. Given a new photograph, the first task is to establish its camera orientation with respect to the ground plane. To do this, the user is asked to pick the location of the horizon with a simple line interface. Other relevant camera parameters (camera height, focal length, pixel size) are set to their most likely values, or extracted from the EXIF tag, if available. Alternatively, the user can specify the height of one or more known objects in the scene, from which the camera parameters can be estimated as in Section 3.1. It is also possible to let the user resize a known "reference object", such as a person, placed at different locations in the image, to compute the camera pose and height, although we haven't found this necessary. In addition, lighting conditions must be estimated for the new scene, which is done in the same way as described in Section 3.2. With the critical parameters of the new photograph successfully estimated, we are now ready for object insertion.

4.1 User Interface

First, we will briefly describe the design of our User Interface. Its job is to let the user 1) easily navigate the entire photo clip art library, and 2) paste an object of choice into the photograph with a single click.

The screen-shot of our UI is depicted on Figure 4. The user photograph is loaded in the central panel. The top row of icons shows all the object classes available in the library. The second row of icons shows all object subclasses for a chosen class. The subclasses are generated either from the object annotations, if available, or by automatic clustering (as described previously). The subclass icons are average images of all the objects in the subclass, a technique inspired by [Torralba and Oliva 2003]. The right-hand-side panel displays the object instances. Initially it contains all the objects in the library, but as the user picks a class of objects, only these instances remain in the panel. The same happens for a subclass. Moreover, the objects in the panel are sorted by how well they match this particular scene.

To insert an object into the scene, the user first chooses an object class (and optionally a subclass), and then picks one of the top matches from the object panel. Now, as the user moves the mouse around the photograph, the outline of the picked object is shown, becoming bigger and smaller according to the correct perspective. A single click pastes the object at the current mouse position, and full segmentation and blending is performed on the fly. Alternatively, the user can first pick a location to paste an object, and then choose among the objects that best fit that particular location. Note that all inserted objects (as well as any labeled ones in the image) preserve their 3D depth information and are rendered back-to-front to accurately account for known occlusions.

4.2 Matching Criteria

Given a scene, we need to order all the objects in the library by how natural they would look if inserted into that scene. This way, a user picking an object from near the top of the list would be pretty confident that the compositing operation would produce good results. The ordering is accomplished as a weighted linear combination of several matching criteria, dealing with different aspects of object appearance.

Camera Orientation: The camera orientation with respect to the object's supporting surface has been pre-computed for every object in the library. We also have the estimate for the ground plane orientation for the user's photograph. Geometrically, an object would fit well into the scene if these camera orientation angles are close, and poorly otherwise. If the object placement location is not specified explicitly by the user, a simple difference between the vanishing line heights of the ground planes of the source and destination scenes is used instead.

Global Lighting Conditions: Each object in the library has an associated scene illumination context that was computed from the object's original source image. An object is most likely to appear photometrically consistent with the new scene if its original scene illumination context match that of the new scene. The distance between the illumination contexts is computed as a weighted linear combination of χ^2-distances between the $L^\star a^\star b^\star$ histograms for sky, vertical, and ground. Experimenting with the weights, we have found (unsurprisingly) that the sky portion of the context is the most important, followed by the ground portion. Vertical portion seems to have little effect.

Local Context: If a user has also specified the location where he wants the object to be placed, local appearance context around (and under) the placement area can also be used as a matching criterion. The context is matched to the corresponding pixels at the placement area using SSD measure. This cue is used primarily as an aid to blending difficult objects, as described in Section 4.3, and for matching locally-varying lighting effects.

Resolution: Again, if the placement location in the image is given, it is easy to compute the desired pixel resolution for the object so as to minimize resizing.

Segmentation Quality: The initial segmentation quality of objects in our library is highly variable. Various approaches can be used to try separating the good ones from the bad (e.g. alignment with brightness edges, curvature, internal coherence, etc). In this implementation, we use a very simple but effective technique, ordering objects by the number of vertices (i.e. labeling clicks) in their boundary representation.

4.3 Object Segmentation and Blending

Once the user picks an object to insert into a given scene, the most important task to insure a quality composite is object matting. Unfortunately, automatic alpha-matting without a user in the loop is not guaranteed to work reliably for all scenarios, despite a lot of recent advances in this field e.g. [Levin et al. 2006]. On the other hand, blending techniques [Perez et al. 2003; Jia et al. 2006] have shown that impressive results can be achieved without an accurate extraction of the object, assuming that the material surrounding the object and the new background are very similar (e.g. a sheep on a lawn can be pasted onto another lawn, but not on asphalt).

Very recently, the idea of combining the advantages of blending

Figure 6: Example of image compositing: Source input image (a) and compositing results with different approaches (b,c,e-g), where red arrows point out problems. See text for full explanation.

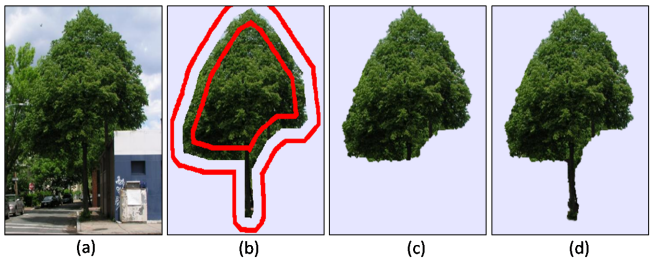

Figure 5: Object Extraction: Given an image (a), the task is to improve the crude polygon-shaped LabelMe segmentation (b). Standard GrabCut (restricted to a red band in (b)) suffers from a "shrinking" bias (c). This can be overcome by introducing flux, a shape-sensitive prior, into the GrabCut framework (d).

and alpha-matting have been introduced to hide matting artifacts as much as possible [Wang and Cohen 2006; Jia et al. 2006]. The key advantage of our system is that the local context criteria (introduced above) has already retrieved appropriate objects which match to some degree the given background, so blending and segmentation have a good chance of supporting each other.

Our compositing method runs, as in [Jia et al. 2006], in three steps: first object extraction, then blending mask computation, and finally Poisson image blending. The novel ideas over existing approaches are: a crucial shape prior for segmentation, a new context-matching term for blending mask extraction, and a way to prevent severe discoloration during blending.

4.3.1 Object Segmentation with a shape prior

Object segmentations provided in the LabelMe dataset have only crude, polygonal boundaries. The task is to find a more faithful outline of the object along with a good alpha mask. For this we extend the popular graph-cut based segmentation approaches [Rother et al. 2004; Li et al. 2004] with a prior for the given shape. Figure 5 gives an example, where the cut is restricted to lie in a small band (red in fig. 5(b)) around the prior polygon shape. Applying GrabCut without shape prior results in fig. 5(c). The tree outline looks convincing, however the tree trunk is lost. This happens because a short expensive boundary has a lower cost than a very long cheap one, sometimes called the "shrinking" bias. To overcome this problem different forms of shape priors, such as shape-based distance transform [Boykov et al. 2006], have been suggested in the past. We use the flux shape prior as introduced recently in the theoretical work of [Kolmogorov and Boykov 2005]. The main advantage over other methods is that it overcomes the shrinking bias while not over-smoothing the whole segmentation. The key idea is that the gradient of the computed contour is similar to the gradient of the given shape. To achieve this, we add an additional unary term to eq. 2 in [Rother et al. 2004] of the form $w_{flux} div(F)$, i.e. the divergence of the vector field F, with the weight $w_{flux} = 25$. The vector field is a weighted gradient field of the distance transform of the shape mask. Precisely, $F = \nabla D \ \ (exp(-|D|/\sigma))$, where $\sigma = 3$, and D is the distance transform of the given contour, with positive distance inside and negative outside. The weighting of F by the magnitude of D is important to favor contours close to the given one. Also, before computing the divergence of F, we smooth the vector field F with a Gaussian (variance 1.5) which essentially smooths the initial polygon shape. The result in fig. 5(d) shows that with flux the tree trunk is recovered nicely.

Finally, in order to recover the full alpha matte of the foreground object we use the border matting system introduced in [Rother et al. 2004]. It is conservative in the sense that it is insensitive to image noise, however, is limited to recovering only simple alpha mattes along the boundary of sharp but not fuzzy edges. The later limitation can be, to some extent, overcome by our context-sensitive blending procedure introduced below.

4.3.2 Context-sensitive Blending

Consider the image compositing task in fig. 6. Given the input image (a), GrabCut (based on a rough polygonal outline around the object) produced an alpha matte which is used to paste the object onto a new background (b). Here conservative border matting [Rother et al. 2004] was not able to extract the hair perfectly. On the other hand, standard Poisson image blending [Perez et al. 2003] (with the boundary constraint applied at the green line in (d) – a dilation of the GrabCut result) blends the hair correctly (c). However, it produces two artifacts: a severe discoloration due to large differences at the boundary constraint, and a blurry halo around the object where foreground and background textures do not match. The key idea is to find a binary blending mask which either blends in the background color or leaves the segmentation unaltered. Figure 6(d) shows the blending mask computed by our technique. In places where the mask coincides with the object (red line) no blending is performed and when the mask is outside the object (at the hair) it blends in the background. This gives a visually pleasing result (e). To achieve this we extend the approach of [Jia et al. 2006] by adding an additional regional term that measures the similarity of foreground and background statistics (see [Rother 2007] for details). This allows the white mask (d) to grow outside the red mask at places with high similarity. In comparison, results using digital photomontage [Agarwala et al. 2004] and drag-and-drop pasting [Jia et al. 2006] (both using our implementation) show more artifacts as seen on Figure 6(f-g). For [Agarwala et al. 2004], we used their Poisson blending technique, which switches off high gradients along the boundary. Note that their result cuts off the hair since the sky in the foreground and background images are differently colored and it has a bias toward short boundaries.

Finally, we modify the standard Poisson image blending [Perez et al. 2003] to prevent severe discoloration. This is desirable since the scene lighting criterion, introduced above, retrieved objects which are already photometrically consistent with the background scene. Poisson blending is expressed as an optimization over the constructed image u that matches its gradient ∇u most closely to the foreground gradient ∇I^F, subject to matching the background

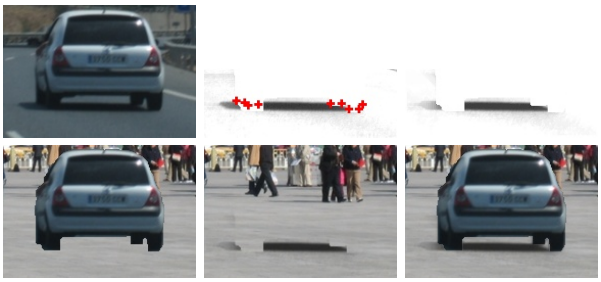

Figure 7: Shadow Transfer. On the top row, we show the source image (left) for a car with initial (center) and final (right) shadow estimates. Our initial estimate, simply based on intensity relative to surrounding area, is refined by enforcing that the shadow is darker closer to the contact points (red '+'). On the bottom row, we show the background patch with the object (left), with the matted shadow (center), and the combination (right), before blending.

image I^B on the boundary of the blending mask $\mathbf{s}$. Let us instead optimize the following function, inspired by [Rother et al. 2006]:

$$E(u) = \lambda \int_{\mathbf{s}} (u - I^F)^2 + \int_{\mathbf{s}} w_{\mathbf{s}}(\nabla I^F) \, \|\nabla u - \nabla I^F\|^2 \qquad (1)$$

where $w_{\mathbf{s}}(\nabla I^F) = 1 + \gamma \exp -\frac{\beta}{2g_{\mathbf{s}}} \|\nabla I^F\|^2$, with $g_{\mathbf{s}} = \left\langle \|\nabla I^F\|^2 \right\rangle_{\mathbf{s}}$. Here $< \cdot >$ denotes a mean value over the domain $\mathbf{s}$; and we set $\lambda = 0.05, \gamma = 800, \beta = 2.5$. The first integral forces the object to retain its original color, controlled by the weight λ. The weighting $w_{\mathbf{s}}$ is low when the gradient $\|\nabla I^F\|$ is very high, which reduces the possibility of severe discoloring.

The blending mask and fractional object mask might intersect in some places (e.g. person's shoulders in fig. 6). We use the approach of [Jia et al. 2006] to handle this case by incorporating the alpha mask in the guidance field for Poisson blending. This means that if the blending mask coincides with the object mask the boundary conditions are perfectly met and no discoloring occurs.

4.4 Transferring Shadows

While psychologists have shown that humans are not very sensitive to incorrect cast shadow direction [Cavanagh 2005], the mere presence of a shadow is a critical cue for object/surface contact [Kersten et al. 1996]. Without it, objects appear to be floating in space. Shadow extraction is a very challenging task and most existing work either requires controlled lighting conditions, e.g. [Chuang et al. 2003], or makes restrictive assumptions about camera, lighting, and shadow properties (e.g. [Finlayson et al. 2006]).

Because we do not know the 3D structure of our objects, synthetically generated shadows rarely look convincing. Instead, we take an image-based approach, transferring plausible shadows from the source image to the background image (illustrated in Figure 7). We consider the ground to be a mostly homogeneous surface, with the object blocking some percentage of the light from reaching an area of that surface. Our goal is to determine the percentage intensity drop at each image position due to the object's shadow.

Our method relies on the tendency of shadows to be darker than the surrounding area and to attenuate with distance from object-ground contact points. Our initial shadow estimate is simply the pixel intensity divided by the non-shadowed ground intensity (with a maximum value of one). We approximate the non-shadowed ground intensity on each row as the median intensity of pixels within a margin (one-quarter object width) around the object, excluding pixels within and directly beneath the object region. Our initial estimate will typically contain errors due to non-homogeneity of the ground surface or shadows that are cast from other objects.

We can reduce these errors by enforcing that the shadow becomes monotonically weaker with distance from the ground contact points. To allow such reasoning, our system automatically classifies each point on the polygon of an object as contact (touching the ground) or not. Such classification is very difficult based on 2D image positions and shapes alone. Our insight is to project each point of the polygon onto the ground plane, providing additional cues for contact point detection. For instance, if a ground-projected point is at a much greater depth than the base of the object, the point is not likely to contact the ground. Our classifier cues include: 2D position relative to the bounding box; 3D position relative to minimum object depth and object width; slope to neighboring points (both 2D and 3D); and whether the point lies on the convex hull of the object region. We labeled the contact points of 200 objects and trained a decision tree classifier [Quinlan 1993], which achieves cross-validation accuracy of 96%.

Finally, to avoid artifacts caused by surrounding objects and their shadows, we limit the shadow darkness to the 90th percentile of darkness over all shadows of an object class in our database, which is computed for each object class using the method described above, relative to the position in the bounding box. We mat the shadow onto the background image by darkening the area around the object by the same percentage as in the source, which is then used in the blending process described above.

5 Results and Discussion

Figures 8, 9 and 10 show more object insertion results that a user was able to create with our system. One needs to look very carefully to notice all the objects that have been added – there are several that are quite difficult to spot. As can be seen, the system handles input images with wide variation in scene geometry as well as lighting conditions. In addition to photographs, our approach can successfully be applied to paintings (Figure 8, bottom center) and CG renderings (Figure 10). The important thing to note is that all these examples were produced in a few minutes by users who are not at all artistically skilled.

In this paper, we have argued for a philosophically different way of thinking about image compositing. In many situations, it is not important which particular instance of an object class is pasted into the image (e.g. when generating realistic architectural renderings as shown on Figure 10). In such cases, what is desirable is a kind of photo-realistic clip art library. Of course, regular clip art is a simple 2D iconic drawing, whereas the real 3D world-based clip art is necessarily more complex. However, we have shown that with the right approach, and a lot of data, this complexity could be successfully hidden from the end user, making it seem as simple and intuitive as regular clip art.

The paper presents a complete system, from database preprocessing to cutting and blending to the graphical user interface. Along the way, we had to solve many difficult issues, resulting in a number of novel algorithms and techniques. Since this is a complex system, failures can occur in several stages of processing. Figure 11 illustrates the three most common causes of encountered failures. First, input scenes with unusual illumination are not similar to any of the images in our library. Therefore, even the best-matched objects will look unrealistic. Second, blending and automatic segmentation errors may occur when objects are porous or of complex shape (e.g. trees and bicycles). These objects contain holes through which their original background is visible. Finally, the shadow transfer algorithm may fail when the object/ground contact points are not estimated correctly, or when something else in the scene is casting a shadow onto the object. Since most of these issues are data-related, it is reasonable to believe that as the underlying image datasets improve, so will our system. This, once again, underscores the main theme of this paper: that the use of large-scale image databases is a promising way to tackle some of the really difficult problems in computer graphics.

Figure 8: Some example images that were created by a user with our system. Can you find all the inserted objects?

Figure 9: Our system can handle a variety of input lighting conditions, from midday (left) to sunset (middle), and can even work on black&white photographs (right). In the latter case the objects have been converted to grayscale after insertion.

Figure 10: Application to architectural renderings. Here, a rendered view of the future Gates Center at CMU (left) is made to look more natural using our system (right). Note that the photo clip art people look much better than the white cutout people of the original rendering. [image credit: Mack Scogin Merrill Elam Architects]

6 Acknowledgments

This work would not have been possible without the efforts of Bryan Russell, Antonio Torralba and the rest of the LabelMe team in designing the labeling tool and maintaining the database. Yue (Annie) Zhao (HCI Institute, CMU) has been instrumental in helping us design and implement the GUI. We also like to thank Victor Lempitsky, Vladimir Kolmogorov, and Yuri Boykov for sharing their insights on using flux for image segmentation. We thank our photo contributers: Antonio Torralba, Robert Kitay, and the Flickr users who placed their work under Creative Commons License: Dey Alexander, M. Caimary, Dan Johansson, and `slightly-less-random`. We also thank Chris Cook (`www.chriscookartist.com`) for letting us use his painting *Blue Liquor Store*. And special thanks to Henry Rowley of Google Inc. for finding our needle in their haystack. This work has been partially supported by NSF grants CCF-0541230 and CAREER IIS-0546547, and a generous gift from the Microsoft Corp.

References

AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., DRUCKER, S., COLBURN, A., CURLESS, B., SALESIN, D., AND COHEN, M. 2004. Interactive digital photomontage. *ACM Trans. Graph. (SIGGRAPH 04) 23*, 3, 294–302.

BERG, T. L., BERG, A. C., EDWARDS, J., MAIRE, M., WHITE, R., TEH, Y.-W., LEARNED-MILLER, E., AND FORSYTH, D. A. 2004. Names and faces in the news. In *IEEE Computer Vision and Pattern Recognition (CVPR)*.

BOYKOV, Y., VEKSLER, O., AND ZABIH, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Analysis and Machine Intelligence 23*, 11.

BOYKOV, Y., KOLMOGOROV, V., CREMERS, D., AND DELONG, A. 2006. An integral solution to surface evolution PDEs via Geo-Cuts. In *European Conf. on Computer Vision (ECCV)*.

CAVANAGH, P. 2005. The artist as neuroscientist. *Nature 434* (March), 301–307.

CHUANG, Y.-Y., GOLDMAN, D. B., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2003. Shadow matting and compositing. *ACM Transactions on Graphics (SIGGRAPH 03) 22*, 3 (July), 494–500.

Figure 11: Typical failure modes of our system. For input scenes with unusual illumination, the object library may not contain illumination conditions that are similar enough, making even the best-matched objects look wrong (left). Additionally, our blending algorithm may fail to blend complex and porous objects (e.g. tree and bicycle, right). Shadow transfer may also yield undesirable results (e.g. car, right).

CRIMINISI, A., REID, I., AND ZISSERMAN, A. 2000. Single view metrology. *International Journal of Computer Vision 40*, 2, 123–148.

DEBEVEC, P. 1998. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of SIGGRAPH 98*, 189–198.

DIAKOPOULOS, N., ESSA, I., AND JAIN, R. 2004. Content based image synthesis. In *Conference on Image and Video Retrieval (CIVR)*.

EVERINGHAM, M., ZISSERMAN, A., WILLIAMS, C., AND GOOL, L. V. 2006. The pascal visual object classes challenge 2006 results. Tech. rep., Oxford University.

FEI-FEI, L., FERGUS, R., AND PERONA, P. 2004. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *IEEE CVPR Workshop of Generative Model Based Vision*.

FINLAYSON, G. D., HORDLEY, S. D., LU, C., AND DREW, M. S. 2006. On the removal of shadows from images. *IEEE Trans. Pattern Analysis and Machine Intelligence 28*, 1, 59–68.

HOIEM, D., EFROS, A. A., AND HEBERT, M. 2005. Geometric context from a single image. In *International Conference on Computer Vision (ICCV)*.

HOIEM, D., EFROS, A. A., AND HEBERT, M. 2006. Putting objects in perspective. In *IEEE Computer Vision and Pattern Recognition (CVPR)*.

JIA, J., SUN, J., TANG, C.-K., AND SHUM, H.-Y. 2006. Drag-and-drop pasting. *ACM Transactions on Graphics (SIGGRAPH 06) 25*, 3 (July), 631–637.

JOHNSON, M., BROSTOW, G. J., SHOTTON, J., ARANDJELOVIĆ, O., KWATRA, V., AND CIPOLLA, R. 2006. Semantic photo synthesis. *Computer Graphics Forum (Proc. Eurographics) 25*, 3, 407–413.

KERSTEN, D., KNILL, D., MAMASSIAN, P., AND BULTHOFF, I. 1996. Illusory motion from shadows. *Nature 379*, 6560, 31–31.

KHAN, E. A., REINHARD, E., FLEMING, R. W., AND BÜLTHOFF, H. H. 2006. Image-based material editing. *ACM Transactions on Graphics (SIGGRAPH 06) 25*, 3 (July), 654–663.

KOLMOGOROV, V., AND BOYKOV, Y. 2005. What metrics can be approximated by Geo-Cuts, or global optimization of length/area and flux. In *International Conference on Computer Vision (ICCV)*.

LEVIN, A., LISCHINSKI, D., AND WEISS, Y. 2006. A closed form solution to natural image matting. In *Proc IEEE Computer Vision and Pattern Recognition (extended Tech. Rep.)*.

LI, Y., SUN, J., TANG, C.-K., AND SHUM, H.-Y. 2004. Lazy snapping. *ACM Transactions on Graphics (SIGGRAPH 04) 23*, 3 (Aug.), 303–308.

PEREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Trans. Graph. (SIGGRAPH 03) 22*, 3, 313–318.

PORTER, T., AND DUFF, T. 1984. Compositing digital images. In *Computer Graphics (Proceedings of SIGGRAPH 84)*, 253–259.

QUINLAN, J. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, Inc.

ROTHER, C., KOLMOGOROV, V., AND BLAKE, A. 2004. GrabCut: interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (SIGGRAPH 04) 23*, 3 (Aug.), 309–314.

ROTHER, C., BORDEAUX, L., HAMADI, Y., AND BLAKE, A. 2006. Autocollage. *ACM Transactions on Graphics (SIGGRAPH 06) 25*, 3 (July), 847–852.

ROTHER, C. 2007. Cut-and-paste for photo clip art. Tech. Rep. MSR-TR-2007-45, Microsoft Research.

RUSSELL, B. C., TORRALBA, A., MURPHY, K. P., AND FREEMAN, W. T. 2005. LabelMe: a database and web-based tool for image annotation. Tech. rep., MIT.

RUSSELL, B. C., EFROS, A. A., SIVIC, J., FREEMAN, W. T., AND ZISSERMAN, A. 2006. Using multiple segmentations to discover objects and their extent in image collections. In *IEEE Computer Vision and Pattern Recognition (CVPR)*.

SHOTTON, J., WINN, J., ROTHER, C., AND CRIMINISI, A. 2006. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conf. on Computer Vision (ECCV)*.

SNAVELY, N., SEITZ, S. M., AND SZELISKI, R. 2006. Photo tourism: exploring photo collections in 3D. *ACM Trans. Graph. (SIGGRAPH 06) 25*, 3, 835–846.

TORRALBA, A., AND OLIVA, A. 2003. Statistics of natural image categories. *Network: Computation in Neural Systems 14*, 3 (August), 391–412.

VON AHN, L., LIU, R., AND BLUM, M. 2006. Peekaboom: A game for locating objects in images. In *ACM CHI*.

WANG, J., AND COHEN, M. 2006. Simultaneous matting and compositing. Tech. Rep. MSR-TR-2006-63.