

BUILDUP: Interactive Creation of Urban Scenes from Large Photo Collections

Jose Ribelles · Diego Gutierrez ·
Alyosha Efros

Received: date / Accepted: date

Abstract We propose a system for creating images of urban scenes composed of the large structures typical in such environments. Our system provides the user with a precomputed library of image-based 3D objects, such as roads, sidewalks and buildings, obtained from a large collection of photographs. When the user picks the 3D location of a new object to insert, the system retrieves objects that have all the required properties (location, orientation and lighting). Then, the user interface guides the user to add more objects enabling non-experts to make a new composition in a fast and intuitive way. Unlike prior work, the entire image composition process is done in the 3D space of the scene, therefore inconsistent scale or perspective distortion does not arise, and occlusions are properly handled.

Keywords Image databases · 3D scene composition · Picture generation

1 Introduction

The promise of effortless visual content creation has always been a large part of the appeal of computer graphics. The huge popularity of contemporary world-building games like Minecraft has demonstrated people's desire for new ways to create, to build, to express themselves visually. But the computer graphics employed in such games is still quite rudimentary. Bruce Branit's short film

J. Ribelles
Institute of New Imaging Technologies, Universitat Jaume I, Castellon, Spain
E-mail: ribelles@uji.es

D. Gutierrez
Universidad de Zaragoza, Zaragoza, Spain
E-mail: diegog@unizar.es

A. Efros
University of California at Berkeley, Berkeley, USA
E-mail: efros@eecs.berkeley.edu



Fig. 1 A person using our system was easily able to build the urban picture shown on the left image, by retrieving appropriate 3D objects shown on the right image.

”World Builder” [3] shows us a future where intricately detailed, breathtakingly photorealistic urban CG worlds could be created by a single user, in under an hour. Development of such technologies for quickly visualizing and experimenting with future urban spaces that do not yet exist, or for creating synthetic city scenes that do not depict a real place would benefit fields such as urban planning or landscape architecture, in applications like revitalizing downtowns or historic places, designing new neighborhoods, planning cities, retrofitting suburbia or shaping transit oriented development. In this paper, our aim is to take a small step toward this future (see Figure 1).

In the past decade, the use of large collections of photographs has been proposed to deal with the synthesis of new visual imagery (see [10, 16] for a survey). One of the most important benefits of using photo collections is the simplification of the image compositing process, such that non-expert users can make a new photograph with very little effort. However, creating complex scenes, such as urban landscapes, involves combining together several large-scale structures, such as roads, sidewalks or buildings, which typically exhibit strong perspective and significant variation in depth. When working in the 2D space of the image [5, 6, 8, 11], problems such as perspective distortion, incorrect occlusion or inconsistent relative size arise, making the image composition of novel content very difficult to deal with. Lalonde et al. [14] proposed that image manipulation be done in the 3D space of the scene in order to automatically scale objects depending on the depth, thus guaranteeing scale consistency in the scene. However, the objects that they allow to be inserted (small elements of an urban scene such as pedestrians, cars, fire hydrants, bicycles, vans, etc.) are still 2D planar impostors parallel to the image plane. But manipulating large-scale structures (e.g. buildings) in the way would produce incorrect occlusion and incorrect perspective.

In this paper we propose a new approach to compose urban pictures from photo collections (BUILDUP), working in the 3D space of the scene, where the user can start a new composition from scratch, and the system allows the user to modify the camera. We have created a 3D object library containing three large-scale object classes: roads, sidewalks and buildings; and we have developed an interactive application in order to build up an urban scene and to evaluate the feasibility of this new approach. The system pipeline is shown

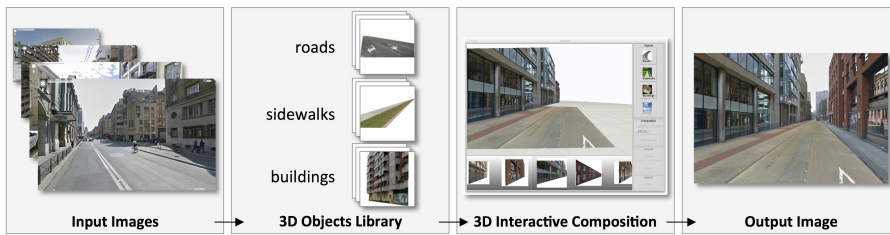


Fig. 2 The system pipeline. We first create a 3D objects library from a big collection of photographs. This library only includes objects of three classes: roads, sidewalks and buildings. Then, a non-expert user selects objects to compose the urban picture in the 3D space of the scene using an interactive application.

in Figure 2. To create the 3D object library we use Google Street View to obtain images, the LabelMe project [18] to annotate objects, and the LabelMe3D toolbox [17] to compute 3D meshes and textures from images and object annotations. In our system, the user can start a new composition from an empty canvas. During the composition, when the user picks the 3D location of a new object, the system queries the library to retrieve the best object instances depending on properties such as location, orientation and lighting. Finally, as the composition is in the 3D space of the scene, the system allows the user to modify the camera, that is, position, orientation and zoom, so that limited viewpoint adjustments could be performed by the user. Thanks to the use of real-world three-dimensional coordinates, the problems of inconsistent scale and perspective do not arise, and occlusions are well resolved.

The rest of the paper is organized as follows: first, we review the work related to our system (Sec 2), next we describe how we create our object library to be used for creating urban compositions (Sec 3), then we present the details of our 3D compositing system (Sec 4), followed by some results (Sec 5), limitations (Sec 6), and finally we present our conclusions (Sec 7).

2 Related work

The use of photo collections for the creation of novel digital pictures has gained popularity in the last decade. In [6], authors propose the idea of creating new visual content starting with an existing photograph and modifying it using a database of images which contain segmented and annotated regions with semantic labels. Given a photograph, the user defines the area to be replaced, and the system provides the most suitable image regions from the database to replace it. After the user selects the desired replacement region, texture synthesis [7] is employed to fill in the defined area in the input image with the region as the source texture. The central limitation of this work was that it could only fill in texture-like regions, such as sky, trees, leaves, etc. Scene Completion [9] addressed this limitation by using a much bigger image library (millions of photographs) which allowed filling in entire regions directly (using

graph cut [13] and Poisson blending [15]), avoiding the use of texture synthesis. Other notable works also create novel imagery by starting with a single image and collaging together pieces of other photographs, both interactively, such as Interactive Digital Photomontage [1], as well as automatically, such as cg2real [12].

Johnson et al. [11] proposed one of the earliest systems that allowed a user to start the composition process from an empty canvas. The user would place keywords in different places on the canvas and the system would issue queries into the large image library, which has been automatically annotated with semantic labels. After retrieving relevant images, they are automatically stitched together using graph cut optimization to generate a single photograph. Following on this, the Sketch2Photo [5] system allowed users to combine text labels with sketches starting from an empty canvas. The system searches the Internet for a background and the elements matching the items in the canvas, which are pasted on the landscape using the usual image blending techniques. As a result of this process, the user obtains several compositions generated automatically arranged by quality. However, Sketch2Photo is not an interactive system as the overall processing time can be up to several hours. Unlike Sketch2Photo, the Photosketcher composing system [8], which is sketch-based, can run at interactive speeds. The user begins by sketching the desired objects, and the system retrieved the best image matches from a library. The user then chooses the best match for each required object, extracts a region from the selected image, and pastes the object in the composition. The user can iterate the full process to refine the composition.

One common problem with all the above approaches is that the synthesis process is done in the 2D image plane, without any concern for the underlying geometry of the 3D world being depicted. As a result, gross mismatches in perspective and scale [5,9], as well as occlusion failures are often evident in the resulting compositions [14] (see Figure 3). The Photo Clip Art work [14] was proposed to address at least some of these concerns by performing perspective-aware object insertion. Given an image, the user first indicates the horizon and the camera height. The system then allows interactive insertion of objects (manually segmented 2D impostors from a large object library) onto the ground plane while placing them at the appropriate scale depending on depth, yielding a scale-consistent composite. While Photo Clip Art is a step in the right direction, it suffers from several problems that our proposed system aims to address. First, it cannot be used for creating content "from scratch", only for inserting objects into existing scenes. Second, and most importantly, it is not a true 3D system – while the objects are inserted in a perspective-aware way, all manipulations happen in the 2D image plane. The objects are represented as frontoparallel (with respect to the image plane) planar impostors, meaning that no out-of-plane rotations are possible. This is problematic for large-scale structures such as building facades, sidewalks, etc. Finally, camera viewpoint is fixed – not even small camera rotations or translations are possible. In comparison, our BUILDUP system is fully 3D. While the objects are still represented as 2D planar impostors, they are not forced to be frontopar-

allel but can be rotated in 3D (also allowing the camera viewpoint to change). The only limitation is if the object represented by the impostor is not really planar, large-scale rotations will make it look unnatural. Moreover, our system allows the user to start with an empty canvas, building up their planar 3D scene representation interactively, in real time (see accompanying videos).

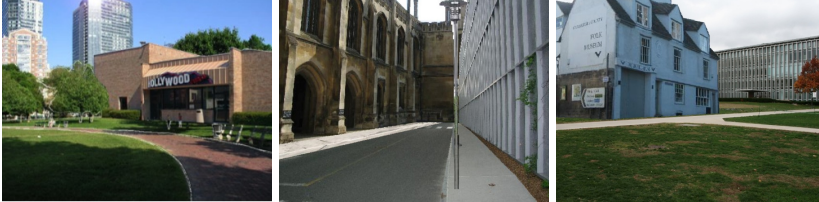


Fig. 3 Examples of gross mismatches in perspective and scale, as well as occlusion failures.

3 Creating the library

This project requires a large library of urban visual elements to accomplish our main goal – letting users create wide variety of high-quality urban pictures easily. Therefore, we first need an image collection to supply enough urban components. While an urban scene is composed of many different types of big and small objects, the process of obtaining and inserting the small object (e.g. cars, pedestrians, etc.) has been successfully addressed in [14]. Therefore, here we focus our attention on the big objects, the building blocks of an urban scene: roads, sidewalks and buildings. Second, we need to recover 3D object coordinates to make the composition in the 3D space of the scene. Last, we propose how to increase the usefulness and versatility of the computed library without growing its size.

In this work, we use the LabelMe database [18] and the Google Street View data as the sources of images for our visual library. The LabelMe project provides a high quality database that contains images with manually-segmented objects for many different scene types and object classes. Furthermore, the database grows over time since the LabelMe project lets users upload new images and to annotate objects using an on-line web tool. Google Street View is a technology that provides panoramic views from positions along many streets all over the world, and consequently the image contents are mostly constrained to urban scenes, but without any segmentation or annotation.

To calibrate the images and recover 3D object positions/orientations, we use the algorithm from LabelMe3D work [17]. Given an image with segmented objects, this approach returns a 3D mesh together with its image texture for each annotated object. Figure 4 shows an example of the inferred geometry where the objects are reconstructed as flat polygons in 3D.

Having a large-scale visual library is very important to be able to compose a wide variety of urban scenes. However, we have to overcome two practical

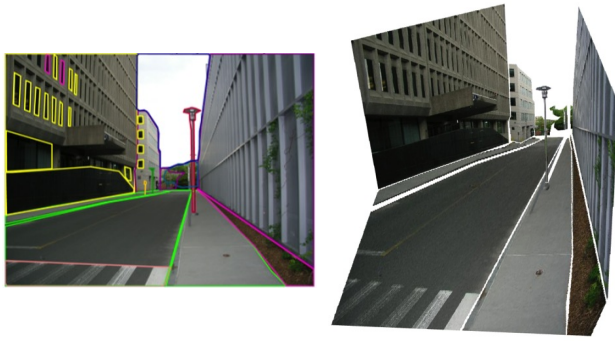


Fig. 4 Obtaining 3D data from an image. The left image shows the objects labelled with the open annotation web tool of the LabelMe project; the right image shows the 3D scene obtained using the LabelMe3D toolbox.



Fig. 5 Small objects appear in front of the big ones producing holey objects useless for our purposes.

difficulties. First, in many photographs, large-scale objects such as buildings, are often occluded by small objects like pedestrians, cars, bicycles, vans, buses, etc., producing holes, which are often difficult to fill in (see Figure 5). Second, a single image will only provide a few big objects: one road, a couple of sidewalks, and one or two buildings. Therefore, we propose two ways to increase the usefulness and versatility of the precomputed objects in our library: 1) exploiting the symmetry in urban scenes, and 2) letting users extend roads and sidewalks to make them useful in many more situations.

3.1 The LabelMe database

The LabelMe dataset contains images provided by the authors as well as other Internet users, taken with a wide range of cameras, covering many themes, including not only urban scenes, but indoor scenes, natural landscapes, etc. We started by downloading a dataset composed of 76,222 images with labelled outdoor objects. We then used the LabelMe3D algorithm to obtain 3D objects, which was successful for only 7% of the data, i.e. 5,365 images (this was mainly due to the annotations not being done in a way that LabelMe3D algorithm expects them to be). Specifically, we got 3D data for 10,865 objects but, unfortunately, most of these objects were not clean, containing holes from the

extracted foreground objects. Filtering the results by hand, we got down to about 100 images, which contained 3D data for 46 buildings, 64 roads, and 32 sidewalks. As we felt this was not enough data, we turned our attention to the larger Google Street View data.

3.2 Google Street View Images

LabelMe3D requires that the source image contains labelled objects with known height distributions, which is necessary to infer camera parameters, mainly focal length, camera pitch and camera height. These parameters are usually unknown for typical consumer photographs. The main advantage of using Google Street View images is that Google cars use the same fixed-height camera at least for a given city. This means we only need one image per city containing enough small known objects to successfully infer focal length and camera height. Unfortunately, camera pitch may be different for each image (since the car bounces on the road). Therefore, we had to manually annotate the images with a horizon.

We scraped 182 images from several cities in Google Street View, uploading these images to the LabelMe database, and labelling as many small objects as possible in one source image per city, and only the roads, sidewalks, buildings and the horizon line for the rest of the images. Then, we processed images with annotations using the LabelMe3D toolbox, inferring the focal length and camera height from the image where the small objects were labelled, and used them to process the rest of the images. Unlike with the LabelMe data, we were able to obtain 3D object information for 98% of the labelled objects, or 475 objects. Although it is not a huge number of objects for a real-world application, we found it was enough to carry out experiments to test our system. Figure 6 shows several examples of retrieved objects.



Fig. 6 Examples of retrieved objects using Google Street View as source of images and the LabelMe3D toolbox to get back 3D data.

3.3 The LabelMe3D toolbox

LabelMe3D toolbox [17] requires as input the source image and its xml file including the 2D outlines in image coordinates for each annotated object. As output, the toolbox extends the xml file adding per object: the 3D outline in eye coordinates, the 3D mesh, texture coordinates, and image texture.

For our application, we had to extend the toolbox as follows. First, in order to order to simplify the later scene composition process we enforce all the 2D outlines to be in counterclockwise (since the open annotation web tool of the LabelMe project just asks for a sequence of 2D points surrounding the object to be labelled, users enclose objects in arbitrary order sometimes clicking points in clockwise order, sometimes in counterclockwise order). Second, we adjust the 3D outlines for roads and sidewalks. It can be observed that roads and sidewalks (see Figure 6) have long parallel sides. However, the obtained 3D coordinates do not provide perfect parallel edges because the user imprecisely selects pixels on the screen when labelling an object. We adjust these labels to enforce parallelism, to facilitate object lengthening explained in next section. The adjustment consists of finding the furthest edge and its adjacent edges. The furthest edge is usually a single line located at the far end, away from the camera. Then, we check parallelism between the previous edge and the next edge. In order to satisfy the parallelism condition, we first compute an average vector using vectors from both edges. Then, we use this average vector as the new vector for both lines, recalculating the points shared with the furthest edge. Also, we assure that the furthest edge is perpendicular to both new adjacent edges just computing the nearest point from a given point to a line segment. We found this adjustment is enough for most of the annotated roads and sidewalks.

3.4 Improving the object library

3.4.1 Symmetry

One simple and easy way to provide the user with more objects is to apply bilateral symmetry. The origin of the world coordinates in the processed images lies at the intersection of the ground plane and the line perpendicular to the ground plane that passes through the camera centre. The X and Z axis lie in the ground plane, the Y axis is oriented towards the direction of the camera centre, and camera is pointing towards $-Z$. So, we simply mirror the image with respect to the $X = 0$ plane. The number of instances of roads and sidewalks are immediately duplicated, while buildings are more tricky especially when facades show written signs. E.g. in the Figure 13(e), we can see the building located on the right of the composition is really the mirror copy of the original but luckily does not have any signs (Figure 15(a) shows the same picture in a bigger size). We decided to provide the user with symmetric buildings and let the user make a decision.

Google Street View can provide as many images as we require. However, it is difficult to find large and clean objects in images, i.e., what we easily find are small pieces of big objects, which are not very suitable for making good compositions. Although this problem arises with roads, sidewalks and buildings, it is in the first two types where it is more pressing. Therefore, we propose to give the users the option of extending some objects. During the composition, users could, for instance, pick at the furthest edge of a sidewalk, and drag to change its length according to the length of other objects previously inserted in the composition (see videos).

We propose to apply this ability to roads and sidewalks in the following manner. When the user picks and drags the furthest edge of an object, we extend the corresponding vertices using the adjacent edges as guides. Then, the extended part is filled by repeating the texture successively in a symmetric way with respect to the picked edge. This process is outlined in Figure 7, and four results are shown in Figure 8.

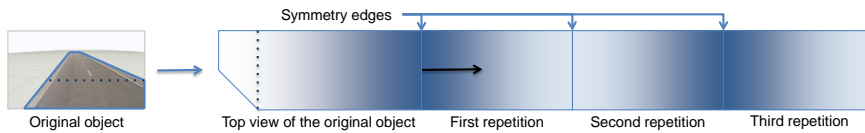


Fig. 7 The user picks and drags the furthest edge to extend the road. The texture is repeated in a symmetric way respect that edge.

3.4.2 Lengthening

In order to guarantee the lengthening is done correctly, edges adjacent to the picked one must be parallel. Otherwise, adjacent edges could cross or separate

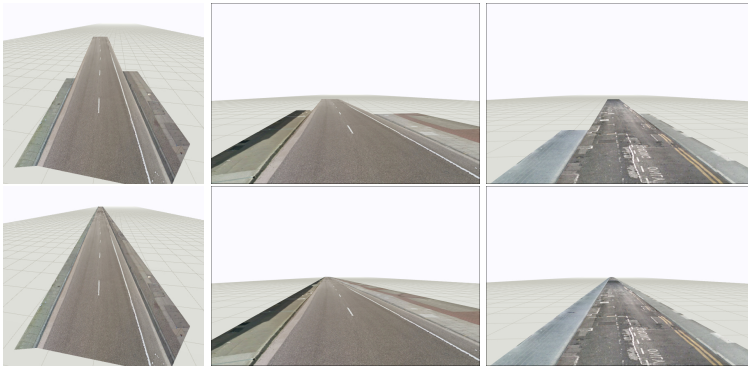


Fig. 8 Examples of extended sidewalks and roads. Images on the top show the original length of objects, and images on the bottom show the same objects made longer. In the left example, the viewpoint has been elevated to display a better view of the extended objects.

excessively during the extension operation. Roads and sidewalks usually fall into this category in urban scenes. The way we propose to fill the extended object also requires that the picked edge is perpendicular to the adjacent ones. All these conditions have been fulfilled using the extensions to the LabelMe3D toolbox proposed in the previous section. Note that this approach will not work for buildings, since their texture is much more intricate for this simple fill in. While we experimented with various texture synthesis approaches, none worked robustly enough to be used in practice.

4 Scene composition

Figure 9 shows a screen-shot of our user interface. The right column of icons shows the object classes available in the library, and some options for camera control and input and output operations. The central panel displays the composition where initially a virtual 3D ground plane is shown in order to make the user aware that the composition will be performed in the 3D world. The lower side panel displays the object instances while the object in the centre of the panel is shown in the composition. This panel is a draggable area to let the users examine the whole selected object class.

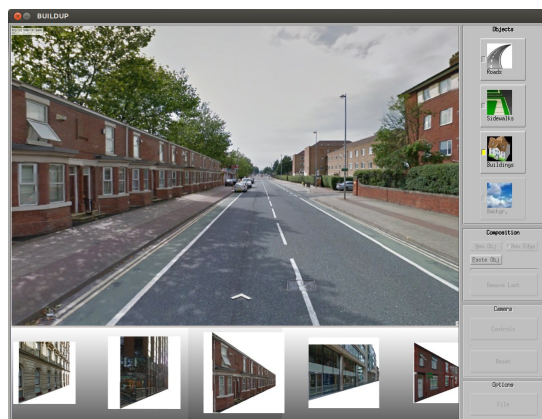


Fig. 9 User interface. The central panel shows the composition. The upper right menu allows users to navigate between different object classes. The lower panel shows the top matched objects that can be inserted into the scene. This panel is a draggable area to let the users to reach more objects, and the object in the centre of the panel is displayed in the composition.

4.1 Starting the composition

To start a new composition from scratch, the user can proceed in two different ways:

1. The user chooses an object among those shown in the lower panel. In this case, the object is inserted onto the canvas at exactly the same position and orientation as it is in the image where it comes from. And the user cannot move it at least for now.
2. The user first draws an edge on the ground plane to restrict where to insert the new object. Then, the system orders the object library to provide the user with those that best fit for that location, and the result is provided through the lower panel. The user drags this panel and the system automatically transforms and displays the objects. In this case, the user can also drag in the central panel to displace an object along the edge.

Sometimes, the user could wish to start by first inserting a photograph in order to set up the lighting scene conditions or just to experiment with a real place. In either case, a new object could be added using either of the two previous methods, although the edge drawing is the easiest one, as the user can take advantage of the vanishing lines that already exist in the photograph. Then, we guide the user to continue the composition in a quick and easy way.

4.2 Guiding the composition

We allow the user to add a new object by selecting an edge previously placed on the ground plane. Since the user can place an edge wherever s/he wishes, this allows for inserting objects anywhere on the canvas. However, this is not very helpful since, typically, the user wants to add new objects along the previously inserted ones in the scene. Therefore, we can restrict the user to only choose one of the contour edges of the objects that are already present in the scene. This behaviour is similar to those [9, 11] where image regions are stitched together to generate the new photograph, instead of using a paste based method as proposed in [5, 14].

After one contour edge is selected, the system guesses the best-suited object class. If the selected edge belongs to a road, most probably the user wants to add a sidewalk. Likewise, when the user selects an edge belonging to the base of a building. However, if the user selects an edge from a sidewalk, both a building and a road could fit well. In this case, the decision depends on the sign of the increment of the X screen coordinate between the two edge endpoints. As vertices are in counterclockwise order, if the increment is positive we choose the road class, and the building class otherwise. Figure 10 shows an example where for the edges on the right and left sides of the sidewalk, the system selects the road class and the building class respectively. Of course, sometimes the user wants to place an object of a different class than what was guessed by our system. In that case, the user can change the class using the buttons on the right panel of the interface.

Best objects for a selected edge are shown in the bottom panel. The object in the middle of this panel is automatically transformed and displayed on the composition. The user can drag on this panel to try another object making the displayed object gradually disappears from the composition while the new



Fig. 10 Guessing the best-suited object class for a sidewalk. The user selects the green edge and the system, depending on the sign of the increment of the X screen coordinate between the two edge endpoints (assuming vertices in counter-clockwise order), automatically selects the road class (left) or the building class (right).

one gets more visible. In PCA [14], the user has to pick the objects, one each time, to see how they suit. Our proposal makes faster and more comfortable the user to operate and test different options before taking a decision.

During the composition, the drag action on the central panel allows the user to displace an object along the edge used when it was inserted in the scene. In the same way, the user can change the length of a road or a sidewalk as it was explained in section 3.4.2. In both cases, the system highlights objects and edges to provide the user with more feedback.

Finally, as the composition is in the 3D space of the scene, the system allows to change the virtual camera parameters: viewpoint, view direction and zoom. Figure 11 shows two views of the same scene. In this example, camera has undergone an exaggerated displacement for illustration purposes. Typically, the user will only adjust the camera slightly, so that planar impostors representing non-planar objects (such as facades with balconies) continue to look realistic.

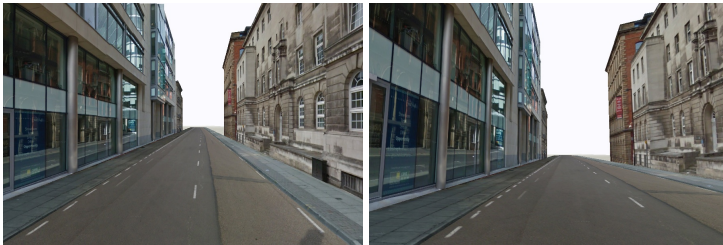


Fig. 11 Two different views of the same scene. Camera parameters have been exaggeratedly modified to make more noticeable the difference.

4.3 Object ordering

Our system can provide access to several thousands of objects in the library. This could be overwhelming for the user, without a good object selection

mechanism, that can offer the "best fitting" objects for any given situation in an intuitive way. Moreover, this process should make sure a user can find enough satisfactory objects among the top 10-20 displayed objects in order to speed up the composition process. Every time the user chooses where to insert a new object the system will offer the best choices to continue with the composition by automatically reordering the entire object library to fit the particular insertion task. Inspired by Photo Clip Art [14], we propose to order the objects using a matching criterion that involves two main aspects: compatible illumination (I) and appropriate object location (L). This criterion is defined as a linear combination of both aspects, $\alpha I + (1 - \alpha)L$, where $0 \leq \alpha \leq 1$.

The illumination constraint allows us to mix objects coming from different images provided that they were captured under similar lighting conditions. To compute illumination consistency, we simply use histograms for sky and ground as proposed in Photo Clip Art. At run time, Photo Clip Art computes both histograms for the photograph inserted in the background. And then, best-suited objects are those with histograms very similar to those of the background (object histograms are sky and ground histograms of its respective source image). In BUILDUP, as we can start from scratch, the histograms of reference are set up once the first object is inserted into the composition.

The appropriate object location is used to guarantee that texture quality is preserved. This means if the object is actually non-planar, it should only be inserted into the composition at a similar position and orientation as in its corresponding source image in order to ensure the visual quality of the object. That is, an object would fit well into the new scene if both orientation and position are close to that in the source image, and poorly otherwise. In order to compute it, we just measure the translation and rotation needed to position an object along the picked edge on the composition. And we do that for each one of its edges. As a result, we store in the object data structure which edge is the cheapest and the required rotation and translation values.

Figure 12 shows an example where the user wants to add a building on the right side of a sidewalk and the system, after the ordering, shows the best buildings on the lower panel.

5 Results and Evaluation

We used Matlab to construct the 3D objects library, while the composing tool has been developed in C++, together with the FLTK library for the user interface, and the OpenGL graphics standard library to display the 3D composition and objects in the lower panel.

Figure 13 shows an example of a composition started from scratch. The first row shows the sequence of object insertion, and source images are shown in the second row. Every arrow connects one object in both rows: the original location in the source image and the new location in the composition. Notice that every object comes from a different source image, while remaining in very



Fig. 12 When the user wants to add a building on the right side of the sidewalk, the matching criteria helps offering the buildings (shown in the lower panel) best suited for that place.

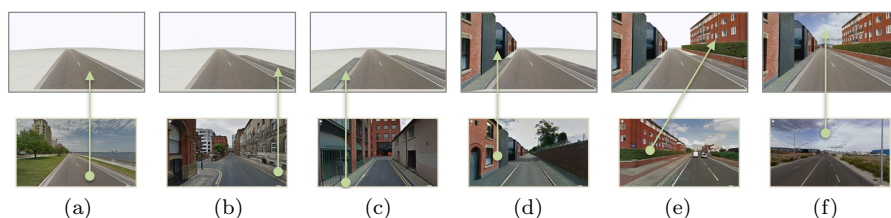


Fig. 13 Sequence of object insertion. The top row shows the sequence of inserted objects, and source images are shown in the bottom row.

similar positions and orientations. The use of the symmetry property is shown in fifth step of this sequence. In the last one, road and sidewalks have been extended, and a background has been added mainly to provide a sky.

Figure 15 shows four more results, produced from scratch in a few minutes by users who were not artistically skilled (see accompanying videos). We can observe the problems of inconsistent scale and perspective do not arise, and occlusions are correctly resolved.

Figure 16 shows five results where the user started from a photograph and then inserted a few objects. In these examples, the goal is to change only some part of the original photograph, and the background remains mostly visible. After providing the background, the user drew one or several edges coincident with vanishing lines present on the original photographs, and then added the new objects in few seconds (see accompanying videos). Top row of Figure 16 shows the background images, and bottom row shows the results after inserting one or several objects in the composition.

We ran a user study with 35 participants, on a set of 10 images, where five were real (R), and five were fake (F), e.g. synthesized by our system. The

participants were mainly undergraduate students of computer science, design and development of videogames, and audio-visual communication, and a few master students on intelligent systems. Table 1 shows the average scores. These results are also shown in the chart of Figure 14 in a more comprehensive way. The subject is asked to rate four aspects on a 5-point Likert scale from 1 to 5: very poor, poor, barely acceptable, good, very good; based on the following questions: A) visual realism, B) beauty, C) useful for a video game; D) useful in urban applications. The column E) shows the percentage of subjects that agree this sentence: "yes, it is a real image". We did not limit the viewing time. Some synthetic (fake) images are shown in Figure 15 while two of the real images used are shown in Figure 18.

Type	A	B	C	D	E (%)
<i>F</i>	4.17	2.86	4.14	4.26	82.86
<i>F</i>	3.60	2.91	3.54	4.26	60.00
<i>F</i>	4.20	3.74	3.71	4.31	80.00
<i>F</i>	3.46	2.89	3.57	3.94	60.00
<i>F</i>	3.94	3.51	3.77	3.91	62.86
<i>R</i>	4.91	2.97	3.54	4.17	100.00
<i>R</i>	4.34	2.94	3.91	4.23	94.29
<i>R</i>	3.49	3.57	3.86	3.94	57.14
<i>R</i>	3.63	3.29	3.83	3.86	60.00
<i>R</i>	4.66	3.29	3.60	4.03	94.29

Table 1 User study average scores for five fake images (F) and five real images (R) on a 5-point Likert scale (from 1 to 5: very poor, poor, barely acceptable, good, very good): A) visual realism; B) beauty; C) useful for a video game; D) useful in urban applications. Column E) shows the percentage of subjects that agree this sentence: "yes, it is a real image".

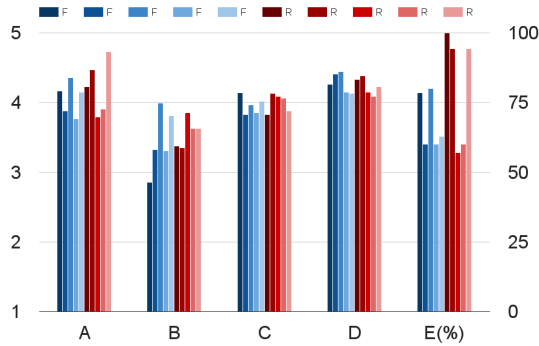


Fig. 14 This chart shows the user study average scores shown in Table 1. For each question, the obtained scores are shown in blue for fake images, while red is used for real images.

Figure 14 shows that our compositions (blue) are perceived to be almost as good as real images (red). On visual realism, the subjects scored real im-

ages only 8.6% higher than the fake images. The differences in beauty and usefulness scores were negligible. Overall, 69% of the participants thought our compositions were real compared to 81% got the true real images (i.e. 19% of the time real images were thought to be fake, demonstrating the same type of trigger-happiness also observed in [9]).

We also measured the usability of the BUILDUP system using the System Usability Scale (SUS) [4]. We invited to 10 people to use our system for creating full compositions starting from scratch, similar to the examples presented in this section. Subjects were observed during the creation of the scene and were recommended not to ask for help. The users were not restricted by time or number of compositions. At the end of the usability study, the subjects filled a standard SUS questionnaire. According to [2], scores below 50 are not acceptable, scores from 50 to 63 are low marginal, scores from 64 to 69 are high marginal and scores above 70 are acceptable. Table 2 summarizes the average score and standard deviation per question. SUS final score shows that our system has an average score of 87.8 and standard deviation of 17.8. This result shows that BUILDUP is quite easily accepted by the users.

Question	1	2	3	4	5	6	7	8	9	10
μ	3.00	4.00	3.80	2.90	3.30	3.70	3.40	3.90	3.30	3.80
σ	0.94	0.00	0.42	1.37	1.06	0.67	0.97	0.32	0.95	0.42

Table 2 Individual SUS average score and standard deviation for BUILDUP.

Figure 17 shows three compositions made by different graphic designers using the well known image editing software Adobe Photoshop. The compositions show pretty good urban scenes similar to those obtained using BUILDUP. However, every composition took several hours (between two and four hours) while compositions done using BUILDUP usually took a few minutes and were made by non-expert users.

6 Limitations

Although most of the urban scenes probably depict straight streets, there are urban scenarios that show curved streets, traffic circles or non-square plazas. Right now, our system works only for straight edges and therefore our results show straight streets. However, we think this limitation could be easily addressed, e.g. we could extend a sidewalk or a road following the spline curve when the floor outline of a building impose it. Another limitation is that compositions made with our system may show some inconsistencies mainly due to the presence of shadows and reflections. For example, when an inserted object contains cast shadows and no other object in the scene causes them. Or when a composition includes a facade with large windows, which reflect objects absent in the scene. Figure 19 shows four examples of these limitations.

7 Conclusions

We have presented an approach for interactive authoring of photorealistic urban scenes by using a large image library. The key novel aspect of this work is that the authoring is done in the 3D space of the scene, instead of the 2D image space, allowing for authoring of large-scale 3D consistent scene structures, something that was not possible before. Currently, our method for building the image library requires large amount of manual interaction, in particular, segmentation of the objects and defining of the horizon for each image. As computer vision techniques improve, this part will be automated, allowing for much larger libraries, which will, in part, further improve the visual quality of the resulting composites.

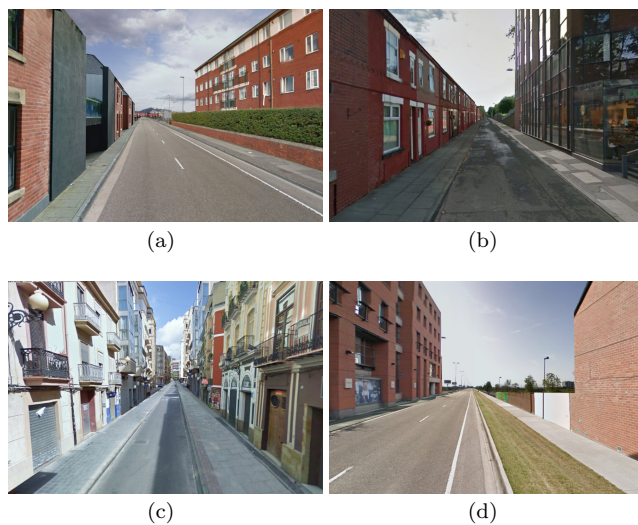


Fig. 15 Four examples obtained using BUILDUP where the user started from scratch and the goal was to create an entire composition.

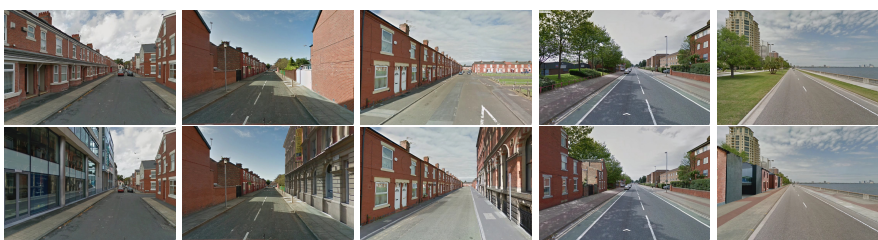


Fig. 16 Compositions (bottom row) where the user started from a photograph (top row). In these examples, the user inserted a few objects and the background image mostly remains visible after the composition.

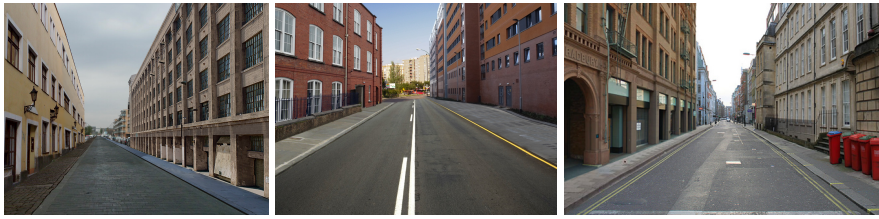


Fig. 17 Compositions done with Photoshop by three artistically skilled users.



Fig. 18 Two real images used in the user study. Left image was qualified as real by 60% of subjects; right image 94.29%.



Fig. 19 Examples of inconsistencies of our system. The two compositions on the left show cast shadows on the floor and no object causing them. The two compositions on the right show object reflections on the large windows while we cannot see those objects in the scene.

References

1. Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D., Cohen, M.: Interactive digital photomontage. *ACM Trans. Graph.* **23**(3), 294–302 (2004)
2. Bangor, A., Kortum, P.T., Miller, J.T.: An empirical evaluation of the system usability scale. *International Journal of Human-Computer Interaction* **24**(6), 574–594 (2008)
3. Branit, B.: World builder (2007). URL <https://www.youtube.com/watch?v=QP3YywgRx5A>
4. Brooke, J.: "SUS-A quick and dirty usability scale." *Usability evaluation in industry*. CRC Press (1996)
5. Chen, T., Cheng, M.M., Tan, P., Shamir, A., Hu, S.M.: Sketch2photo: internet image montage. In: *ACM SIGGRAPH Asia 2009 papers*, vol. 28, pp. 124:1–124:10. ACM, New York, NY, USA (2009)

6. Diakopoulos, N., Essa, I., Jain, R.: Content based image synthesis. In: Proc. of International Conference on Image and Video Retrieval, pp. 299–307 (2004)
7. Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, SIGGRAPH '01, pp. 341–346. ACM, New York, NY, USA (2001)
8. Eitz, M., Richter, R., Hildebrand, K., Bubeck, T., Alexa, M.: Photosketcher: interactive sketch-based image synthesis. IEEE Computer Graphics and Applications (2011)
9. Hays, J., Efros, A.A.: Scene completion using millions of photographs. ACM Transactions on Graphics (SIGGRAPH) **26**(3) (2007)
10. Hu, S.M., Chen, T., Xu, K., Cheng, M.M., Martin, R.R.: Internet visual media processing: A survey with graphics and vision applications. The Visual Computer **29**(5), 393–405 (2013)
11. Johnson, M., Brostow, G.J., Shotton, J., Arandjelović, O., Kwatra, V., Cipolla, R.: Semantic photo synthesis. Computer Graphics Forum **25**(3), 407–413 (2006)
12. Johnson, M.K., Dale, K., Avidan, S., Pfister, H., Freeman, W.T., Matusik, W.: Cg2real: Improving the realism of computer generated images using a large collection of photographs. IEEE Transactions on Visualization and Computer Graphics **17**(9), 1273–1285 (2011)
13. Kwatra, V., Schödl, A., Essa, I., Turk, G., Bobick, A.: Graphcut textures: image and video synthesis using graph cuts. In: ACM SIGGRAPH 2003 Papers, vol. 22, pp. 277–286. ACM, New York, NY, USA (2003)
14. Lalonde, J.F., Hoiem, D., Efros, A.A., Rother, C., Winn, J., Criminisi, A.: Photo clip art. In: ACM SIGGRAPH 2007 papers, vol. 26. ACM, New York, NY, USA (2007)
15. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. ACM Trans. Graph. **22**(3), 313–318 (2003)
16. Reinhard, E., Efros, A.A., Kautz, J., Seidel, H.P.: On visual realism of synthesized imagery. Proceedings of the IEEE **101**(9), 1998–2007 (2013)
17. Russell, B.C., Torralba, A.: Building a database of 3d scenes from user annotations. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2711–2718 (2009)
18. Russell, B.C., Torralba, A., Murphy, K.P., Freeman, W.T.: Labelme: a database and web-based tool for image annotation. International Journal of Computer Vision **77**(1-3), 157–173 (2008)