A Multi-Lens Stereoscopic Synthetic Video Dataset

Fan Zhang, Wu-chi Feng, Feng Liu

Intel Systems and Networking Lab
Portland State University
Portland, Oregon, USA
{zhangfan, wuchi, fliu}@cs.pdx.edu

ABSTRACT

This paper describes a synthetically-generated, multi-lens stereoscopic video dataset and associated 3D models. Creating a multi-lens video stream requires small inter-lens spacing. While such cameras can be built out of off-the-shelf parts, they are not "professional" enough to allow for necessary requirements such as zoom-lens control or synchronization between cameras. Other dedicated devices exist but do not have sufficient resolution per image. This dataset provides 20 synthetic models, each with an associated multi-lens walkthrough, and the uncompressed video from its generation. This dataset can be used for multi-view compression, multi-view streaming, view-interpolation, or other computer graphics related research.

General Terms

Measurement, Documentation, Performance

Keywords

Stereoscopic video, multi-view compression

1. INTRODUCTION

As the number of megapixels continues to grow in imaging hardware, the use of such higher resolution can be applied to enable a host of alternative applications. These applications will allow for creating a better user experience, rather than just adding more pixels. Fields like stereoscopic imaging, light-field cameras, camera arrays and the like promise to allow users to have better experiences with their imaging data.

For multi-lens camera systems, where lenses are placed very near to each other allow for a number of scenarios. Linear arrays of lenses can be used for stereoscopic imaging applications where the additional lenses can allow for better disparity management of the viewing scenario (environment), allowing for a more pleasant 3D user experience. Camera arrays such as the Point Gray ProFusion 25 capture a two-dimensional array of synchronized video streams which can allow for better video stabilization [8]. Two such examples are shown in Figure 1.

ACM MMSys '15, March 18-20, 2015, Portland, OR, USA

http://dx.doi.org/10.1145/2713168.2713199

Figure 1: Multi-lens Camera Systems

There are several limitations to today's multi-lens video capture hardware. In order to tailor stereoscopic videos to a variety of display scenarios, various small inter-lens distances are required. For example, the standard inter-ocular distance is 2.5". If we want to arrange multiple lenses into a linear array in order to obtain multiple inter-ocular distances simultaneously with a distance difference like 0.5", we will require a lens spacing of less than 0.5". Unfortunately, very few cameras meet such a requirement. Point-of-view (POV) cameras are a possibility but have very limited ability to synchronize their streams. Furthermore, such POV cameras, like the ones on the left side of Figure 1, are usually very wide-angle with radial distortion. Other cameras like the 5x5 lens ProFusion 25, while having the hardware array, have images of only VGA quality. This impacts the ability to study systems issues such as compression and adaptation.

To begin the multimedia systems work before the hardware is readily available, we have created a synthetically-generated, multi-lens video dataset that can be used to explore research such as multi-lens video compression algorithms for streaming, the effect of non-linearly placed synchronized video lenses, and other multi-lens systems problems such as view interpolation. This dataset paper describes the creation of the models and the rendered video frames for a multi-lens video system. This will allow researchers to study problems brought about by real-world multi-lens systems in isolation or in groups including: registration between lenses, synchronization, camera noise, and camera distortion. At a higher-layer, the dataset will support multi-view compression, multi-view streaming, or some computer graphics related research.

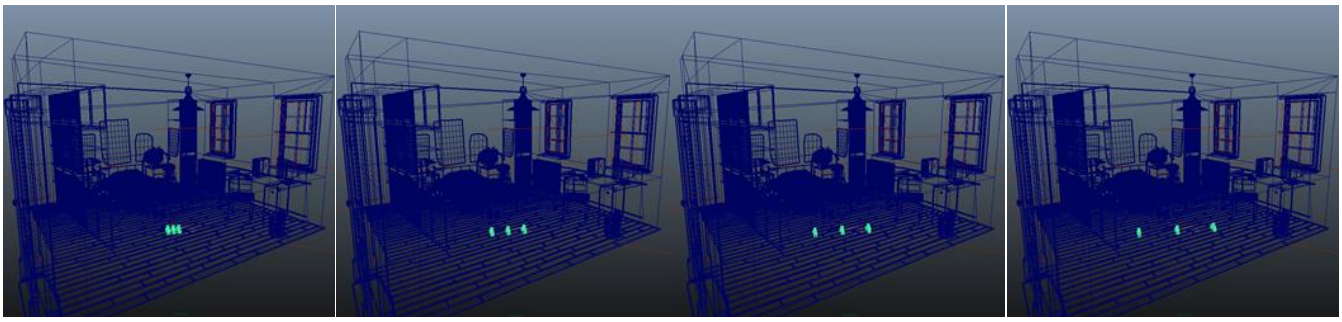
Contributions of our work: We have taken 20 publicly available 3D models and added textures and walkthroughs (panning and zooming shots) with multiple-lens. Researchers can then modify the models as well as the camera placement and movements as they require. Additionally, for each of the 20 scenes, we have generated 300 frames for each lens in an 8-camera array.

2. CREATING THE DATASET

Starting with a number of 3D modelled scenes, we created the dataset through a number of steps including: shading and

Figure 2: Base Maya Model Example

Figure 3: A Final Scene Example



Cameras 4/5 Cameras 3/6 Cameras 2/7 Cameras 1/8

Figure 4: Relationship and Placement of Cameras for 8-lens Stereoscopic Video

texturing; lighting; animation and camera control; and rendering. We will describe these steps in the remainder of this section.

2.1 Modeling

Creating models of realistic scenes is quite difficult and time consuming. To get the models for the scenes, we used the Autodesk Maya 2014 software. We leveraged other artists' Maya models to serve as the basis of our work so that we did not need to build the 3D scenes from scratch. The models can be downloaded from 3Drender [3]. The data files downloaded are 3D models either with no shaders, textures, or videography. An example of a single scene is shown in Figure 2. The Maya models can serve as the base scene for which realism needs to be added.

For the 20 models downloaded, we edited the source files to add shaders, textures, and lighting properties. We spent a significant amount of time providing the right textures to be mapped onto each of the 3D models within the scene. For example, the blanket has a more realistic texture to provide for more realism. For shading, Maya provides different shader types such as the Lambert shader and Phong shader. The Lambert shader is an evenly diffused shading type that creates dull or matte surfaces. The Phong shader allows surfaces to have specular highlights and reflectivity. Depending upon the reflexivity requirement of each object we assigned either a Lambert shader or Phong shader to each object. For example, the leaves on the plant use the Phong shader, while the blanket uses the Lambert shader. This creates the appropriate matteness or reflectivity for each object within the scene. To do advanced texturing, we can also set up UV mappings and assign advanced textures, such as a real wood floor material to the ground.

To create a sense of depth and the perception of color and materials, we need to set up CG lighting for the 3D scenes. Maya provides several light types including spot lights and point lights. Spot lights place light in specific areas. Point lights cast light from a single specific point in space. In addition to these light types, Maya also has the advanced mental ray lighting option which can simulate an open-air sunlight effect for the entire scene. This is the easiest way to create a nice-looking view, which we used for most of our scene rendering. An example of the scene from Figure 2 after we have added shaders, textures, and lighting is shown in Figure 3. Please refer to the Maya tutorial for a more detailed introduction [4].

2.2 Camera Placement and Movement

To make dynamic videos, we need to add object animation or camera movement to the static scenes. Each of our videos has 300 frames. Among the frames, we set several keyframes to establish a movement scheme for an object or the camera. Maya can then create the animation curves automatically between keyframes. For each scene, we created several keyframes to make the resultant video stream more interesting and to introduce some motion for any compression research that might use the dataset.

In order to create 8 view videos, we used the stereo camera rig provided by Maya to simulate the parallel multi-camera setting. Maya provides a stereo camera rig that contains three cameras: a left camera, a center camera, and a right camera. The center camera is used solely for positioning of the camera, while the left and right cameras are used for the actual generation of frames. To create 8 linearly-aligned lenses, we needed 4 synchronized streoescopic camera rigs for our videos. To align all cameras

Figure 5: An Example Multi-lens Camera Array: This figure shows an example of the 8 linear camera array views. The individual camera views are arranged from left to right.

parallel with each other with the same interaxial separation, the positions of the center cameras of all 4 stereo camera rigs are set to be exactly same. With 8 cameras (numbered 1 to 8 from left to right), if we assume cameras 4 and 5 have a baseline of x, then cameras 3 and 6 should have a baseline of 3x. Furthermore, cameras 2 and 7 have a baseline of 5x, and cameras 1 and 8 have a baseline of 7x. The baseline value can then be adjusted according to the required spacing of the cameras for a particular application. An example of this for the example scene is shown in Figure 4.

The last step is to render the 8 view videos for each scene. To accomplish this, we need 8 Maya binary files each containing one camera setting to render one camera view. Instead of rendering the videos to video files, we rendered each video frame into PNG format so that there are no pre-biased DCT artifacts in the video dataset. An example of a rendered scene is shown in Figure 5.

3. DESCRIPTION OF THE DATASET

The contributed dataset consists of twenty different scenes, each consisting of two primary components: additions to publicly available 3D Maya data models to make them look realistic (including camera videography) and a rendered set of multi-view imaging data. The scenes will provide a breadth of different scenes for use in experimental evaluation. In the remainder of this section, we describe the specific details of the datasets.

The Maya modeling data includes a number of components including, the 3D models for the actual scenes, the lighting and shader information, the camera rigs, and the animation path.

Maya data files contain MEL scriptings language code [1]. MEL is a scripting language similar to the style of PERL or TCL. This makes the format fairly easy to modify. The dataset contains the original files so that users can then try out different camera angles, camera placements, and other options. While data files can be either ASCII or binary, our files are in ASCII to maximize the extensibility and usability of the data. One ASCII Maya scene saved with all components mentioned above (shaders, textures, lighting, animation, cameras) is approximately 100 Mbytes. The entire Maya 3D dataset is approximately 2 GBytes in size.

For each of the scenes we have rendered an animation with a camera panning and zooming. For each video, we used the Maya software to generate 300 frames with resolution 1280x720 (i.e., 720p). While the dataset could have been generated with more frames and/or higher image quality (e.g., full HD @ 1920x1080 or 4K), this has a significant impact on the ability to store the dataset. The dataset including the 20 scenes, 8 linearly-aligned views, and 300 frames per camera array yields a dataset that is approximate 45 GBytes in size. Each of the frames is numbered by view and frame number, with each scene consisting of 2400 frames. We have compressed the data using PNG in order to minimize the artifacts in the dataset. Researchers that do not need to adjust the number of frames or the placement of cameras can just use the video frames in the set. The 20 scenes that are in the dataset are shown in Figure 8 at the end of this paper.

3.1 Dataset Discussion

Synthetically generated datasets are a bit different than their real-world counterparts for a number of reasons. In the remainder of this subsection, we briefly describe the impact of this choice.

First, using a synthetically generated set of video frames means that the frames are perfectly synchronized. For our stereoscopic video compression work, we require both a very small inter-lens spacing as well as accurate synchronization. Unfortunately, it is very difficult to build such hardware. Without fine synchronization, the disparity calculation (distance between the "same" points of two views in the camera array) and the motion vector calculation will be impacted. Using the synthesized dataset allows us to remove synchronization issues and to study the effects of synchronization in isolation if so required.

Second, the video frames are devoid of noise, have no lens distortion, and are perfectly aligned. Without noise or lens distortion, compression, computer vision algorithm, and motion compensation all become easier. We believe this makes it easier to understand the impact of using multiple video lenses in isolation. Real-world issues like lens distortion and noise can independently be added in through CCD noise and lens distortion can also be added [7][12].

Third, the video frames are not truly realistic. We have made the shaders, textures, and lighting as realistic as possible; however, they are still computer generated. We expect that even though they are computer generated that it will not negatively impact or bias compression performance.

For the 20 scenes generated, we have taken the 300 frames from the leftmost lens and compressed them into H.264 format to see how the synthetically generated videos compress. For this, we used the reference H.264/AVC reference encoder to compress the data [2][11]. We then compressed it at a number of quantization values and calculated the resulting PSNR values. The resulting rate distortion curves are shown in Figure 6. As shown in the figure, the rate distortion curves are similar to traditional rate distortion curves. In particular, there is a comparison paper that shows the differences in using different H.264 encoders. Our compression results are within the range of their published results [13]. As a result, we expect that the proposed dataset can be used for experimentation and have it at least be representative of results compared to real-world videos. The one outlier is the video scene that achieves 52 dB PSNR at 2.5 Mbps. This scene is the relatively simple scene with the plate of fruits sitting on a blanket.

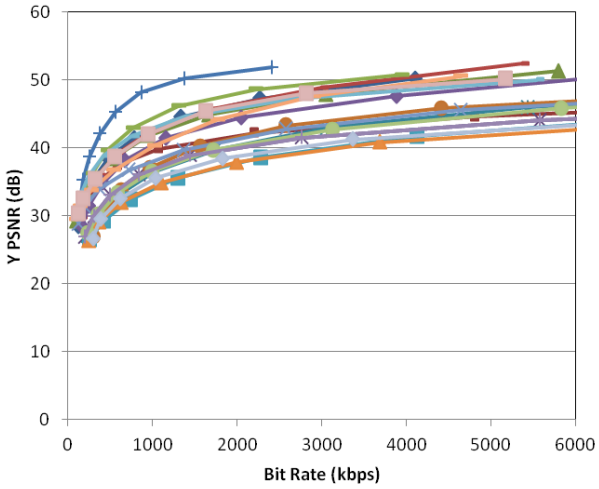


Figure 6: Rate Distortion Curves: This figure shows the rate distortion curves for the 20 videos compressed with the H.264 reference code. The Y channel PSNR value is shown.

4. APPLICATIONS OF THE DATASET

There are a number of uses of the dataset that we envision for the multimedia systems community. We will describe some of these in the remainder of this section.

4.1 Multi-view Compression and Streaming

Multi-view video coding is an important problem for the multimedia systems community. Multi-view video can come in a number of flavors. It can mean an array of linearly aligned cameras or a grouped set of cameras that are roughly pointed to the same area. In such cases, the ability to take advantage of inter-camera redundancy for use in compression is useful [10][14]. Multi-view compression, while great for compression efficiency, makes it not so amenable to streaming. If a particular view is required by the user, the entire multi-view set needs to be transmitted. Obviously, this can place a tremendous overhead to view just a single stream. Unfortunately, most multi-view codecs are set-up to maximize inter-lens compression. A standard example given in the MVC reference coder is shown in Figure 7.

This dataset will allow for the exploration of the inter-dependency between compression and streaming [5]. There are several such threads of research that could be pursued with this dataset. First, for stereoscopic multi-lens video, understanding the relationship between motion estimation and disparity calculation (the difference in pixels of a single point between the left and right eyes) is unknown. Motion estimation typically finds the best visual color (luminance) match, whereas disparity calculations for stereo video are object matching. Still, the underlying algorithms may benefit from each other to improve compression speed. Second, in the adaptation and streaming of multi-view video streams (either stereoscopic or not), understanding the overhead

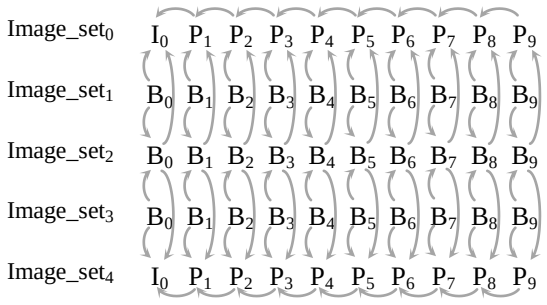


Figure 7: Typical Multi-view Frame Dependencies

of using inter-lens compression and adaptation is still not well understood. For example, should all lenses be kept separate for streaming purposes or are there efficient mechanisms to limit the amount of data needing to be transmitted when only a sub-set is required? Third, with the synthetic dataset, we can begin to explore how important camera placement is for the capture and compression of depth images (e.g. [6]). This requires a scene that is readily repeatable which means that synthetic is most like the best prospect for such research.

4.2 View Interpolation

This multi-view video dataset will provide a useful benchmark for view interpolation research. View interpolation is a classic topic in computer graphics and computer vision. It is useful for a wide variety of applications, such as multi-view video synthesis and editing, video stabilization, high frame-rate video synthesis, and virtual reality. View interpolation takes images taken at multiple viewpoints and synthesizes a new image from as if it were viewed by the camera from a new viewpoint. A large amount of view interpolation algorithms have been developed over the past decades [9]. However, view interpolation is still a challenging task and faces a few challenges. First, it is still difficult to "correctly" or visually plausibly handle occlusion and dis-occlusion especially in the region with significant depth discontinuity that some region visible from the new viewpoint is occluded from existing viewpoints. Second, non-Lambertian reflection and semi-transparent surfaces are difficult to render correctly. Third, while novel views can be interpolated sometimes in a visually plausible way, it is challenging to create a novel video as it requires interpolating frames in a temporally coherent way. This large multi-view video dataset will be useful to evaluate existing and forthcoming view interpolation methods, and identify new problems and research opportunities on this topic.

5. CONCLUSION

In this dataset paper, we have introduced a Maya 3D model data set that includes realistic shaders, textures, lighting, animation, and a linear array of cameras, creating a set of 20 synthetically generated video streams. We have also, for each model, generated a set of 300 frame x 8 camera video data set. This dataset can be used to study compression of multi-view videos and can also be used for computer vision work.

6. ACKNOWLEDGMENTS

This material is based upon work supported by NSF IIS-1321119, CNS-1205746 and CNS-1218589. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Credits for models in Figure 8 (left to right, top to bottom): Row 1: model - Giorgio Luciano, paintings - Craig Deakin; model - David Vacek, design - David Tousek; model – A. Kin Fun Chan, Dan Konieczka; Row 2: model - Giorgio Luciano; model - Jeremy Bim; model - Christophe Desse, Matthew Thain; Row 3: model - Dan Konieczka; model - Andrew Kin Fun Chan, Dan Konieczka, Senthil Kumar; model - David Vacek, design - David Tousek; Row 4: model - Dan Wade; model - Dan Konieczka, Juan Carlos Silva; model - Jeremy Birn; Row 5: model, texture, lighting - Jeremy Birn; model - Ted Channing; model - Juan Carlos Silva; Row 6: model, texture, lighting -Jeremy Birn, model - Christophe Desse, Matthew Thain; model - Jeremy Birn. Rig or Material "Squirrel" used with permission (© Animation Mentor 2014). No endorsement or sponsorship by Animation Mentor. Downloaded at www.animationmentor.com/ free-maya-rig/; Row 7: model - Alvaro Luna Bautista; model - Serguei Kalentchouk.

7. REFERENCES

[1] http://download.autodesk.com/global/docs/maya2014/en_us/

[2] http://iphome.hhi.de/suehring/tml/

[3] http://www.3drender.com/challenges/

[4] D. Derakhshani, *Introducing Autodesk Maya 2014: Autodesk Official Press*, May 2013, ISBM-13 978-1118574904.

[5] Wu-chi Feng, Feng Liu, "Understanding the Impact of Inter-Lens and Temporal Stereoscopic Video Compression", in *Proc. of NOSSDAV 2012*, Toronto, Canada, June 2012.

[6] Sang-Uok Kum, and Ketan Mayer-Patel, "Real-Time Multidepth Stream Compression", *ACM Trans. on Multimedia Computing, Communications, and Applications*, Vol. 1, No. 2, pp. 128 – 150, May, 2005.

[7] Kodak Appl. Notes, "CCD Image Sensor Noise Sources", *Image Sensor Solution Application Notes*, Jan. 2005.

[8] B. Smith, L. Zhang, H. Jin, A. Agarwala. "Light Field Video Stabilization" in *IEEE International Conference on Computer Vision (ICCV)*, Sept 29-Oct 2, 2009.

[9] R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer, 2010.

[10] G. Toffetti, M. Tagliasacchi, M. Marcon, A. Sarti, S. Tubaro, K.Ramchandran, "Image Compression in a Multi-Camera System Based on a Distributed Source Coding Approach," in Proc. *Euro. Signal Process.Conf.*, Antalya, Turkey, 2005.

[11] A. Tourapis, K. Suhring, G. Sullivan, "H.264/AVC Reference Software Manual", Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, July 2009.

[12] G. Vass, T. Perlaki, "Applying and Removing Lens Distortion in Post Production", Colorfront Ltd., 2003.

[13] D. Vatolin, D. Kulikov, A. Parhin, M. Arsaev, "MPEG-4 AVC / H.264 Video Codecs Comparison", CS MSU Graphics & Media Lab Technical Report, Moskow State University, May 2011.

[14] C. Yeo, K. Ramchandran, "Robust Distributed Multi-view Video Compression for Wireless Camera Networks", *IEEE Transactions on Image Processing*, May 2010.

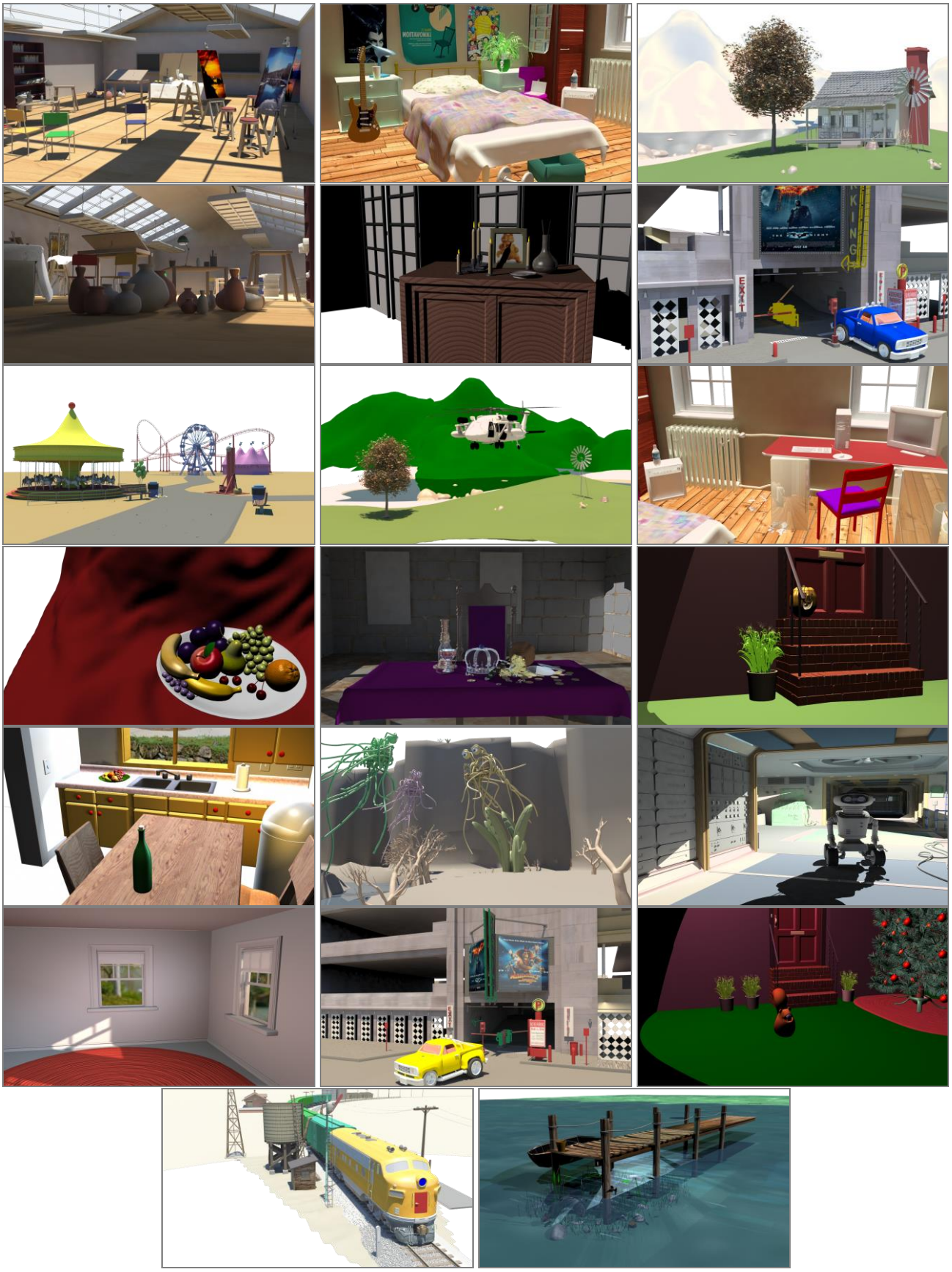


Figure 8: The 20 scenes created for the dataset