SphereCraft: A Dataset for Spherical Keypoint Detection, Matching and Camera Pose Estimation

Supplemental Material

Christiano Gava[1] Yunmin Cho[2] Federico Raue[1] Sebastian Palacio[1]
Alain Pagani[1] Andreas Dengel[1,3]
{christiano.gava,federico.raue,sebastian.palacio,alain.pagani,andreas.dengel}@dfki.de
min@aimmo.co.kr
[1] DFKI [2] Aimmo Germany GmbH [3] University of Kaiserslautern-Landau

1. Licenses and Sources

To circumvent the costs of acquiring real-world scenes — which would still lack ground-truth camera poses — or purchasing professional synthetic designs, we collected free models available on the Internet to create our set of synthetic scenes. We refrain from scenes with restrictive licenses to ensure free access to the scientific community. Furthermore, to offer an easy way for other researchers to adapt our dataset to their needs (higher resolution, more images, other camera poses, etc.) we chose Blender [4] as the rendering platform. Blender is free software, easy to install and well tested in all major operating systems.

Many scenes were enhanced with assets also freely available online. Some were used with little or no modification, like Simple Room and Urban Canyon [8]. Others were obtained directly from the Blender website, namely Barbershop, Classroom, Italian Flat, and Lone Monk. There is also a scene built in house: Vitoria. It was nonetheless augmented with several online assets. Finally, for the mannequins used to enhance Showroom, we used the SMPL-X Blender add-on [1, 7]. Table 1 summarizes the licenses of 3D models and assets along with their sources.

2. Rendering

We render and release images and depth maps at 2048 x 1024 pixels. This resolution was chosen to strike a good balance between level of detail and total size of the dataset. It is worth mentioning, however, that images and depth maps can be easily rendered at much higher resolutions. This can be achieved by simply adjusting the resolution in the configuration file(s) provided with each synthetic scene (see Sec. 3) and rendering it again. The relevant parts of each configuration file are as follows.

- `resolution_x`: the desired width of the equirectangular image, in pixels.
- `resolution_y`: the desired height of the equirectangular image, in pixels. Must be half of the width specified above. The redundancy is intentional. Moreover, this can be helpful in future versions to render images other than spherical using the same mechanism.
- `render_images`: flag to enable/disable rendering of RGB images.
- `render_depth_maps`: flag to enable/disable rendering of depth maps. Disabling the rendering of depth maps saves time and thus is convenient if depth maps are not required.
- `start_cam_index`: index of the first *anchor* camera to render. Combined with `num_anchor_cams`, it can be used to split the rendering into several GPUs.
- `num_anchor_cams`: number of *anchor* cameras to render. For small scenes, this is normally the number of anchor cameras embedded in the Blender project. For large scenes, this can be combined with `start_cam_index` to split the rendering into several GPUs.
- `num_sat_cams`: number of satellite cameras to generate for each anchor. In this paper, this number is 9 so that we form clusters of 10 images (1 anchor + 9 satellites).
- `angle_x_degrees`: a scalar determining the range of possible rotations around the x-axis of the camera, specified in degrees. For instance, 45 means rotations around the x-axis will be selected from

Scene	License	Source
Bank	3D model license Standard(free)	https://www.turbosquid.com/ko/3d-models/3d-bank-interior-furnitures-model/902643
Barbershop	CC-BY	https://www.blender.org/download/demo-files/
Berlin	3D model license Standard(free)	https://www.turbosquid.com/ko/3d-models/interior-3d-1278629
Classroom	CC0	https://www.blender.org/download/demo-files/
Garage	3D model license Standard(free)	https://www.turbosquid.com/ko/3d-models/automotive-garage-workshop-interior-3d-model-1868183
Harmony	Royalty Free License	https://www.cgtrader.com/free-3d-models/interior/living-room/interior-b
Italian Flat	CC-BY	https://www.blender.org/download/demo-files/
Kartu	3D model license Standard(free)	https://www.turbosquid.com/ko/3d-models/bedroom-bed-3d-model-1524240
Lone Monk	CC0	https://www.blender.org/download/demo-files/
Middle East	3D model license Standard(free)	https://www.turbosquid.com/ko/3d-models/middle-east-shooting-game-environment-low-poly-3d-model-1866000
Passion	3D model license Standard(free)	https://www.turbosquid.com/ko/3d-models/vrayforc4d-scene-files---3d-model-1210984
	3D model license Standard(free)	https://www.turbosquid.com/ko/3d-models/study-nook-living-room-3d-model-1601414
	3D model license Standard(free)	https://www.turbosquid.com/ko/3d-models/bedroom-set-bed-3d-1466047
	3D model license Standard(free)	https://www.turbosquid.com/ko/3d-models/shower-3d-model-1524283
Rainbow	3D model license Standard(free)	https://www.turbosquid.com/ko/3d-models/cartoonclassroominterior-3d-model-1867360
Seoul	Royalty Free License	https://www.cgtrader.com/free-3d-models/interior/house-interior/3d-floor-plan-apartment
Shapespark	3D model license Standard(free)	https://www.turbosquid.com/ko/3d-models/shapespark-example-room-model-1964014
Simple	CC BY-NC-SA 3.0	https://rpg.ifi.uzh.ch/fov.html
Tokyo	Royalty Free License	https://free3d.com/3d-model/living-room-3728.html
Urban Canyon	CC BY-NC-SA 3.0	https://rpg.ifi.uzh.ch/fov.html
Vitoria	3D model license Standard(free)	https://www.turbosquid.com/ko/3d-models/modern-kitchen-3d-1821120
	3D model license Standard(free)	https://www.turbosquid.com/ko/3d-models/3d-house-1628048
	3D model license Standard(free)	https://www.turbosquid.com/ko/3d-models/cozy-room-3d-model-1641507
	3D model license Standard(free)	https://www.turbosquid.com/ko/3d-models/bedroom-bed-model-1609547
	3D model license Standard(free)	https://www.turbosquid.com/ko/3d-models/bathroom-mirror-3d-model-1592086
	3D model license Standard(free)	https://www.turbosquid.com/ko/3d-models/3d-model-ceo-office-design-1765491
Warehouse	3D model license Standard(free)	https://www.turbosquid.com/ko/3d-models/max-warehouse-pbr-gaming/1017253

Table 1. Licenses and sources of 3D models and assets used for the creation of the synthetic scenes. Medieval Port and Showroom were available as free models when we found and downloaded them, but the links are unfortunately no longer available.

$[-45°, 45°]$. This only affects the generation of satellite cameras. Combined with `angle_y_degrees` and `angle_z_degrees` it is used to render satellite images with a number of different orientations and produce images with severe distortions.

- `angle_y_degrees`: like `angle_x_degrees`, but for the y-axis.
- `angle_z_degrees`: like `angle_x_degrees`, but for the z-axis (vertical). While in this paper rotations around x- and y-axes are limited to $[-45°, 45°]$, we allow full rotation around the z-axis, *i.e.* rotations are randomly selected from $[-180°, 180°]$.
- `max_translation`: this corresponds to the r_{max} introduced in Sec. 3.1 of the main manuscript, *i.e.* a scalar, given in meters, specifying the radius of a sphere centered at an anchor camera within which the positions of the satellite cameras will be randomly sampled.
- `random_seed`: seed used in the pseudo-random generator for the creation of the satellite cameras. Satellite cameras will be rendered at exactly the same poses as long as this seed remains unchanged. This ensures reproducibility but can also be used to create a different set of satellite cameras.

Along with the data, we also release a Python script for rendering the synthetic scenes. It takes a configuration file as input and produces the `images`, `depthmaps` (if enabled), and `extr` directories (see Sec. 3). Depending on the scene and desired resolution, rendering can take some time. To address this, we designed the configuration files to allow parallel rendering. This is the reason configuration files have the `start_cam_index` and `num_anchor_cams` entries outlined above. Together, they can be used to split the whole scene into chunks. Each chunk leads to a unique configuration file. By invoking the rendering script with different configuration files, it is possible to render the scene in parallel using multiple GPUs, one for each configuration file. Naturally, the `random_seed` should be modified in each configuration file to prevent the generation of duplicate satellite camera poses.

3. Data Organization

Each synthetic scene is organized in directories and files as follows:

- `blender`: directory containing the Blender project. Anchor cameras are embedded. Generation of satellite cameras is controlled by the configuration files.
- `config`: directory containing one or more configuration files. Each file specifies the parameters used for rendering the scenes with Blender [4], such as resolution, number of satellite cameras per anchor camera, whether to render depth maps, range of rotations around each axis, etc. See Sec. 2 for details.
- `depthmaps`: directory containing the depth maps stored as 32-bit float, single-channel, `exr` files. We use lossless compression for maximum accuracy.
- `extr`: directory containing the camera poses (or extrinsics) as `dat` files. Each file contains the rotation

matrix R stored as a 3×3 matrix that, following convention [6], encodes the transformation from world to camera coordinate system. However, here the vector t does not follow convention and represents the camera position in world coordinate system. This is convenient to compute distances between cameras and determine neighborhood.

- `images`: directory containing the images as `jpg` files.

- `keypoints`: directory containing a sub-directory named after each keypoint detector used in this paper (`akaze`, `kp2d` and so on). Files are stored as `npz` and follow the same convention regardless of the keypoint detector used to produce it. Each `npz` is composed of three `npy` files: the first, `keypointCoords.npy`, contains the location of each keypoint on the unit sphere using (ϕ, θ) coordinates, representing latitude and longitude, respectively. This representation is simultaneously compact and independent of the resolution of the images. It also allows keypoints detected on images of different resolutions to be transparently integrated into training or testing of neural networks. The second, `keypointDescriptors.npy`, stores all keypoint descriptors. Finally, `keypointScores.npy` stores all keypoint scores, *i.e*. their strength or response.

- `gt_correspondences`: like `keypoints`, it is subdivided into directories containing `npz` files for each separate keypoint detector. Each file contains two embedded `npy` files: The first, `correspondences.npy`, stores the index of the ground-truth keypoint correspondence in the second image for each keypoint in the first image, or -1 if no correspondence is found. The second, `scores.npy`, informs the confidence on the ground-truth correspondence as a number in the range $[0, 1]$, computed based on the similarity between the descriptors of the matched keypoints. This confidence, or score, reflects how "important" a keypoint match is and can be used to train and evaluate learn-based keypoint matchers.

- `mesh`: directory storing the `.obj` file representing the 3D mesh.

- `[keypoint detector]_train.txt`: list of proposed training image pairs for each keypoint detector used in this paper, *e.g*. `sift_train.txt`. Exclusive to synthetic training scenes. See Sec. 3.1 and Table 2 in the main manuscript.

- `[keypoint detector]_val.txt`: list of proposed validation image pairs for each keypoint detector. Also exclusive to synthetic training scenes.

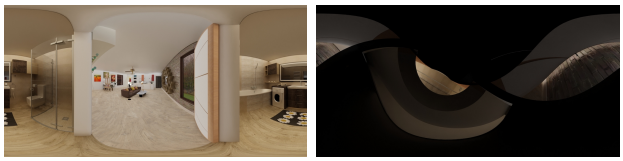

Figure 1. Anchor image (left) and a defective satellite (right). This satellite is added to the black list.

- `[keypoint detector]_test.txt`: list of proposed testing image pairs for each keypoint detector. Exclusive to synthetic testing scenes. See Sec. 3.1 and Table 3 in the main manuscript.

- `black_list.txt`: a collection of images to be excluded from training, evaluation and testing. The reason is as follows. As satellite cameras are randomly generated, sometimes the resulting location is physically unrealistic, like under the floor or "inside" a piece of furniture (like a couch) nearby the anchor camera, yielding meaningless or defective images (see Fig. 1). To exclude these images from training, evaluation and testing, after rendering, all images are inspected by a human. When such an image is identified, it is added to this file which is later used to discard these images when creating the `*_train.txt`, `*_val.txt` and `*_test.txt` files above.

Real scenes are reserved for testing and contain only a subset of the directories specified above, namely `images` and `keypoints`. Although it is possible to use a spherical SfM [5] approach to estimate camera neighborhood and then create a `test.txt` file, we intentionally omit `test.txt` to motivate the prediction of keypoint correspondences for all possible image pairs.

4. Real Scenes

Along with the synthetic scenes, we release a set of 9 real scenes: 4 indoors and 5 outdoors. Table 2 summarizes the type of the scene, the number of images (N), resolution and camera used to capture the scene. Sample images are shown in Fig. 6 in the main manuscript, where it is possible to notice a black stripe at the bottom. This is the result of masking out the tripod used to support the camera during the acquisition process. Images were properly anonymized and do not depict any human or personal data that can be associated to any particular person.

References

[1] SMPL-X Blender add-on. https://smpl-x.is.tue.mpg.de/index.html. Retrieved June, 2023. 1

[2] Weiss AG. Civetta. https://weiss-ag.com/civetta360camera/. Retrieved June, 2023. 4

Scene	Type	N	Resolution	Camera
Berlin Street	O	186	7070 x 3535	Civetta
Church	I	54	7070 x 3535	Civetta
Corridors	I	116	5376 x 2688	Ricoh Theta-S
Meeting Room 1	I	18	5376 x 2688	Ricoh Theta-S
Meeting Room 2	I	21	5376 x 2688	Ricoh Theta-S
Stadium	O	74	7070 x 3535	Civetta
Town Square	O	35	7070 x 3535	Civetta
Train Station	O	112	7070 x 3535	Civetta
Uni	O	71	7070 x 3535	Civetta

Table 2. Real scenes captured with Civetta [2] and Ricoh Theta-S [3] cameras. Regarding the scene type, I = Indoor, O = Outdoor.

[3] Ricoh Company. Ricoh Theta S. https://theta360.com/en/about/theta/s.html. Retrieved June, 2023. 4

[4] Blender Foundation. Blender. http://www.blender.org/. Retrieved March, 2023. 1, 2

[5] Christiano Couto Gava and Didier Stricker. A Generalized Structure from Motion Framework for Central Projection Cameras. In *Computer Vision, Imaging and Computer Graphics Theory and Applications*. Springer, 2016. 3

[6] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 3

[7] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive Body Capture: 3D Hands, Face, and Body from a Single Image. In *CVPR*, pages 10975–10985, 2019. 1

[8] Zichao Zhang, Henri Rebecq, Christian Forster, and Davide Scaramuzza. Benefit of Large Field-of-View Cameras for Visual Odometry. In *ICRA*, pages 801–808. IEEE, 2016. 1