Annex A. The Qualcomm Rasterized Images for Super-resolution Processing dataset

Figure 11. Example of data modalities available in the QRISP dataset. *(first row, from left to right)*: Native 270p, Negative 2 mipmap biased 270p, Negative 1.58 mipmap biased 360p, Negative 1 mipmap biased 540p. *(second row, from left to right):* 540p depth, 540p motion vectors, Native 1080p, Enhanced 1080p

A.1. Motivation

The Qualcomm Rasterized Images for Super-resolution Processing dataset was created to facilitate the development and research of super-resolution algorithms for gaming. The dataset consists of parallel captures of various scenes in different modalities and resolutions. It is designed to be diverse, with a variety of backgrounds and models, to better generalize to new video games. The dataset can be found at `https://developer.qualcomm.com/software/ai-datasets/qualcomm-rasterized-images`.

A.2. Dataset modalities

The dataset consists of sequences of consecutive frames captured at 60 frames per second. For each frame, multiple modalities are rendered at different resolutions ranging from 270p to 1080p. Table 4 provides a list of these modalities, which can be either generated using default parameters or mipmap biased, jittered, or both mipmap biased and jittered. In addition to the modalities listed in the table, a JSON file containing camera parameters (including jitter offsets) is included for each segment. The following provides details about each type of modality:

Color Rendered images are stored in 8-bit RGBA PNG files. In addition to low-resolution and native renders at 1080p resolution, we generate an additional color modality referred to as "Enhanced". This was produced by rendering color images at 2160p with MSAAx8 applied followed by 2x downsizing, resulting in a high-quality 1080p target image.

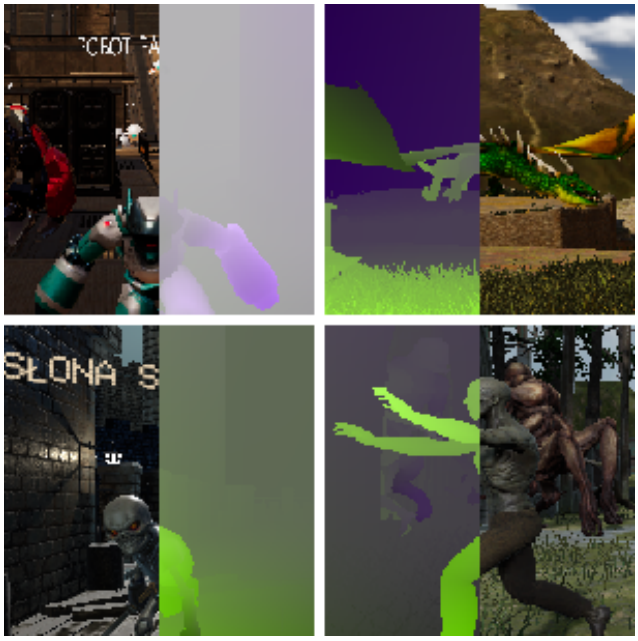

Figure 12. Example of animated characters and textual UI elements added manually to each scene. For each crop, we show both color and motion information to illustrate that these objects are moving and that the added UI does not have associated motion (nor depth) information.

Motion Motion vectors are stored in a 16-bit EXR file, with the vertical velocity stored in the first channel and the horizontal velocity stored in the second. Unity uses a Y-up coordinate system so the vertical velocities may need to be negated to match the coordinate system used during motion compensation. Values are stored in the $[-1, 1]$ range, so they need to be scaled by the number of pixels in the corresponding dimension to convert the velocity in pixel units.

Depth We save the depth information from the 32-bits z-buffer in the four channels of an 8-bit PNG file. The depth value corresponds to the distance between the rendered pixel and the camera near plane, scaled to the $[0, 1]$ range, where 0 represents the near plane and 1 the far plane. To convert back from $[0, 255]^4$ to $[0, 1]$, we used the following equation:

$$depth = R/255 + G/255^2 + B/255^3 + A/255^4 \quad (4)$$

Where R, G, B, and A are the four 8-bit channels from the PNG file.

Jittering and other camera-related information For each segment, we provide a JSON file with camera intrinsic parameters (near plane, far plane and FOV) and frame-level information (jitter offset, camera position and orientation).

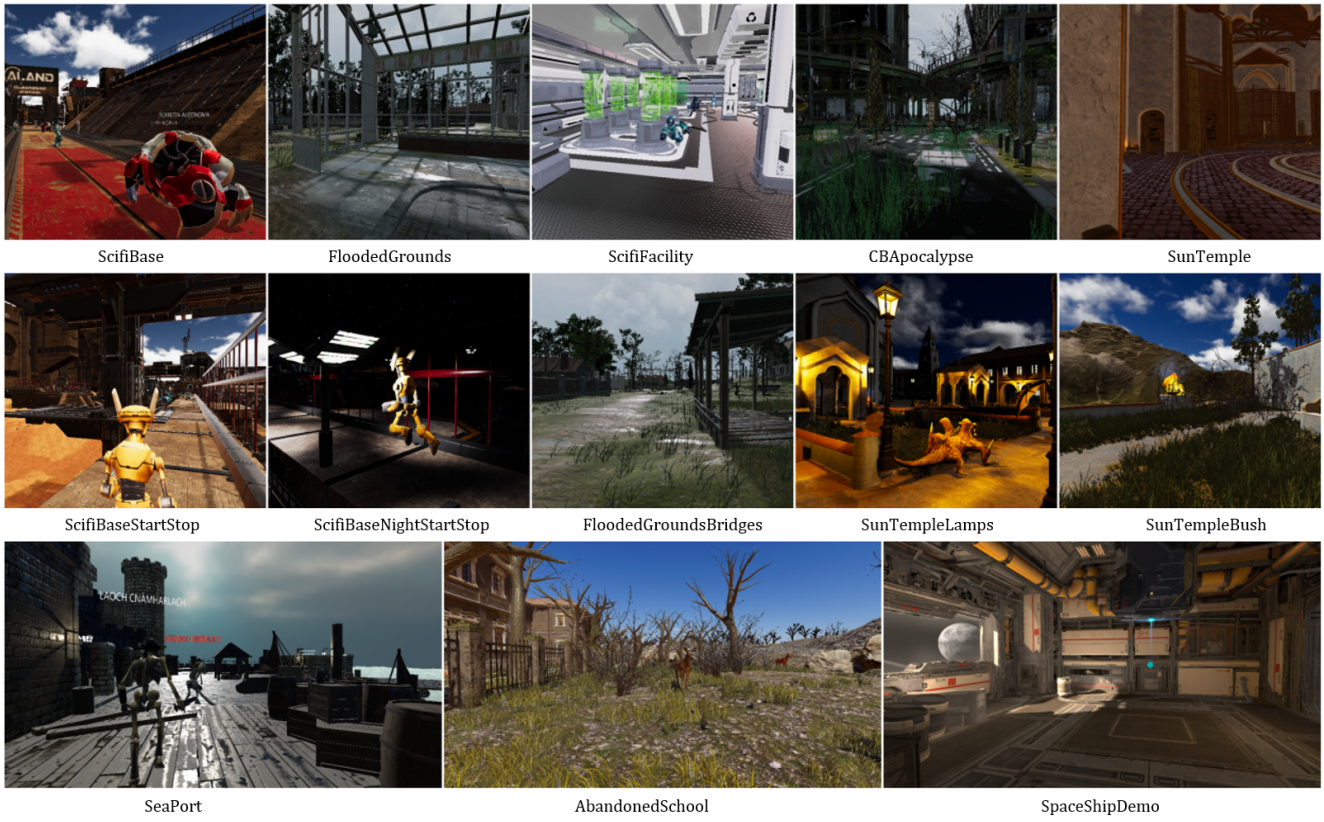

Figure 13. Overview of the 13 scenes from the QRISP dataset. The first two rows correspond to the 10 training scenes. The last rows shows the 3 scenes used for evaluation.

A.3. Dataset composition and collection process

The dataset consists of renders from 13 scenes in total, with 10 of them allocated for training and the remaining 3 reserved for evaluation. Table 5 provides a list of the scenes and the number of segments and frames available for each.

Scenes compositions Our dataset includes 3D assets sourced either from the Unity Asset Store (link), or from open-source GitHub projects. To make the data more representative of real-world gaming scenarios, we manually add animated characters to the scene. We also occasionally add textual UI elements on top of animated characters to make the algorithms more robust to elements without associated depth or motion vector information. A list of the assets used is provided in Table 6.

Capture process In Unity, we use multiple "twin" cameras and shaders which all follow the same path to capture multiple resolutions and modalities simultaneously within the same frame. To ensure synchronization of pixels between the different renders, we may need to adjust certain parameters on a per-scene basis to make sure the rendering and animation is framerate dependent, like removing any asynchronous effects (e.g. wind physics for grass). We obtain jittered modalities by shifting the corresponding camera by a sub-pixel offset drawn from a cyclic Halton$(2, 3)$ sequence of length 16 and sometimes include "stops" in the camera path where the camera remains stationary.

Preprocessing/cleaning/labeling Frame-blurring post processing (bloom, motion blur, etc.) and anti-aliasing was disabled for all modalities, except for the "Enhanced" modality where we used MSAA as described earlier.

Camera path and recording Sections of camera path were pre-recorded using the Unity game engine running at 60 fps (frames per second). During capturing, the camera followed the pre-recorded path through the scene and frames were captured at regular intervals of 30 frames for training data and 300 frames for evaluation data. In the case of training scenes including camera stops (see Table 5), the camera usually stayed stationary for 10 frames. Most training scenes have 120 frames between each segment of frames to increase diversity.

A.4. Commercial baselines

In this dataset, we have also included images generated by commercial solutions integrated into Unity on the same frames used for evaluation. At the time of the dataset collection, these included Nvidia's DLSS 2.2.11.0 and AMD's FSR 1.2, which can serve as reference baselines to assess the performance of new algorithms.

A.5. Potential use beyond super-resolution

While this dataset was primarily created to facilitate the development of super-resolution algorithms for gaming applications; however, we believe that it could be useful for other tasks, such as optical flow estimation.

Resolution	Name	Jittering applied?	Mipmap bias offset	Modality type
1080p	Native	No	0	Color
	Enhanced	No	NA	Color
540p	DepthMipBiasMinus1	No	-1	Depth
	DepthMipBiasMinus1Jittered	Yes	-1	Depth
	MipBiasMinus1	No	-1	Color
	MipBiasMinus1Jittered	Yes	-1	Color
	MotionVectorsMipBiasMinus1	No	-1	Motion vector
	MotionVectorsMipBiasMinus1Jittered	Yes	-1	Motion vector
	Native	No	0	Color
360p	DepthMipBiasMinus1.58	No	-1,58	Depth
	DepthMipBiasMinus1.58Jittered	Yes	-1,58	Depth
	MipBiasMinus1.58	No	-1,58	Color
	MipBiasMinus1.58Jittered	Yes	-1,58	Color
	MotionVectorsMipBiasMinus1.58	No	-1,58	Motion vector
	MotionVectorsMipBiasMinus1.58Jittered	Yes	-1,58	Motion vector
	Native	No	0	Color
270p	DepthMipBiasMinus2	No	-2	Depth
	DepthMipBiasMinus2Jittered	Yes	-2	Depth
	MipBiasMinus2	No	-2	Color
	MipBiasMinus2Jittered	Yes	-2	Color
	MotionVectorsMipBiasMinus2	No	-2	Motion vector
	MotionVectorsMipBiasMinus2Jittered	Yes	-2	Motion vector
	Native	No	0	Color

Table 4. Per-resolution breakdown of the modalities available in the QRISP dataset. Each modality can be either generated using default parameters, mipmap biased, jittered, or both mipmap biased and jittered.

Scene	Split	Segments	Frames Per Segment	Total Frames	Includes stops?
FloodedGrounds	Train	21	30	630	No
SciFifacility	Train	21	30	630	No
SunTemple	Train	36	30	1080	No
CB-Apocalypse	Train	30	30	900	No
SciFiBase	Train	41	30	1230	No
FloodedGroundsBridges	Train	20	30	600	Yes
ScifiBaseNightStartStop	Train	20	30	600	Yes
ScifiBaseStartStop	Train	21	30	630	Yes
SunTempleBush	Train	11	30	330	Yes
SunTempleLamps	Train	21	30	630	Yes
AbandonedSchool	Test	2	300	600	Yes
SpaceShipDemo	Test	2	300	600	Yes
Seaport	Test	1	300	300	Yes
Total Training	-	242	30	7260	-
Total Test	-	5	300	1500	-
Total	-	-	-	8760	-

Table 5. List of dataset scenes, with split information, the number of segments and frames per segment, and whether it includes static segments where the camera remains stationary.

Scene	Assets Used	Source
CB-Apocalypse	CBU: Apocalypse Edition	Unity Asset Store
	ULTIMATE ANIMATION COLLECTION	Unity Asset Store
	Animal Pack Deluxe	Unity Asset Store
	Customizable Survivors Pack	Unity Asset Store
SciFifacility	Sci-Fi Facility	Unity Asset Store
	Robot Warriors Cartoon	Unity Asset Store
FloodedGrounds FloodedGroundsBridges	Flooded Grounds	Unity Asset Store
	Ghoul-zombie	Unity Asset Store
	Zombie	Unity Asset Store
	Fantastic Creature #1	Unity Asset Store
SciFiBase ScifiBaseNightStartStop ScifiBaseStartStop	Sci-Fi base	Unity Asset Store
	Robot 1	Unity Asset Store
	Robot Warriors Cartoon	Unity Asset Store
	Robot Sphere	Unity Asset Store
SunTemple	Sun Temple	Unity Asset Store
	Animal Pack Deluxe	Unity Asset Store
	Dragon for Boss Monster : HP	Unity Asset Store
SunTempleBush	Sun Temple	Unity Asset Store
	Real Landscapes - Valley Forest	Unity Asset Store
SunTempleLamps	Sun Temple	Unity Asset Store
	Dragon for Boss Monster : HP	Unity Asset Store
	Flooded Grounds	Unity Asset Store
AbandonedSchool	Animal Pack Deluxe	Unity Asset Store
	HQ Abandoned School (Modular)	Unity Asset Store
SpaceShipDemo	Space Ship Demo	https://github.com/Unity-Technologies/SpaceshipDemo
Seaport	Old Sea Port	Unity Asset Store
	Fantasy Monster - Skeleton	Unity Asset Store
	Dungeon Skeletons Demo	Unity Asset Store

Table 6. List of assets used for each scene.

Annex B. Supplementary results

Upscaling	Model	AbandonedSchool			SeaPort			SpaceShipDemo			Average		
		PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
2×	Bicubic	28.69	0.8257	0.218	26.18	0.8623	0.239	33.66	0.9137	0.200	29.51	0.8672	0.219
	DLSS 2.2	29.78	0.8428	0.173	27.41	0.8880	0.203	33.43	0.9140	0.185	30.21	0.8816	0.187
	Xiao *et al.*	30.84	0.8632	0.145	28.43	0.9079	0.165	36.39	0.9516	0.110	31.89	0.9075	0.140
	Ours-S (f16-m1)	30.15	0.8477	0.156	28.08	0.8971	0.183	35.30	0.9374	0.140	31.18	0.8941	0.160
	Ours-M (f32-m3)	30.66	0.8564	0.136	28.62	0.9086	0.169	36.11	0.9482	0.116	31.80	0.9044	0.140
	Ours-L (f64-m5)	**31.03**	**0.8650**	**0.134**	**29.00**	**0.9154**	**0.161**	**36.60**	**0.9541**	**0.107**	**32.21**	**0.9115**	**0.134**
3×	Bicubic	26.77	0.7434	0.304	24.61	0.8039	0.356	31.46	0.8628	0.306	27.61	0.8034	0.322
	Xiao *et al.*	29.11	0.8039	0.197	27.01	0.8864	0.239	34.61	0.9284	0.163	30.24	0.8729	0.200
	Ours-M (f32-m3)	29.04	0.7983	0.188	27.24	0.8749	0.243	34.41	0.9234	0.178	30.23	0.8655	0.203
	Ours-L (f64-m5)	**29.32**	**0.8043**	**0.175**	**27.68**	**0.8857**	**0.232**	**35.02**	**0.9341**	**0.153**	**30.67**	**0.8747**	**0.187**
4×	Bicubic	25.61	0.6834	0.365	23.59	0.7577	0.432	30.06	0.8194	0.376	26.42	0.7535	0.391
	Xiao *et al.*	27.94	0.7550	0.248	25.91	0.8510	0.306	33.22	0.9031	0.222	29.02	0.8364	0.259
	Ours-M (f32-m3)	27.91	0.7495	0.234	26.19	0.8443	0.306	33.08	0.8976	0.234	29.06	0.8305	0.258
	Ours-L (f64-m5)	**28.16**	**0.7566**	**0.214**	**26.55**	**0.8554**	**0.291**	**33.54**	**0.9090**	**0.209**	**29.42**	**0.8403**	**0.238**

Table 7. Per-scene breakdown of PSNR, SSIM and LPIPS scores for our model, compared to DLSS 2.2 and our implementation of Xiao *et al.* [59] for 2×, 3× and 4× upscaling.