Supplementary Material
Totems: Physical Objects for Verifying Visual Integrity

In this document, we first discuss additional implementation details regarding ray refraction operations and training of the radiance field (Section A). We conduct additional experiments investigating the number and configuration of totems, and show results on more scenes in Section B. We address different modes of image manipulation and provide a brief FAQ in Section C.

A Additional method details

A.1 Pixel-to-ray mapping

Overview. Recall that the first step of our method is to infer the underlying 3D scene using the refracted rays corresponding to the distorted totem pixels. For a given image $\mathcal{I}$ and a set of spherical totems $\mathcal{J}$ indexed $j = \{1 \dots |\mathcal{J}|\}$, with center positions P_j relative to the camera, radii R_j, and IoR n_j, we compute the mapping from a totem pixel in the image $\mathcal{I}_{u,v}$ to the scene light ray corresponding to refraction through the totem $\mathbf{r}_{out} = \mathbf{o}_{out} + \mathbf{d}_{out} * t$. We decompose this mapping procedure into two steps:

1. Begin with a ray $\mathbf{r}_{in} = \mathbf{o}_{in} + \mathbf{d}_{in} * t$ corresponding to pixel $\mathcal{I}_{u,v}$. Compute the first intersection D with totem j and the direction $\mathbf{d}_{ref_1}$ of the refracted ray entering the totem.
2. Take the intermediate ray $\mathbf{r}_{mid} = \mathbf{o}_{mid} + \mathbf{d}_{mid} * t$, where $\mathbf{o}_{mid} = D$ and $\mathbf{d}_{mid} = \mathbf{d}_{ref_1}$. Compute the second intersection E with totem j and the direction $\mathbf{d}_{ref_2}$ of the refracted ray exiting the totem.

In Sec. 4.1 in the main text, we use a general intersect function to compute the ray-totem intersections D and E and a general refract function to compute the refracted ray directions $\mathbf{d}_{ref_1}$ and $\mathbf{d}_{ref_2}$. Below we provide the formula and implementation details for these two functions.

Define intersect. Given a ray $\mathbf{r} = \mathbf{o} + \mathbf{d} * t$ and a sphere with radius R and center position P, the intersect function first confirms the validity of the ray-sphere intersection, then computes the two intersection positions and returns the one closest to the ray origin $\mathbf{o}$ and in the ray direction $\mathbf{d}$. Since an intersection X must satisfy both the sphere equation $\|X - P\|_2^2 = R^2$ and the ray equation $X = \mathbf{o} + \mathbf{d} * t$, we formulate the intersect function as the optimization below:

$$\mathsf{intersect}(P, R, \mathbf{d}, \mathbf{o}) := \mathbf{o} + \mathbf{d} * \left(\arg\min_t \left| R^2 - \|\mathbf{o} + \mathbf{d} * t - P\|_2^2 \right| \right) \quad (1)$$

To solve for t, we set the inner optimization term to 0 and use a quadratic solver $\mathsf{quad}\,(a, b, c)$ with input arguments $a = \|\mathbf{d}\|_2^2$, $b = 2\langle \mathbf{o} - P, \mathbf{d} \rangle$, $c = \|\mathbf{o} - P\|_2^2$.

Before this step, we confirm that the input ray has valid intersections with the sphere by checking if the discriminant term in the quadratic solution is positive.

Define refract. The function refract computes the refracted direction $\mathbf{d}_{ref}$ of an incident ray $\mathbf{d}_{in}$. Given the unit surface normal $\mathbf{N}$ at the ray-medium intersection, IoR of the incident medium n_1 and the refractive medium n_2, we derive an analytical solution using the Snell's law:

$$\mathsf{refract}(n_1, n_2, \mathbf{N}, \mathbf{d}_{in}) := \frac{n_1}{n_2}\left(\mathbf{N} \times (-\mathbf{N} \times \mathbf{d}_{in})\right) - \mathbf{N}\sqrt{1 - \frac{{n_1}^2}{{n_2}^2}\|\mathbf{N} \times \mathbf{d}_{in}\|_2^2} \quad (2)$$

A.2 Totem pose

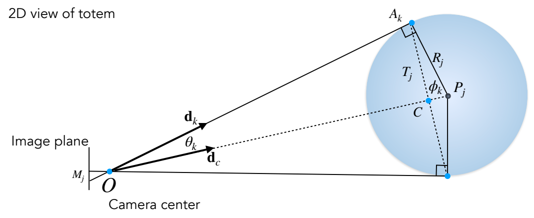

Fig. 1: 2D visualization of the intersection between the spherical totem j and tangent rays $\mathbf{r}_k = O + \mathbf{d}_k * t$ corresponding to the boundary pixels of totem mask M_j. This intersection forms a circle in 3D space with radius T_j and center C. We use this circle to obtain an initial estimate of the totem center P_j and also use its 2D projection on the image plane to regularize the totem position during joint optimization by enforcing consistency with the provided totem mask M_j.

Totem pose initialization. To obtain an initial estimate of the totem positions, we use an optimization procedure to fit the boundary of the mask pixels M_j to a circle that corresponds to the intersection between the spherical totem j and the tangent cone formed by the boundary rays (Figure 1).

First, assuming the camera center O is at the origin, we express the rays corresponding to the boundary pixels of a totem mask M_j as $\mathbf{r}_k = \mathbf{d}_k * t$ and normalize $\mathbf{d}_k$ to have unit length. When projected into 3D space, these K rays form a tangent cone with the totem, and we estimate the cone axis by averaging the unit-length boundary vectors:

$$\mathbf{d}_c = \frac{1}{K}\sum_k \mathbf{d}_k. \quad (3)$$

We then normalize $\mathbf{d}_c$ to unit length and solve for the angle between $\mathbf{d}_k$ and $\mathbf{d}_c$:

$$\theta_k = \arccos(\mathbf{d}_k \cdot \mathbf{d}_c). \quad (4)$$

With known totem radius R_j and given that the tangent ray $\mathbf{r}_k$ is perpendicular to $\overrightarrow{P_jA_k}$, where A_k is the point of tangency, we solve for the complementary angle $\phi_k = \frac{\pi}{2} - \theta_k$ and estimate the radius of the circular intersection $T_j = \frac{1}{K}\sum_k R_j \sin(\phi_k)$. Next, we solve for the slant height t_{est} of the tangent cone by minimizing the objective function:

$$t_{est} = \arg\min_t \left| \frac{1}{K} \sum_k ||\mathbf{d}_k * t - C|| - T_j \right|, \tag{5}$$

where C is the cone base center expressed as $C = \frac{1}{K}\sum_k \mathbf{d}_k * t$. Intuitively, the boundary rays of a totem mask M_j defines a cone with a fixed opening angle and we optimize the slant height t_{est} for this cone such that the cone radius matches the previously solved radius T_j. Using the estimated t_{est}, we then compute the estimated totem center P_j step by step:

$$C = \frac{1}{K}\sum_k \mathbf{d}_k * t_{est} \tag{6}$$

$$|\overrightarrow{OC}| = ||C|| \tag{7}$$

$$|\overrightarrow{P_jC}| = \frac{T_j^2}{|\overrightarrow{OC}|} \quad \text{(similar triangles)} \tag{8}$$

$$|\overrightarrow{P_jO}| = |\overrightarrow{P_jC}| + |\overrightarrow{OC}| \tag{9}$$

$$P_j = \mathbf{d}_c * |\overrightarrow{P_jO}|. \tag{10}$$

Due to inaccuracies in the totem masks, particularly for real images in which the totems are manually segmented, we note that the above procedures involve a number of approximations. Thus, we find that using this as an initialization and further refining the totem positions yields better reconstruction results.

Totem IoU Loss. To regularize the totem positions during optimization, we use an IoU loss between the bounding box extracted from the totem mask and the bounding box estimated from the totem position during training. To obtain the latter bounding box, we reverse some of the calculations above and start with the current totem position P_j and camera origin O to obtain segment $|\overrightarrow{P_jO}|$; together with known radius R_j, we solve for the circle radius T_j and cone center C using similar triangles. We obtain the normal vector of the circular intersection as:

$$\mathbf{n} = \frac{P_j - C}{||P_j - C||}. \tag{11}$$

With the normal vector $\mathbf{n}$, center C, and radius T_j, we have a defined 3D circle. Next, we evenly sample $N = 1000$ points along the circle and project them onto the image plane using perspective projection. Taking the minimum and maximum x and y coordinates of these projected points yields the bounding box used for the IoU loss.

A.3 Radiance field training

Pre-processing. We describe two pre-processing steps for improving reconstruction quality. First, for each scene light ray $\mathbf{r}_{out} = \mathbf{o}_{out} + \mathbf{d}_{out} * t$ computed from a totem pixel $\mathcal{I}_{u,v}$ (Sec. 4.1 main text), we shift $\mathbf{o}_{out}$ to the ray's intersection with the plane $z = 0$ and scale $\mathbf{d}_{out}$ to have unit length in the z direction. Next, we map the normalized rays from camera space to a cube space $[-1, 1]^3$ and filter out rays if the mapped ray origins fall outside of the cube space by a threshold. This automatically removes rays with large refraction angles. These rays can make training unstable, as small updates to the totem positions result in large changes in refracted ray directions.

Training details. We first train the neural radiance model alone for 100 epochs and then jointly optimize with the totem positions for another 49.9k epochs. Training takes approximately 5 hours on one NVIDIA GeForce RTX 2080 Ti. For the neural radiance model, we follow the same training and rendering procedures in Mildenhall *et al.* [1]. During joint optimization, instead of estimating the absolute totem positions, we learn the relative translation from initial totem positions obtained in Sec. A.2. For training the totem parameters, we use the Adam optimizer with a learning rate of 0.00001 and scales the learning rate with $\gamma = 0.99$ every 100 epochs.

B Additional experiments

B.1 Number of totems

We experiment with placing different numbers of totems in a simulated scene in Fig. 2. While using two totem views results in a poor reconstruction of the scene, the result improves when using four or six totems. Quantitatively, we find that using two totems yields the worst reconstruction error (0.16 L1 error), and using four and six totems attains better reconstruction error with four totems being slightly better (0.08 and 0.09 respectively). We note that while the reconstructions from four and six totems are also qualitatively similar, placing more totems results in more occlusion of the actual scene. Therefore, we chose to proceed with a four-totem scene setup in further experiments.

B.2 Patch-level detection

In addition to detecting image-level manipulation, here we detect whether each patch of an image is manipulated and provide new quantitative measures and visualizations to demonstrate the performance of our detection method.

To remain consistent with Tab. 2 in the main paper, we use the same 37 images, including 7 unmaniplated images, 7 CAF images, 8 spliced images, and 15 images with added color patches. For each image, we extract 900 patches of 64×64 resolution, over a 30×30 grid evenly spaced horizontally and vertically above the totem area. After extraction, only patches that overlap with corresponding protect regions are kept. This results in a total of 17064 patches, of

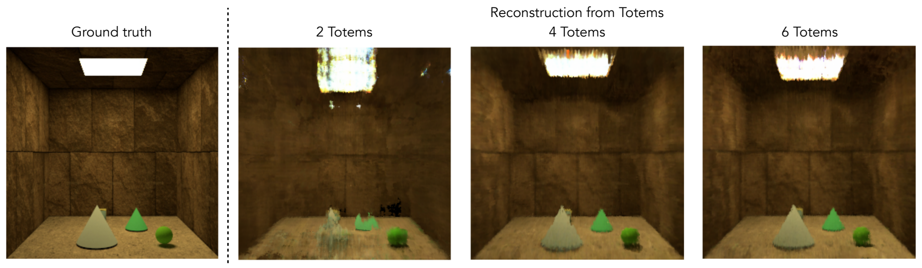

Fig. 2: **Number of totems.** Reconstructions obtained by varying the number of totems in the scene. We find that four totems is sufficient to obtain a reasonable reconstruction, while also balancing the visibility of the background scene.

which 1621 are manipulated (that is, having $> 10\%$ manipulated pixels). The exact number of patches for each manipulation can be found in Figure 3. For each patch, we compute L1 or LPIPS against the corresponding patch from the reconstructed view.

In Figure 3, we visualize the distribution of our two metrics (L1 and LPIPS) for each type of patches. For both metrics, the distributions exhibits a qualitative difference between real patches and manipulated ones, giving an overall lower score to real patches. Indeed, our metrics help detecting manipulations. In Table 1, they lead to nontrivial gains in terms of average precision. Note the imbalance between the numbers of real and manipulated patches.

		CAF+Real (7.26% manip.)	Splice+Real (8.20% manip.)	Color+Real (5.78% manip.)	All Patches (9.50% manip.)
Ours with totem opt.	+L1	0.4412	0.5026	0.8554	0.6455
	+LPIPS	0.4954	0.6315	0.8169	0.7086
Naïve Detector (Random Decision)		0.0726	0.0820	0.0578	0.0950

Table 1: **Patch-level detection comparisons:** Average precision of our method on patches created with different types of images. For example, CAF+Real contains all patches from CAF images and unmanipulated images, while the last column (All Patches) shows results on all patches from all images. To show class imbalance, we also report the precision of a naïve detector that randomly detects its output, which equals the ratio of manipulated patches.

B.3 Totem configuration

Totems can be placed anywhere between the camera and the scene region to be protected (*e.g.* the subject). We show reconstruction and detection results

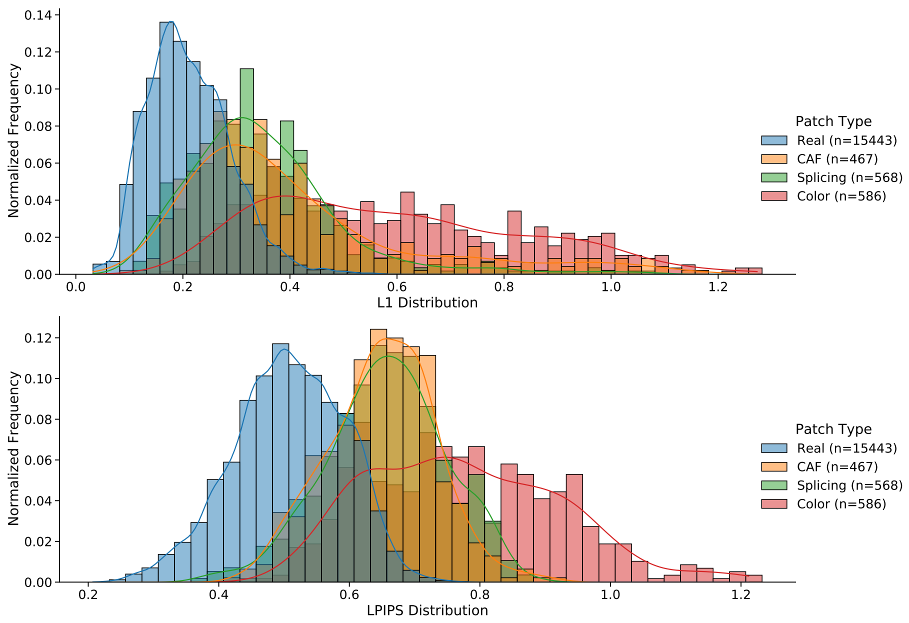

Fig. 3: Distributions of L1 and LPIPS metrics on *patches* (unmanipulated *real* patches and patches manipulated in different ways). For both metrics, our method overall gives real patches lower scores than manipulated ones.

(Fig. 4) for the same scene and manipulation type (CAF) while varying the totem configuration.

Note that configurations that contain totems farther from the camera (row 1 and 4) have smaller protected regions. This is not due to reduced reconstruction quality (column 3), but because we use the same density threshold (Sec 4.2 main text) for images with less number of totem pixels (*i.e.* overall lower projection density). Future work can explore a less rigorous threshold strategy that takes the total amount of totem pixels into account.

B.4 Additional results

In Fig. 5, we show additional detection results for the following manipulations: 1) inserting randomly colored patches, 2) adding people by image splicing, 3) removing people with Photoshop CAF, or 4) shifting people in both camera and totem views to the same reference position. Our method measures the patch-wise L1 distance between the protected region of the reconstruction and the manipulated image and shows a heatmap that highlights potential manipulations.

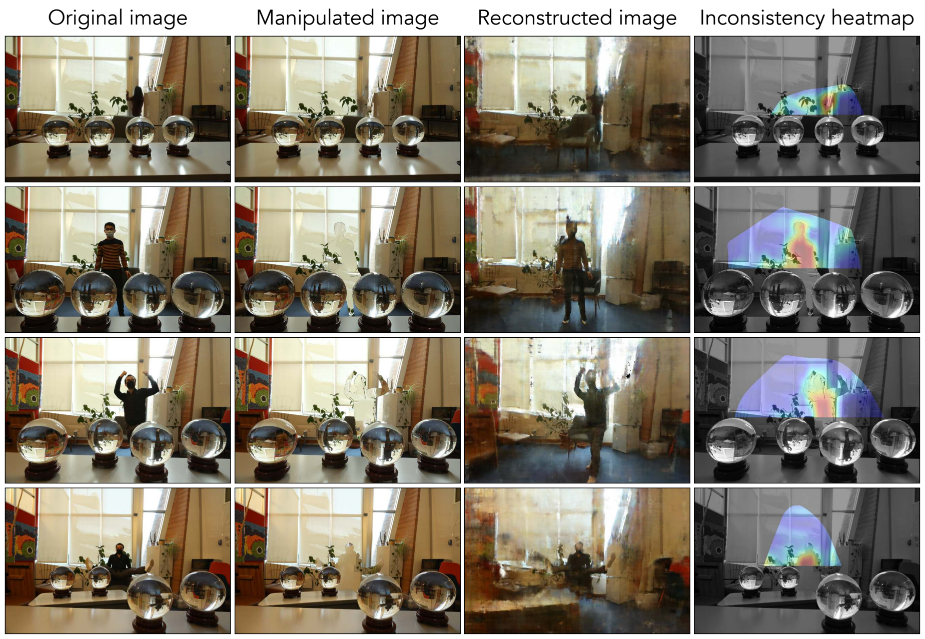

Fig. 4: **Comparison of different totem configurations.** Additional reconstruction and detection results for the same scene while varying the subject and totem configurations. We manipulate all above scenes by removing people with Photoshop Content Aware Fill. Our method has consistent reconstruction quality and detection results under various totem configurations.

C Scope of the totem framework

The goal of the totem framework is to propose a novel geometric and physical approach to image forensics, demonstrate its potential, and inspire further cross-domain research. While our current method and choice of totems cannot yet defend all types of manipulations, we discuss specific manipulation settings and use cases below.

C.1 Discussion of possible attacks

Image manipulation. Totem identities are unknown to the adversary a priori. Spherical totems are more visible due to their interpretable distortion patterns. This prompts the adversary to manipulate the totem views to avoid being detected under human inspection. In an ideal scenario, a totem with more complex and compact geometry would be less noticeable and ultimately less likely to be manipulated by the adversary. For such a case, we have demonstrated through many examples that when only the image is manipulated and totem views re-

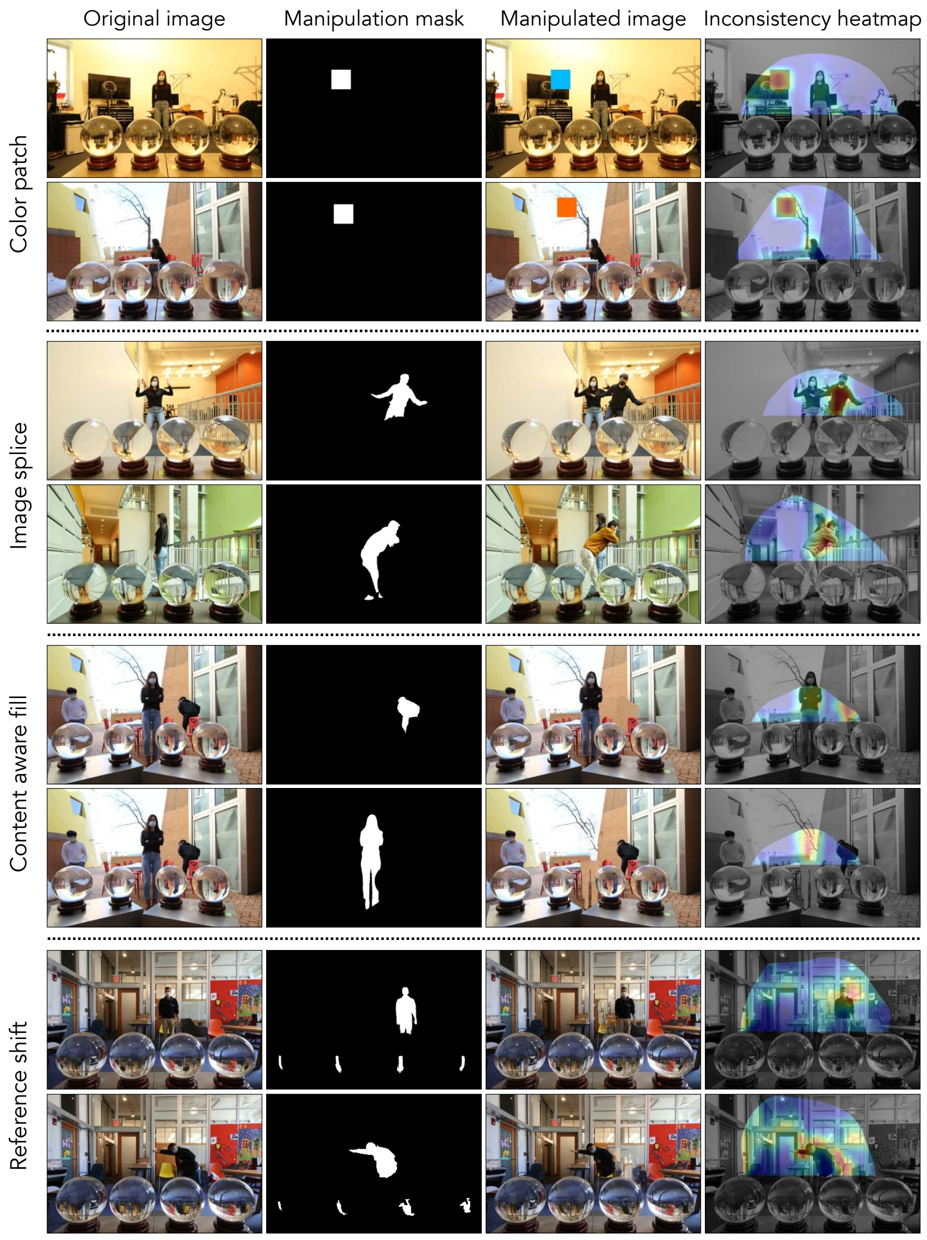

Fig. 5: **Additional results.** Additional detection results for different scenes and different types of manipulations. Our method compares the totem protected region of the scene reconstruction with the manipulated image and shows potential manipulations via the inconsistency heatmap. Note that the scene reconstruction is learned only using the pixels within the totems.

main intact, our method can reliably detect a variety of manipulations (*i.e.* color patches, image splice, CAF).

Joint image and totem manipulations. If the adversary notices and attempts to manipulate the totem views, there are a few different possibilities:

- **Cropping out totems.** In this case, the image is no longer verifiable; only verifiable images are protected.
- **Scrambling totem pixels.** Our method can still reliably reconstruct the scene when small portions of the totem views are manipulated (*e.g.* the reference shift examples). If the resulted reconstruction seems drastically different from the camera view, it implies that large portions of the totem views have been manipulated.
- **Geometric manipulation.** We demonstrate that geometric manipulation of the totem views is detectable through the reference shift example in Fig.5 and Fig.7 in the main paper. The adversary shifts the subject in both camera and totem views to the same reference position in the scene. The resultant manipulation seems reasonable under human inspection but creates geometric inconsistency and makes the reconstructed subject distorted. The reconstruction disagrees with the manipulated camera view, making this manipulation detectable.
- **Color manipulation.** If the adversary changes the color of an object (*i.e.* jacket) in both camera and totem views without tampering with the geometry, the reconstruction will contain the manipulated color and agree with the manipulated camera view. This is a case where our method can fail.

Limitations. Currently, our method reliably detects manipulations of big objects (*i.e.* the entire subject). As research in sparse-view scene reconstruction continues to develop [2], we expect improved reconstruction results with less noise and more semantic details, allowing more detailed manipulations such as smaller objects or facial expressions to be detected. Another limitation is that our detection method is designed to highlight discrepancy between the reconstruction and the camera view, which means it currently does not highlight manipulations in the totem views. This is important to address in future work.

C.2 Frequently asked questions

Who owns/uses totems? The subject owns and sets up their unique totems as a manipulation defense for the digital content captured by anyone or any device. A common confusion is that the photographer carries totems around and is responsible for setting up totems. While there are many active defense methods available to other stakeholders (*e.g.* digital signatures, encryption cameras, etc.), method designed for the subject is under-explored and we hope to inspire more research in this area.

Why not treat totems as cameras and use Structure from Motion (SfM)? SfM and other methods that rely on correspondences will not generalize to totems with complex geometry and distortions.

References

1. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis (2020) 4
2. Niemeyer, M., Barron, J.T., Mildenhall, B., Sajjadi, M.S.M., Geiger, A., Radwan, N.: Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2022) 9