

Efficient Rendering of Heterogeneous Polydisperse Granular Media, Supplemental Material

Thomas Müller^{1,2} Marios Papas¹ Markus Gross^{1,2} Wojciech Jarosz³ Jan Novák¹

¹Disney Research ²ETH Zürich ³Dartmouth College

1 Implementation

In this section we provide additional implementation details.

Heterogeneous Grain Mixtures. We implement the assignment of grain types to bounding spheres as *Mixtures*. Such a *Mixture* implements a function which takes a bounding sphere’s index and position as input arguments and maps these to natural numbers representing grain types. Useful implementations of *Mixtures* include *explicit* mixtures—e.g. from simulation data—which load an explicit mapping from disk, and various procedural mixtures, such as the homogeneous deterministic pseudo-random mixture used by Meng et al. [2015]. Additionally, we implemented a mixture which selects one of two grain types based on Perlin noise [Perlin 2002], or a checkerboard. We show examples of supported mixtures in Figure 1.

Fast Procedural Grain Instantiation. Similarly to Meng et al. [2015], we support procedural instantiation of grains filling a scene-provided watertight bounding mesh. We adopt their tile-based approach, where a pre-computed sphere packing inside a unit cuboid is repeated ad infinitum in a 3D grid. The key difference between our approach and their approach is which bounding spheres are considered to be *inside* the bounding mesh. They require bounding sphere centers to lie within the mesh, whereas we require the entire spheres to be contained in the mesh. Our slightly more strict definition allows us to only intersect those bounding spheres which lie along a ray segment passing through the mesh. Meng et al. [2015], in contrast, resort to explicitly labeling the tiles within the meshes bounding box in a per-scene pre-computation step, marking those tiles in which bounding spheres should be intersected. This can lead to wasted computation before *and* during rendering, since a single tile can contain thousands of bounding spheres, of which many may lie outside of the mesh. For each of these intersected spheres a check has to be performed to truly know whether it lies within the bounding mesh. For both their method and ours a simple way exists for checking whether a given bounding sphere lies within the mesh. Meng et al. [2015] shoot a ray from the center of the bounding sphere, and evaluate whether the ray intersects the mesh from inside or from outside. We traverse a kd-tree of the scene to check whether there are triangles closer to the center of the bounding sphere than the sphere’s radius. Finally, after a bounding sphere which passes the aforementioned checks has been hit, a grain is instantiated and path-traced.

This is the author’s version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in ACM Transactions on Graphics. © 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.
SA ’16 Technical Papers., December 05 - 08, 2016, Macao
ISBN: 978-1-4503-4514-9/16/12
DOI: <http://dx.doi.org/10.1145/2980179.2982429>

PPT and ST Rendering Pseudocode. Algorithms 1 and 2 describe the procedure of path-tracing when hitting a grain’s proxy, and shell tracing in continuous volumes, respectively.

2 Shell Transport Functions

In this section we provide additional material supporting our claims about STFs.

2D Cosine Hemisphere. In Figures 2 and 3 we show slices through the 2D directional STF component of the 4D STFs described by Moon et al. [2007] when computed on homogeneous continuous media derived from snow and dielectric spheres, respectively. As can be seen, the directional components of STF with large radii converge to cosine hemispherical distributions. Small shells generally have a small multiple-scattering component, and thus the worse approximation of the multiple-scattering lobe with a cosine-weighted hemisphere does not play as big a role.

Investigation of Approximations. In Figure 4 we show the effects of our various STF approximations on performance and bias. We provide TTUV and MRSE values for each approximation step we perform, and across a wide range of volume densities.

3 Grain-Scattering Distribution Functions

In this section we provide additional material supporting our claims about GSDFs.

Investigation of Approximation. In Figure 5 through Figure 32 we visualize the approximation error introduced by our GSDFs in the directional domain $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ for various grain types. Figures 9 through 32 follow a common color scale, whereas Figures 5 through 8 have their own color scale in order not to overexpose the visualization.

References

- MENG, J., PAPAS, M., HABEL, R., DACHSBACHER, C., MARSCHNER, S., GROSS, M., AND JAROSZ, W. 2015. Multi-scale modeling and rendering of granular materials. *ACM TOG* 34, 4 (July), 49:1–49:13.
- MOON, J. T., WALTER, B., AND MARSCHNER, S. R. 2007. Rendering discrete random media using precomputed scattering solutions. In *Proc. EGSR*, 231–242.
- PERLIN, K. 2002. Improving noise. In *ACM Transactions on Graphics (TOG)*, vol. 21, ACM, 681–682.

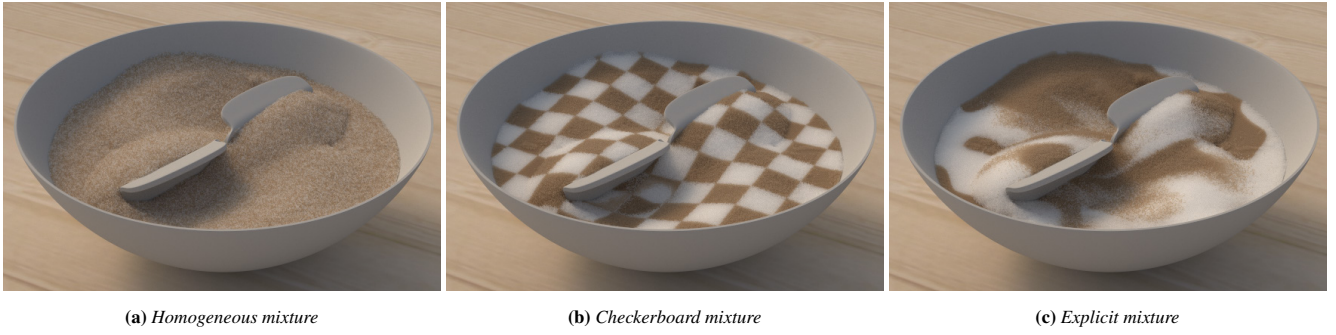


Figure 1: Various supported grain mixtures. Grains can be mixed homogeneously (a), arranged in a procedural 3D checkerboard pattern (b) or distributed explicitly; e.g. governed by a simulation (c).

Algorithm 1 Path Tracing with Proxy Objects

```

function HANDLEPROXYVERTEX( $\mathbf{x}_o, \vec{\omega}_o$ )
  ( $f_g, \mathbf{x}_c$ )  $\leftarrow$  GETPROXYSPHERE( $\mathbf{x}_o$ )
   $\vec{n}_o \leftarrow (\mathbf{x}_c - \mathbf{x}_o) / \|\mathbf{x}_c - \mathbf{x}_o\|$ 
   $\beta_o \leftarrow \cos^{-1}(\vec{\omega}_o \cdot \vec{n}_o)$ 
   $\alpha_g \leftarrow \alpha_g^0(\beta_o) + \alpha_g^+(\beta_o)$ 
   $t \leftarrow \alpha_g t$ 
   $\xi \leftarrow$  sample uniformly from  $[0, 1)$ 
  if  $\xi \leq \alpha_g^0(\beta_o) / \alpha_g$  then
     $\vec{\omega}_i \leftarrow \vec{\omega}_o$ 
     $\mathbf{x}_i \leftarrow \mathbf{x}_o + 2(\vec{\omega}_i \cdot \vec{\omega}_o)\vec{\omega}_o$ 
  else
    ( $\mathbf{x}_i, \vec{\omega}_i$ )  $\leftarrow$  sample from  $p_g(\cdot, \cdot | \beta_o)$ 
     $L_i \leftarrow L_i + t \cdot \text{DIRECTLIGHT}(\mathbf{x}_i, p_g^{\vec{\omega}_i}(\cdot | \beta_o))$ 
  end if
end function

```

- \triangleright Obtain GSDF f_g and grain center \mathbf{x}_c
- \triangleright Compute proxy normal
- \triangleright Compute inclination angle
- \triangleright Compute grain albedo conditional on β_o
- \triangleright Adjust path throughput
- \triangleright Randomly choose scattering type
- \triangleright No scattering
- \triangleright Pass straight through bounding sphere
- \triangleright Scattering
- \triangleright Importance sample scattering
- \triangleright Accumulate direct lighting

Algorithm 2 Path Tracing with Shell Transport Functions

```

1: function HANDLESHELLVERTEX( $\mathbf{x}_c, \vec{\omega}_c$ )
2:   ( $f_s, r$ )  $\leftarrow$  GETLARGESTFITTINGSHELL( $\mathbf{x}_c$ )
3:   if  $r < r_1$  then
4:     USEVPT( $\mathbf{x}_c, \vec{\omega}_c$ )
5:   else
6:      $\alpha_s \leftarrow \alpha_s^0(r) + \alpha_s^1(\alpha, g, r) + \alpha_s^m(\alpha, g, r)$ 
7:      $t \leftarrow \alpha_s t$ 
8:      $\xi \leftarrow$  sample uniformly from  $[0, 1)$ 
9:     if  $\xi < \alpha_s^0(r) / \alpha_s$  then
10:       $\mathbf{x}_s \leftarrow \mathbf{x}_c + r\vec{\omega}_c$ 
11:       $\vec{\omega}_s \leftarrow \vec{\omega}_c$ 
12:     else if  $\alpha_s^0(r) / \alpha_s \leq \xi < \alpha_s^1(\alpha, g, r) / \alpha_s$  then
13:       $d_1 \leftarrow$  sample uniformly from  $[0, 1)$ 
14:       $\mathbf{x} \leftarrow \mathbf{x}_c + d_1 r\vec{\omega}_c$ 
15:       $f_p \leftarrow$  GETMEDIUMPHASE( $\mathbf{x}$ )
16:       $\vec{\omega}_s \leftarrow$  sample from  $f_p(\cdot)$ 
17:       $\theta_s \leftarrow \cos^{-1}(\vec{\omega}_c \cdot \vec{\omega}_s)$ 
18:       $d_2 \leftarrow r\sqrt{1 - d_1^2 \sin^2(\theta_s) / r^2} - d_1 \cos \theta_s$ 
19:       $\mathbf{x}_s \leftarrow \mathbf{x} + d_2 \vec{\omega}_s$ 
20:       $L_i \leftarrow L_i + t \cdot \text{DIRECTLIGHT}(\mathbf{x}, f_p(\cdot))$ 
21:     else
22:      ( $\mathbf{x}_s, \vec{\omega}_s$ )  $\leftarrow$  sample from  $p_s^m(\cdot, \cdot | \alpha, g, r)$ 
23:       $L_i \leftarrow L_i + t \cdot \text{DIRECTLIGHT}(\mathbf{x}_s, p_s^{m, \vec{\omega}_s}(\cdot | \alpha, g, r, \theta))$ 
24:     end if
25:   end if
26: end function

```

- \triangleright There is no shell which fits
- \triangleright Fall back to VPT
- \triangleright Compute shell albedo conditional on α, g, r
- \triangleright Update path throughput
- \triangleright Randomly choose scattering type
- \triangleright No scattering
- \triangleright Pass through the shell
- \triangleright Single scattering
- \triangleright Sample free-flight distance
- \triangleright Compute single-scattering location
- \triangleright Sample medium phase function
- \triangleright Compute scattering angle
- \triangleright Compute distance to shell surface
- \triangleright Compute outgoing location on shell surface
- \triangleright Compute direct light
- \triangleright Multiple scattering
- \triangleright Importance sample multiple scattering from tabulation
- \triangleright Accumulate direct light

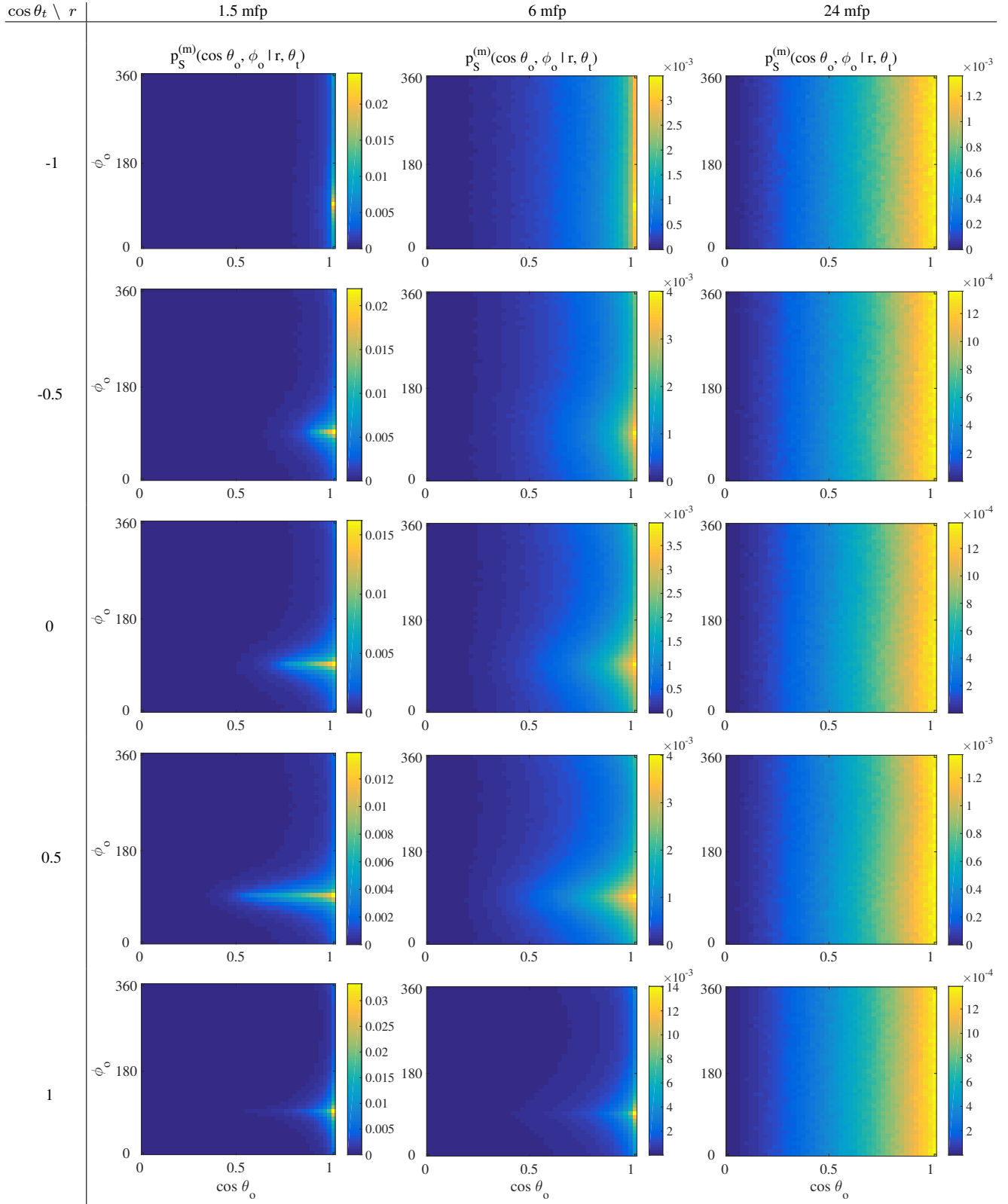


Figure 2: Directional multiple scattering component of the shell transport function corresponding to snow. This figure visualizes the dependence of the shape of the directional component on the shell's radius r (horizontal) and the teleportation angle θ_t (vertical). As r increases, the shape of the directional component approaches a cosine hemisphere.

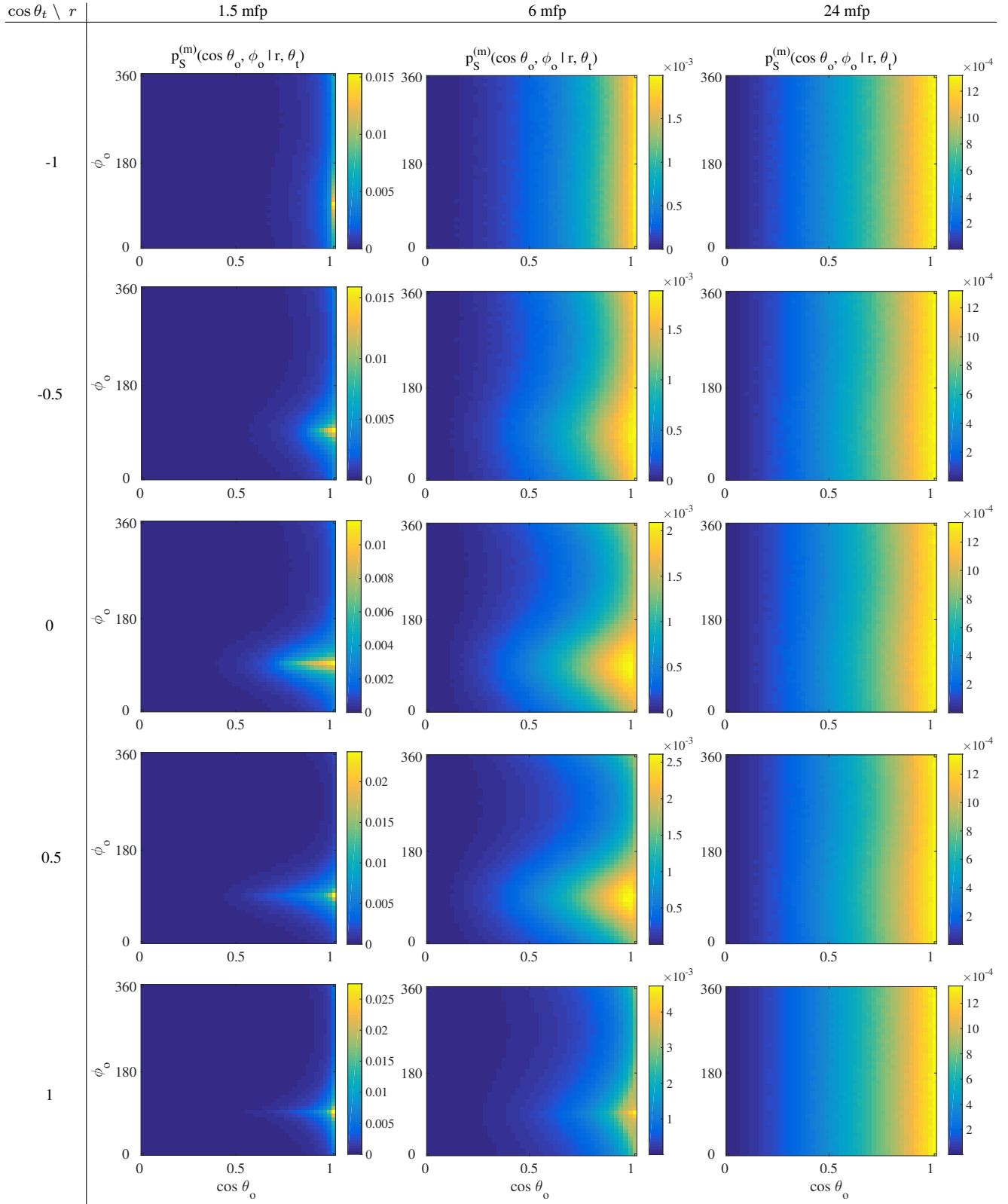


Figure 3: Directional multiple scattering component of the shell transport function corresponding to dielectric spheres. This figure visualizes the dependence of the shape of the directional component on the shell's radius r (horizontal) and the teleportation angle θ_t (vertical). As r increases, the shape of the directional component approaches a cosine hemisphere.

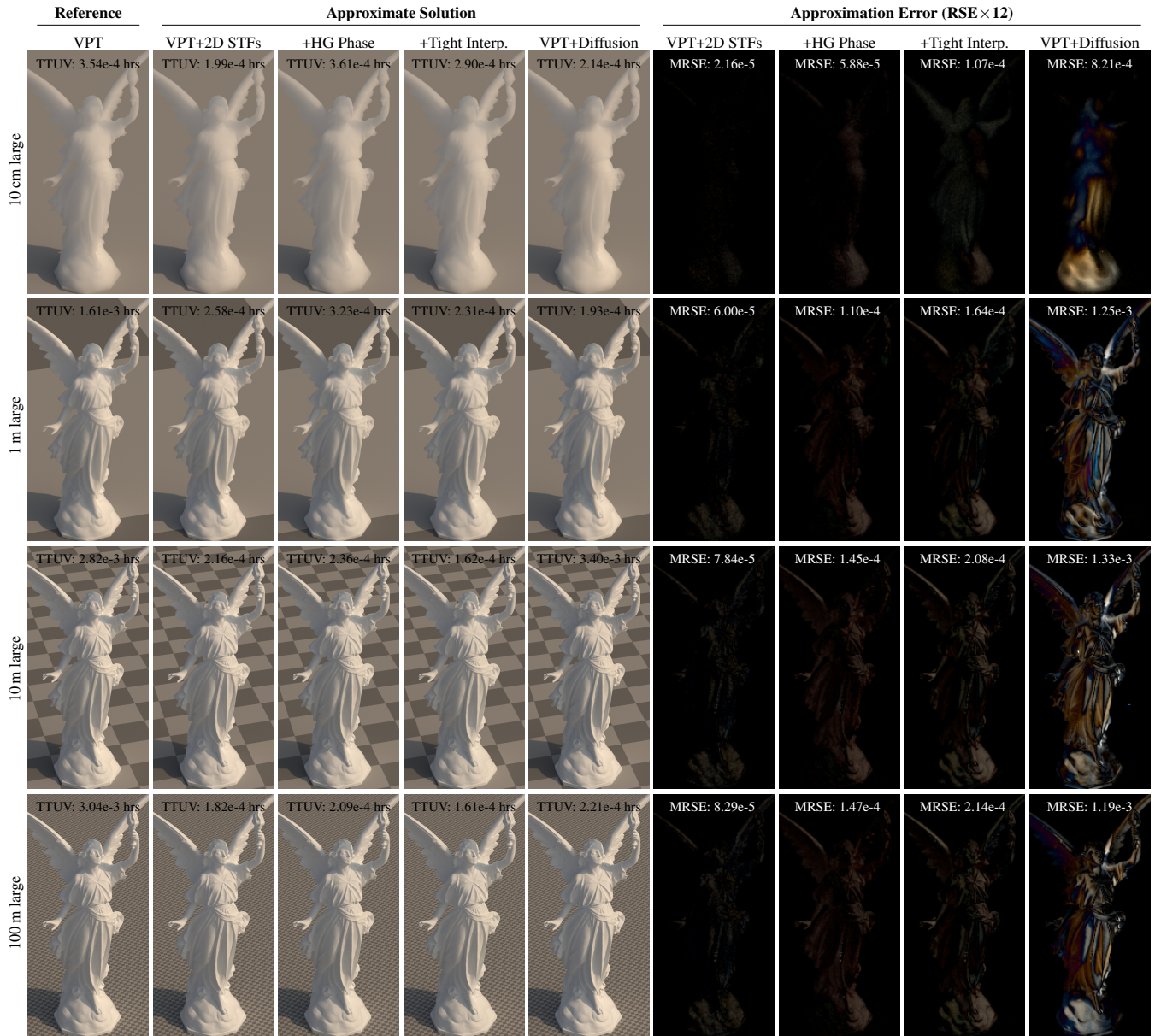


Figure 4: We demonstrate the error introduced by each of the approximations we apply to STFs in the LUCY scene, which contains the Lucy statue filled with a homogeneous continuous participating medium derived from densely packed snow grains. The first column shows volumetrically path traced (VPT) reference images, which can be matched perfectly by the 4D STF in the case of homogeneous continuous media. The following three columns demonstrate the visual impact of our approximations: 2D STFs approximate $p_s^{m, \vec{\omega}}$ as a cosine-weighted hemispherical distribution, using HG phase function significantly reduces the dimensionality of continuous-volume-appearance space to permit dense tabulation, and interpolation between tabulated STFs allows fitting STF tightly into the mesh. The last column shows the diffusion approximation employed by Meng et al. [2015], which yields higher error than all our approximations combined.

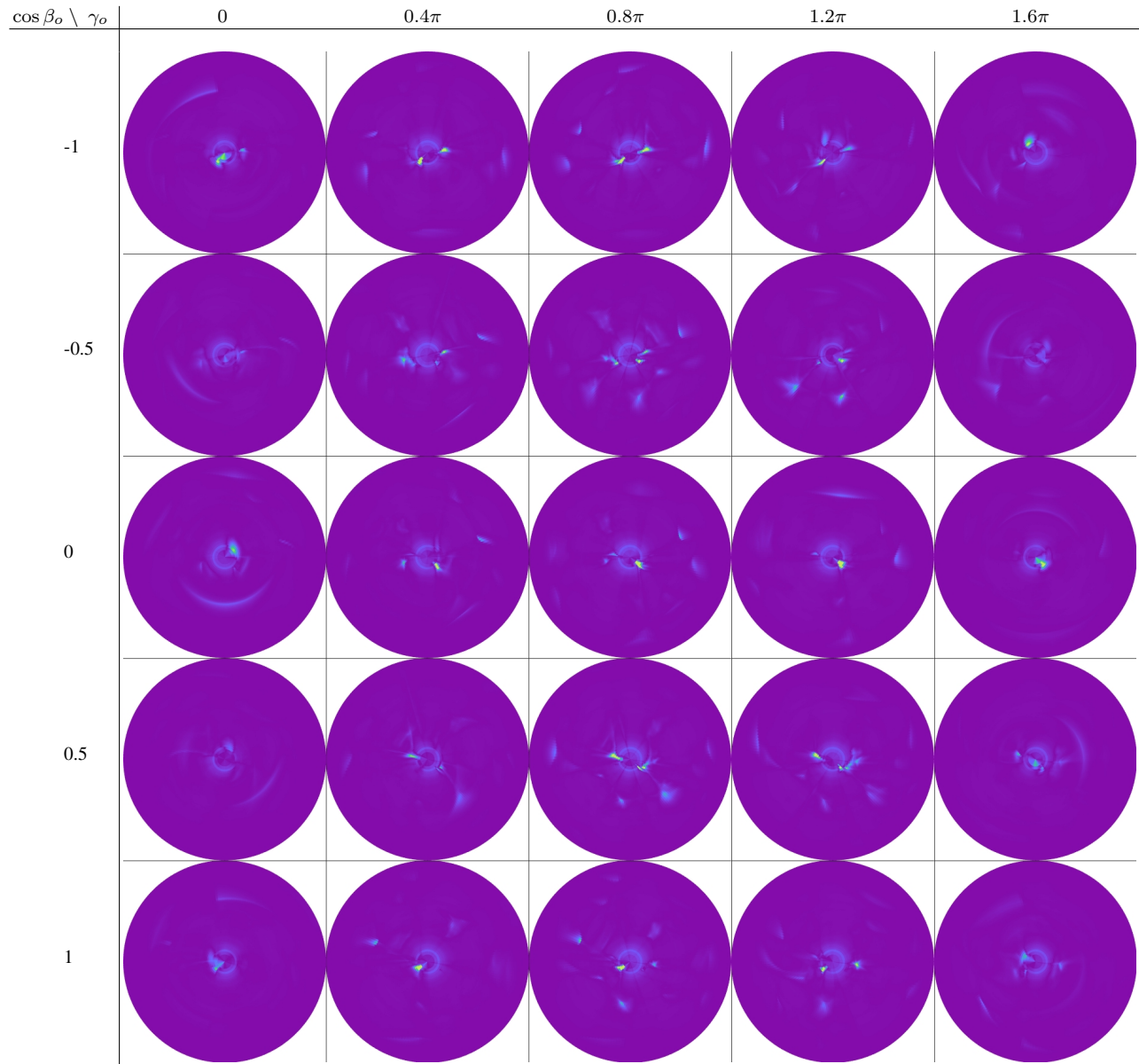


Figure 5: Front-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on sugar grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

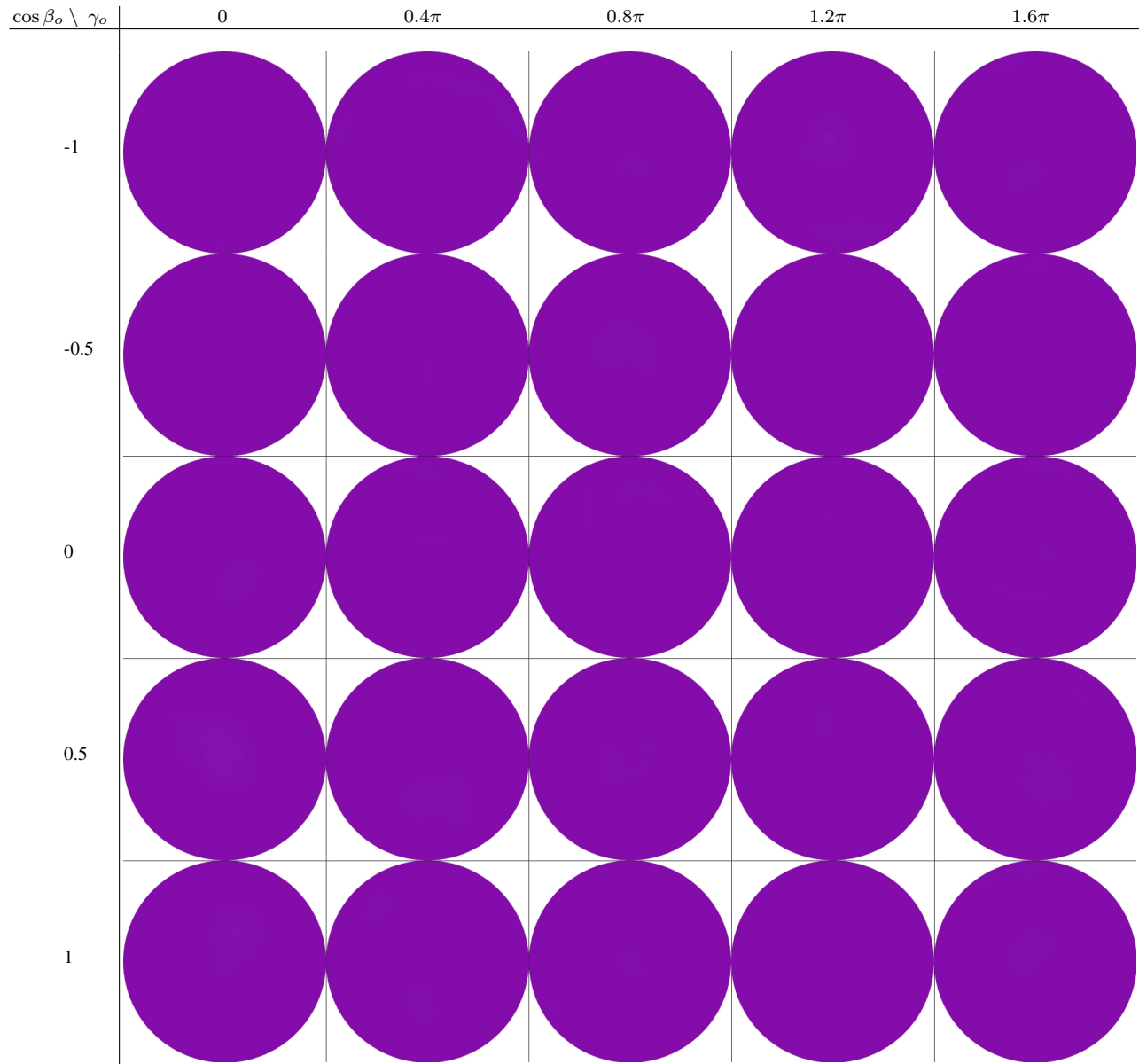


Figure 6: Back-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on sugar grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

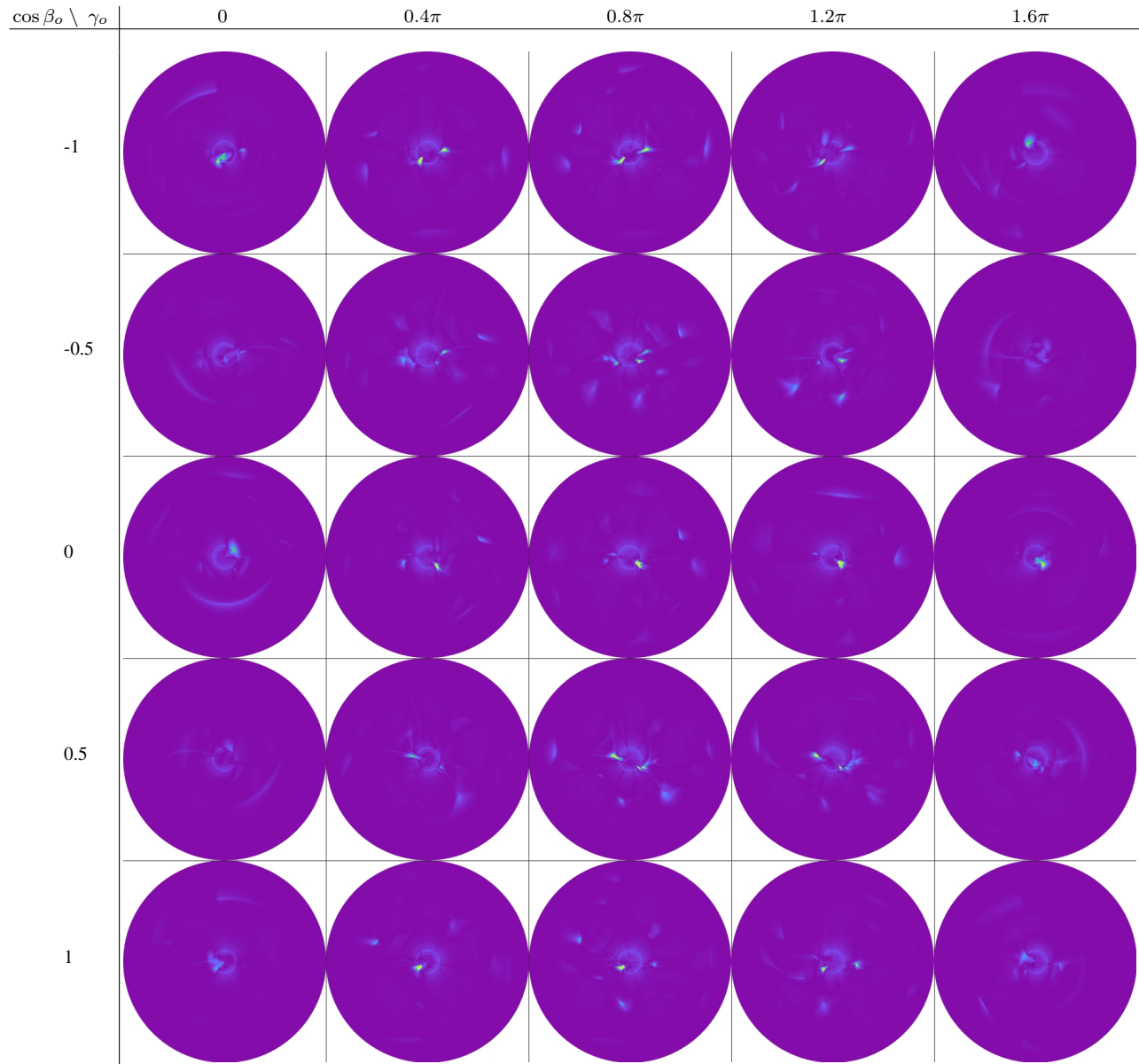


Figure 7: Front-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on brown sugar grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

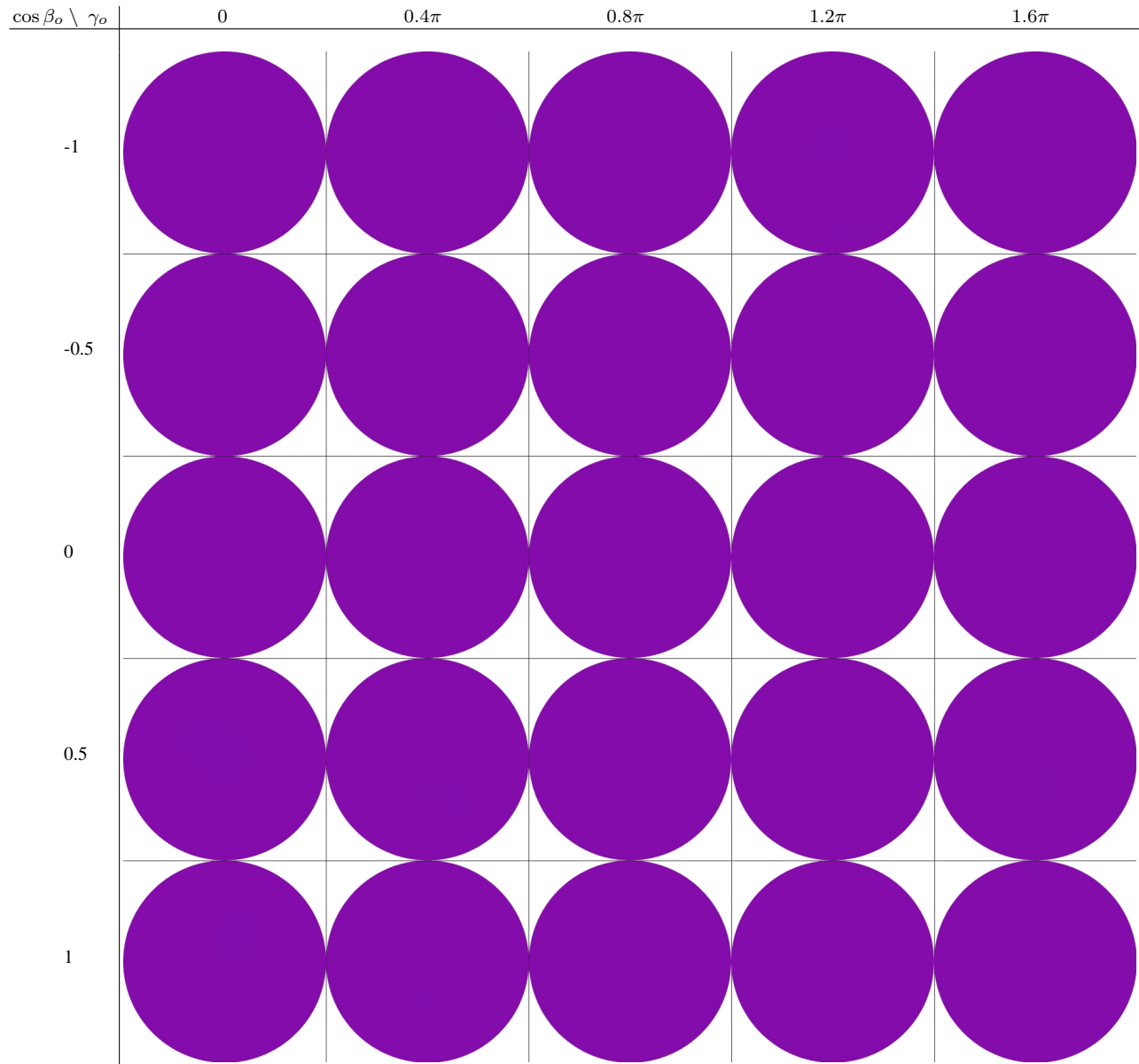


Figure 8: Back-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on brown sugar grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

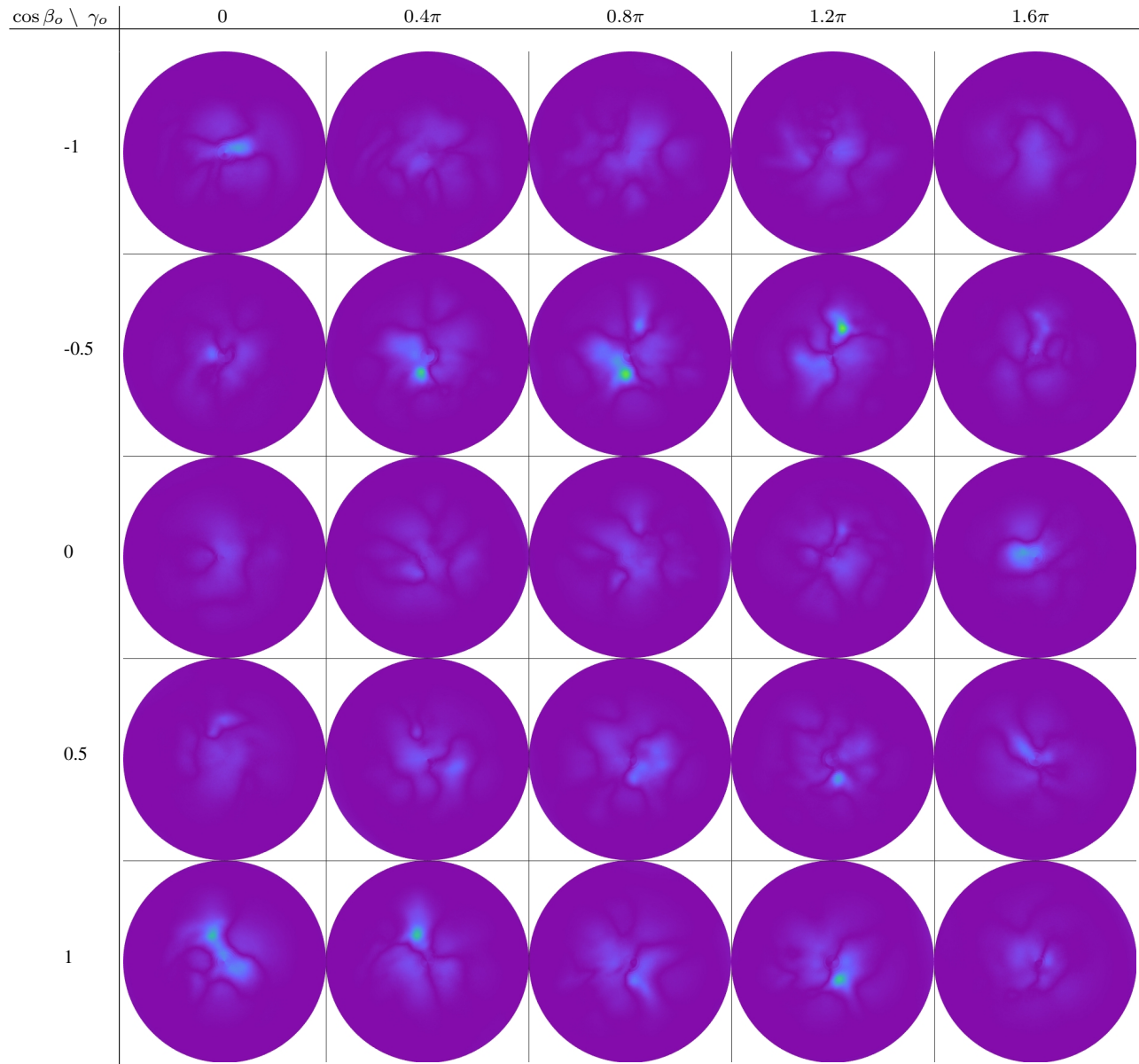


Figure 9: Front-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on ice grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

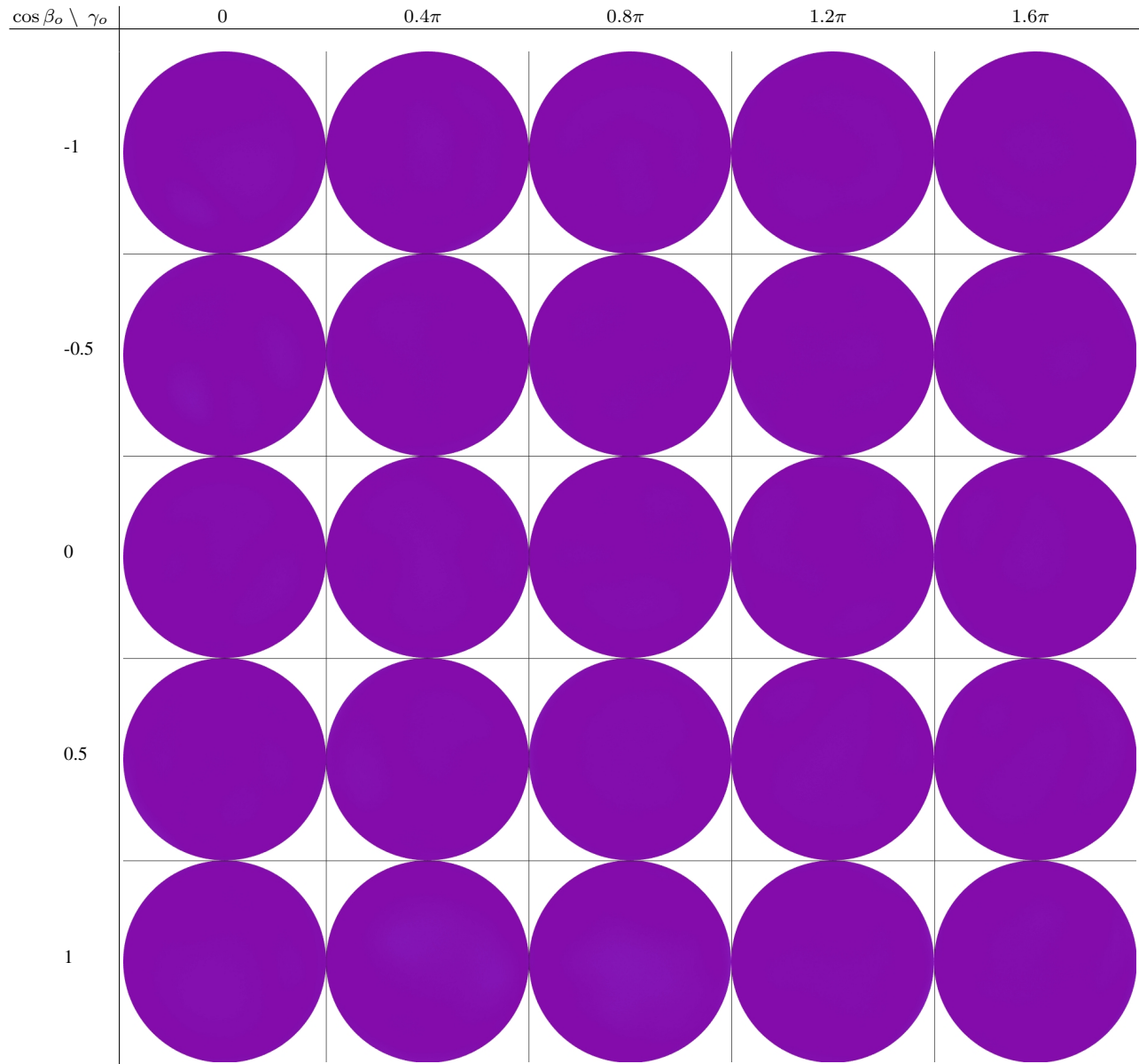


Figure 10: Back-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on ice grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

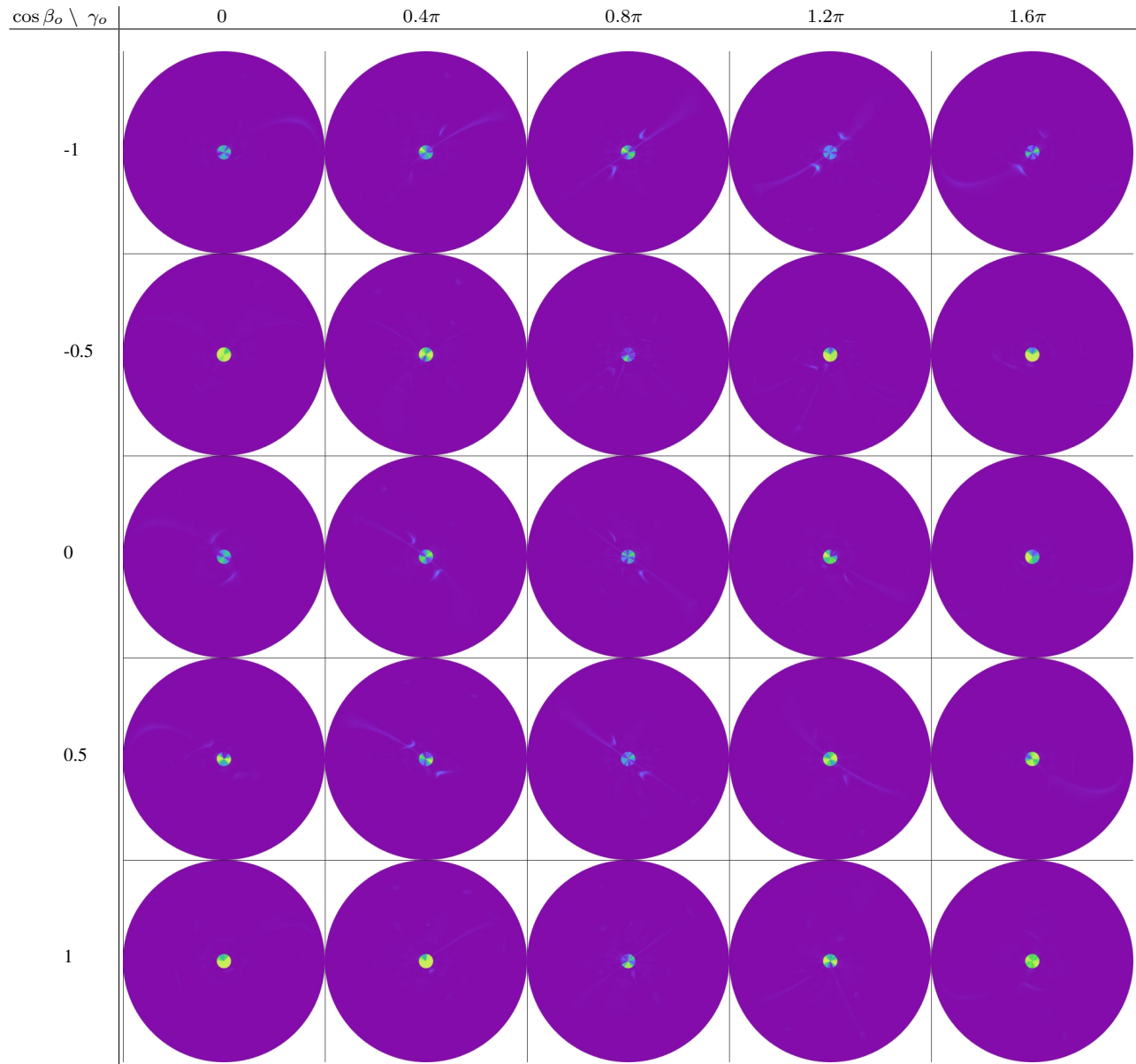


Figure 11: Front-facing directional GSDF approximation error $e_{fg}(\vec{\omega}_i, \vec{\omega}_o)$ on snowflake grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

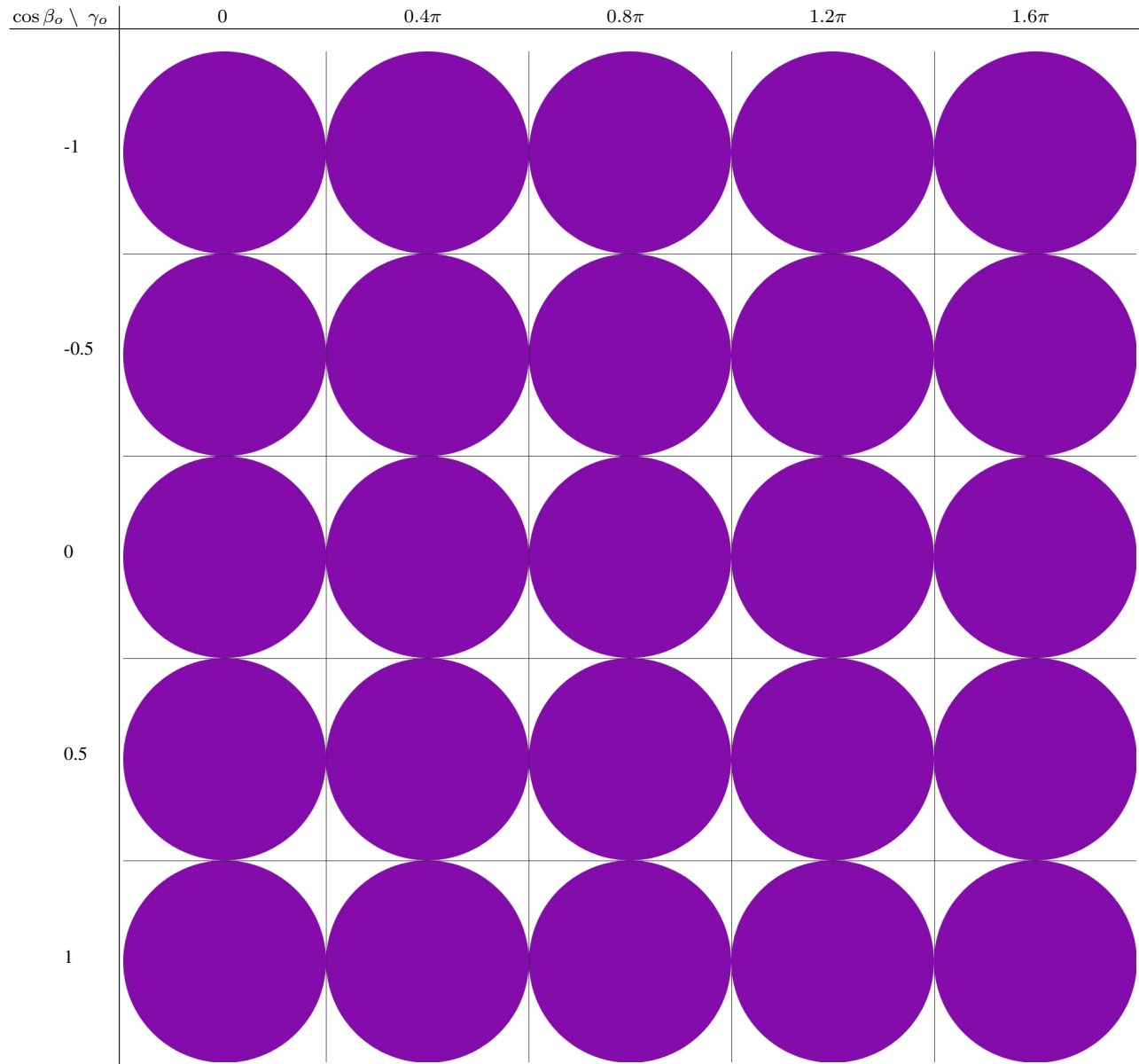


Figure 12: Back-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on snowflake grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

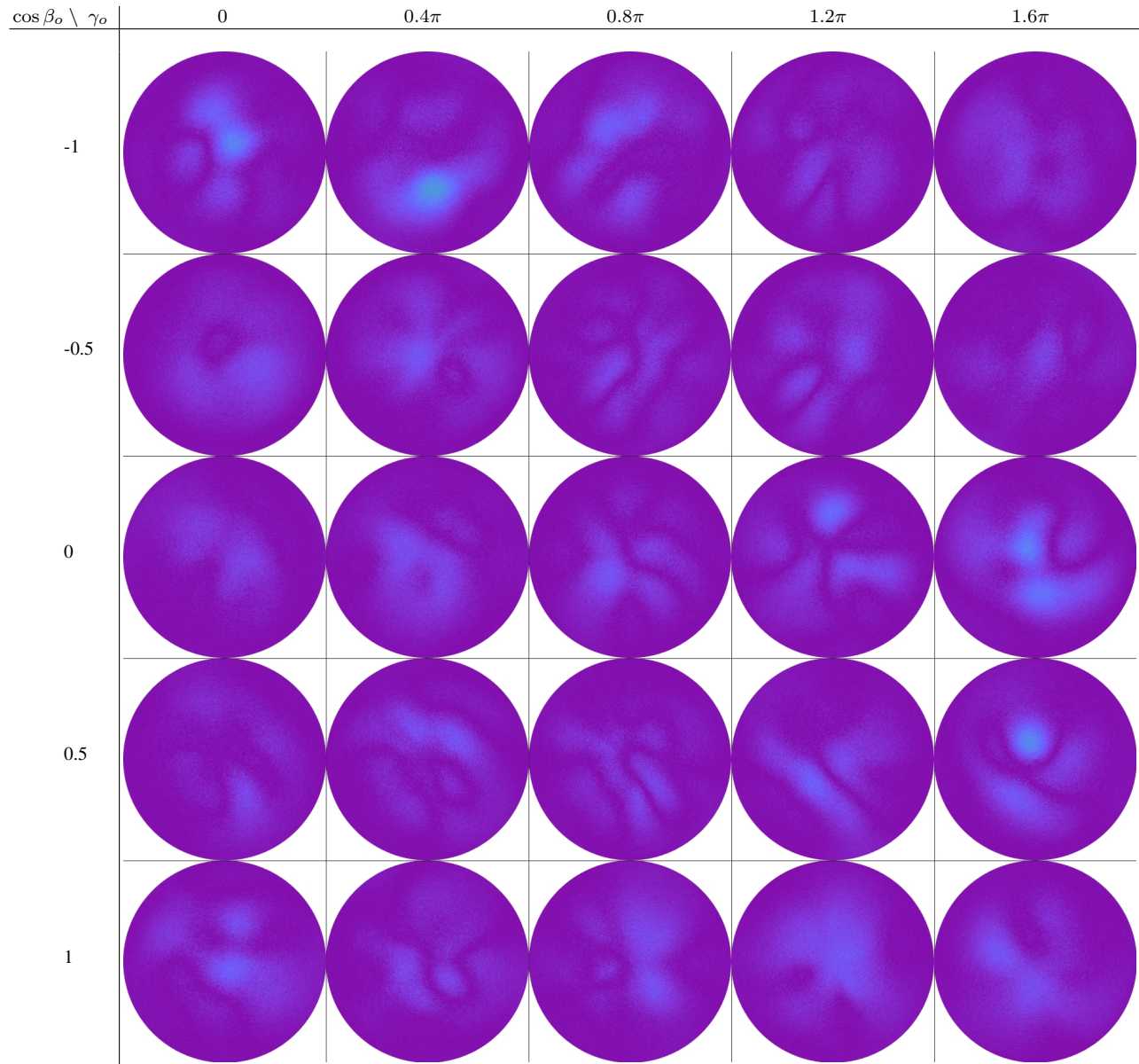


Figure 13: Front-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on white sand grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

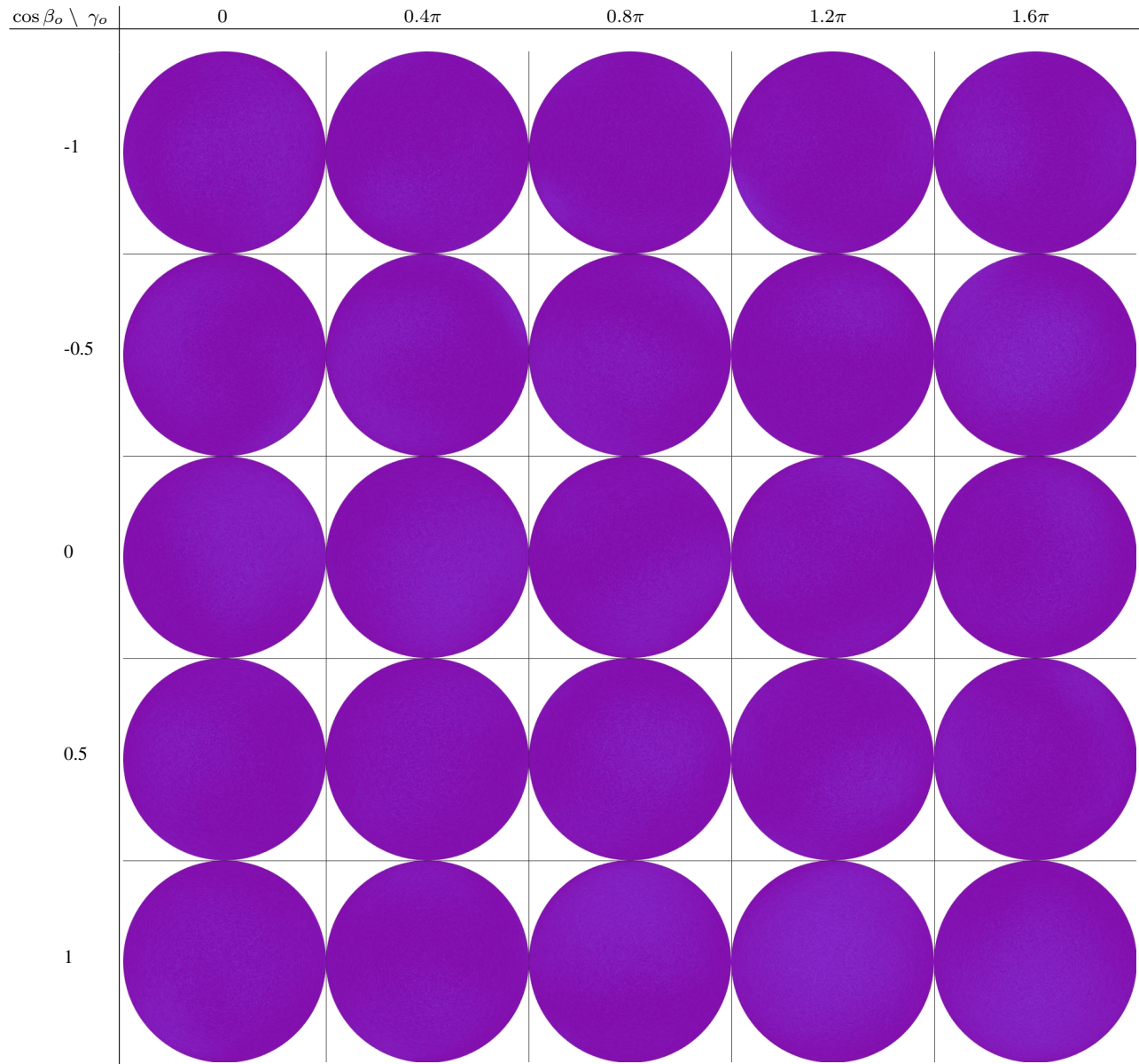


Figure 14: Back-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on white sand grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

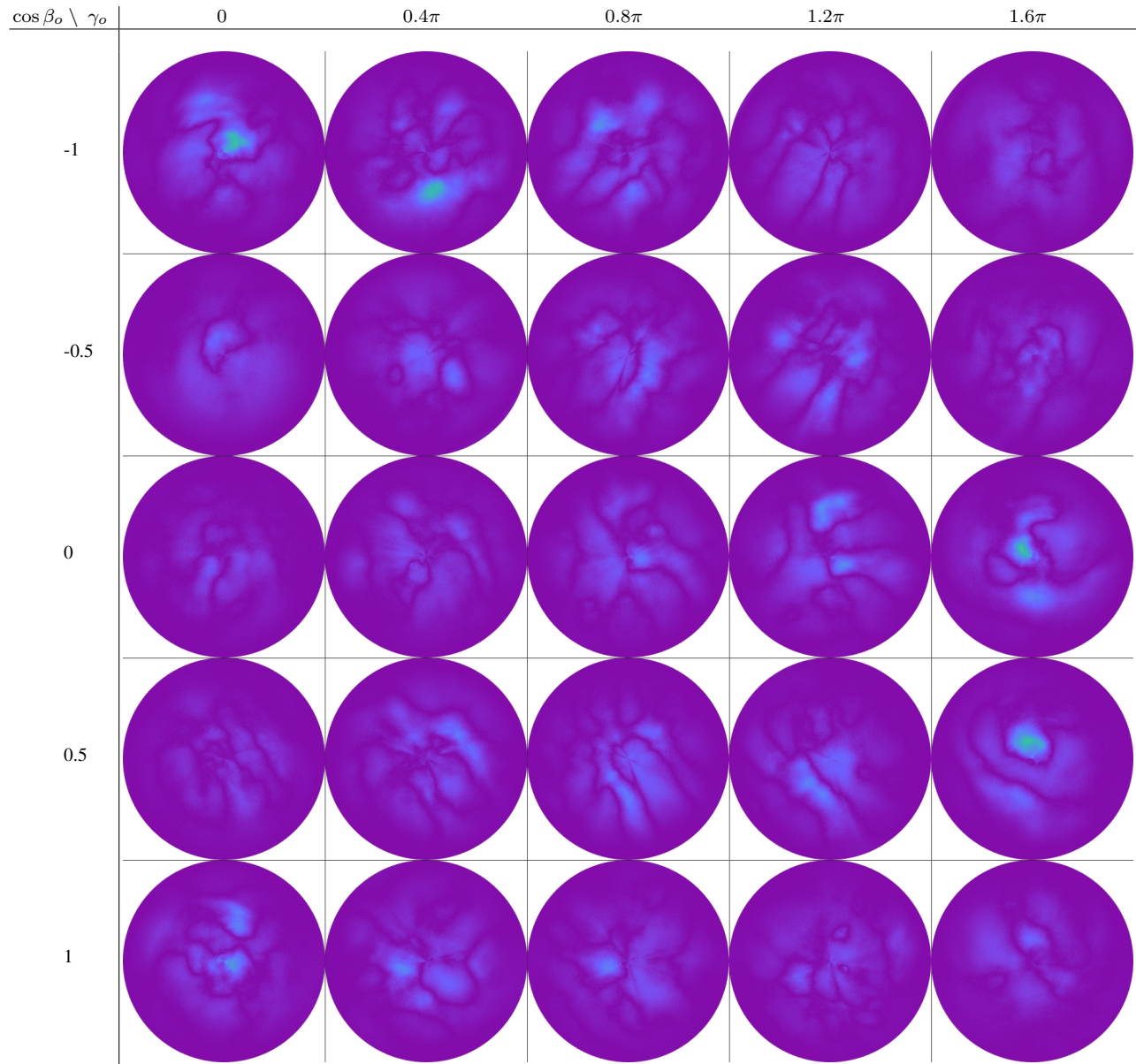


Figure 15: Front-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on yellow sand grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

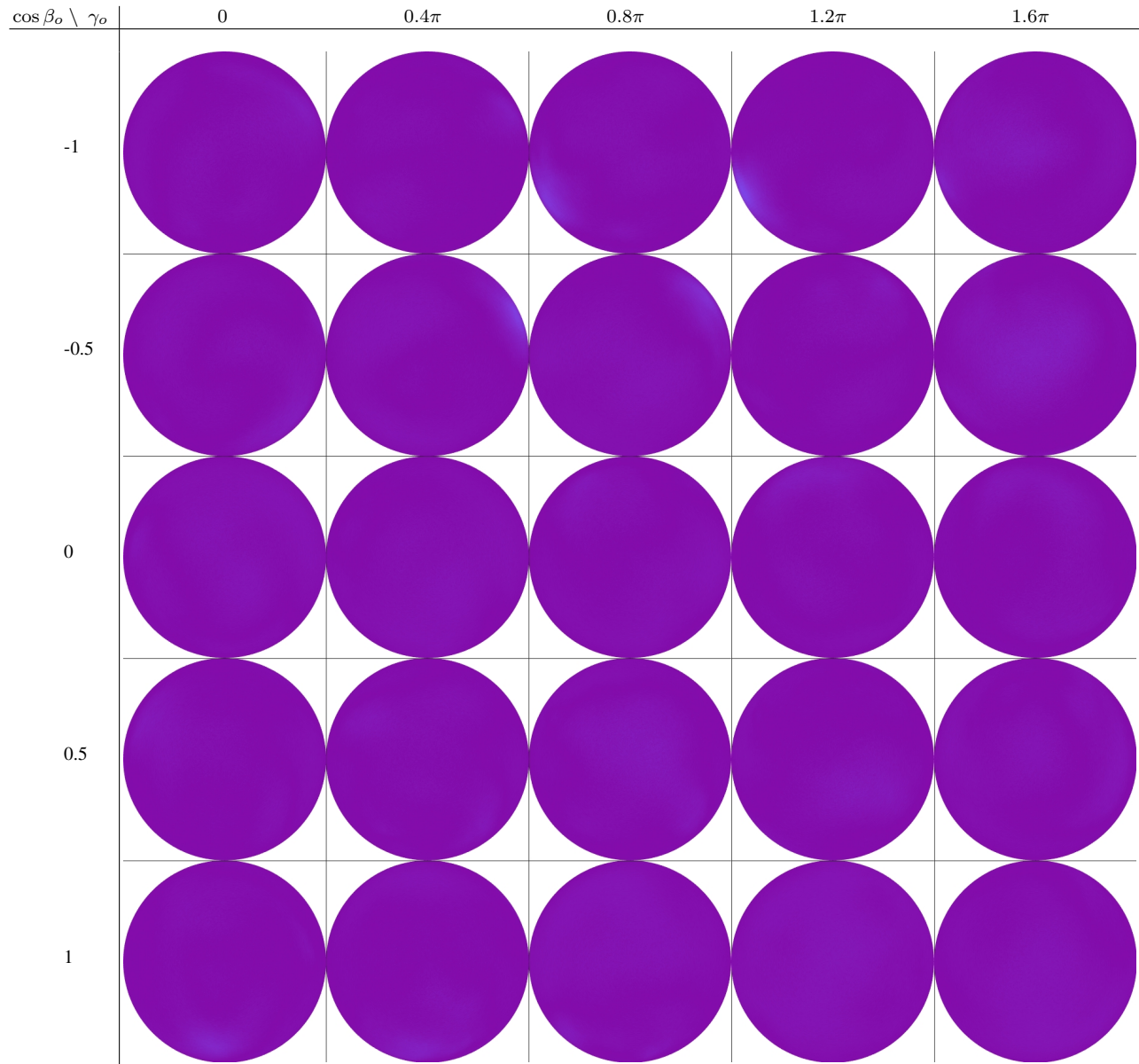


Figure 16: Back-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on yellow sand grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

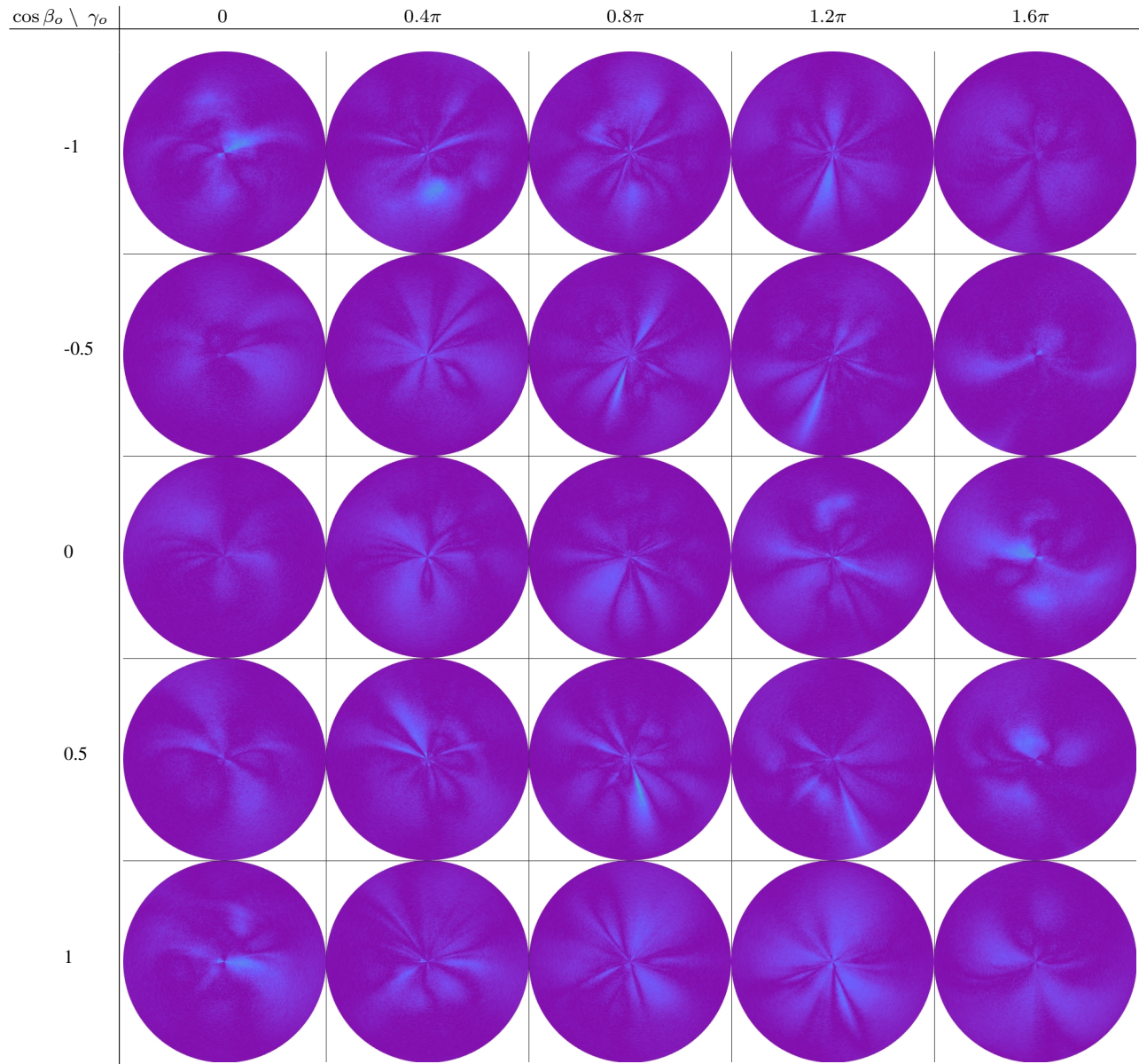


Figure 17: Front-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on brown sand grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

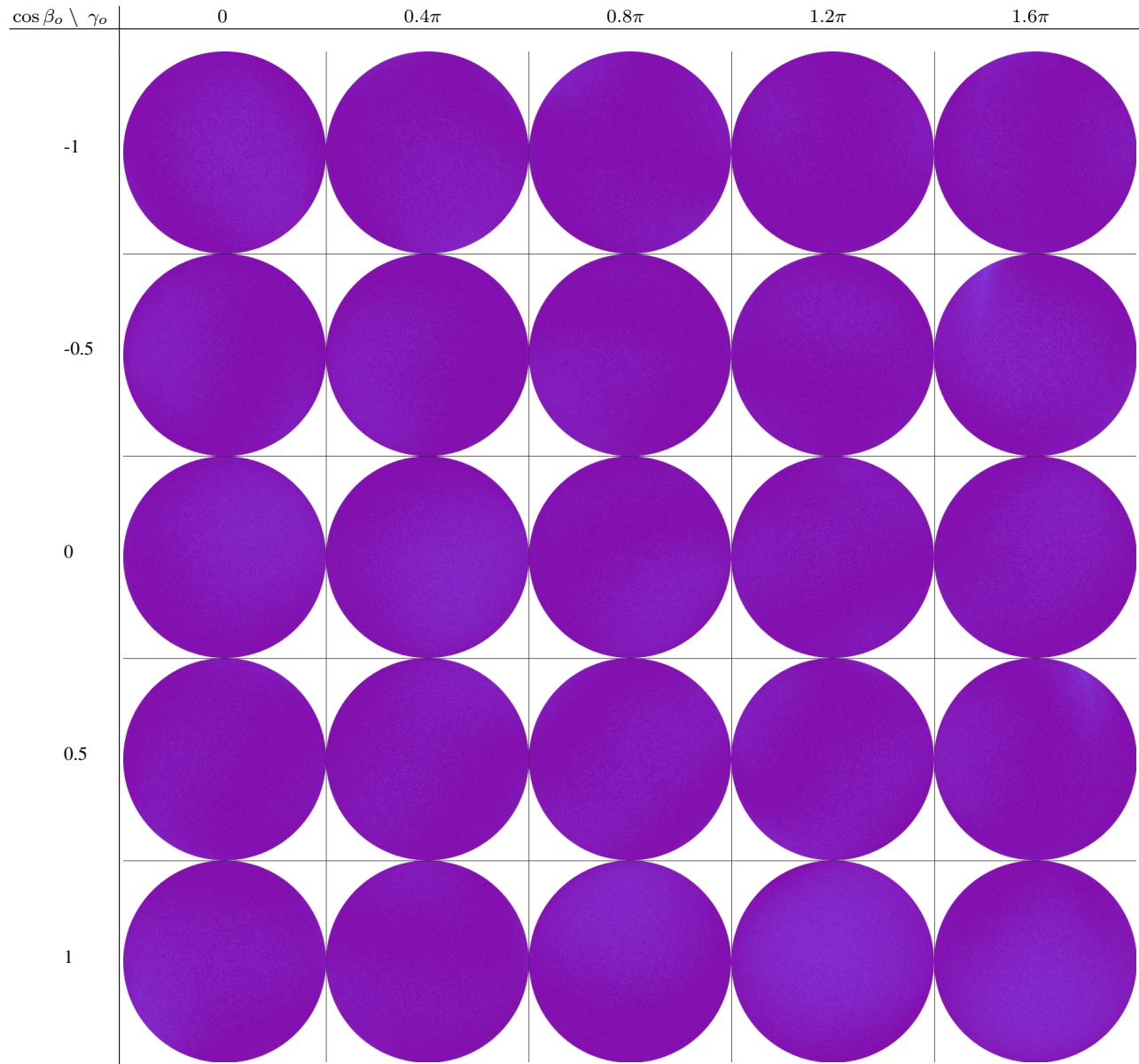


Figure 18: Back-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on brown sand grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

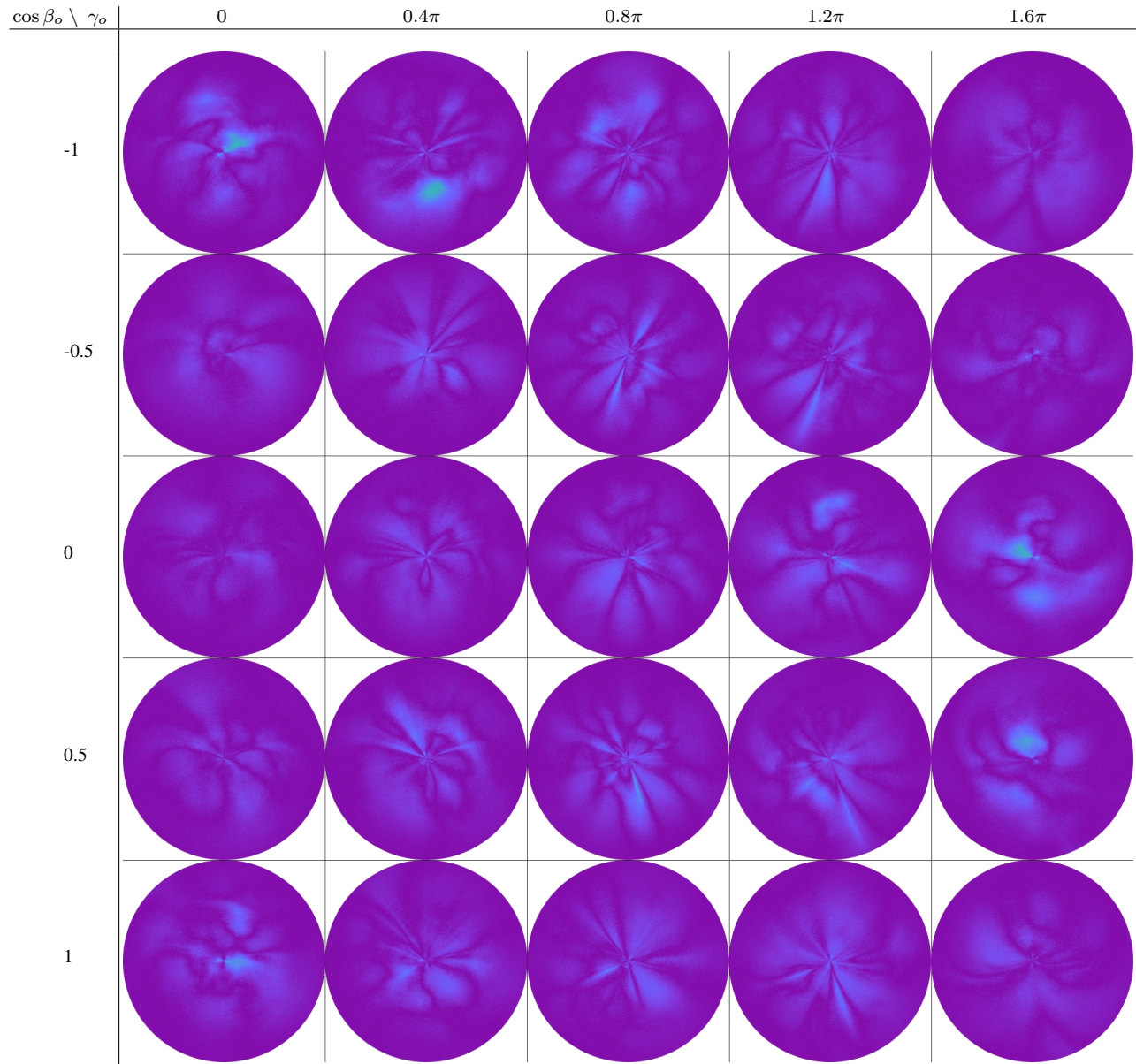


Figure 19: Front-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on dark brown sand grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

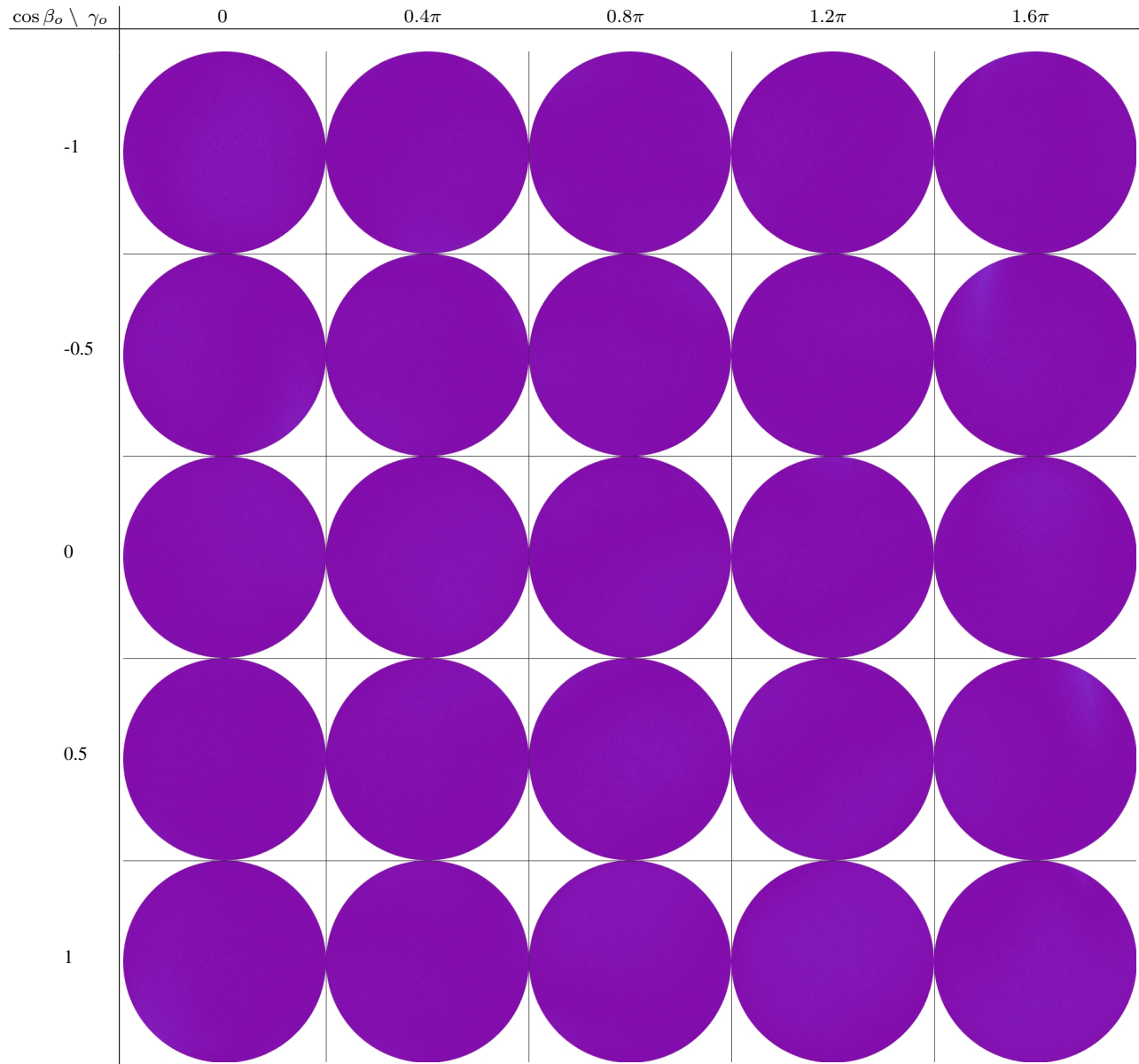


Figure 20: Back-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on dark brown sand grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

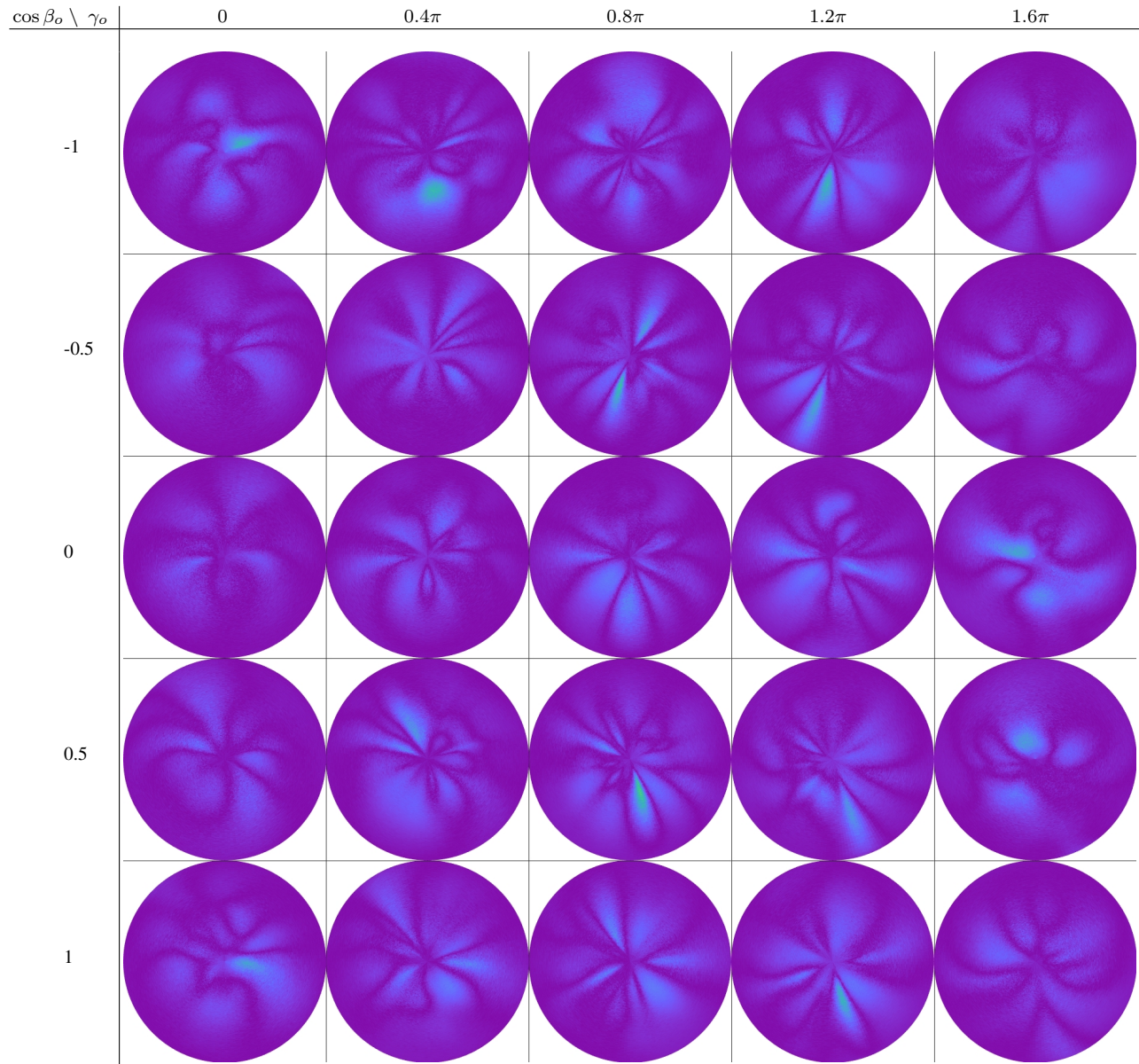


Figure 21: Front-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on black sand grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

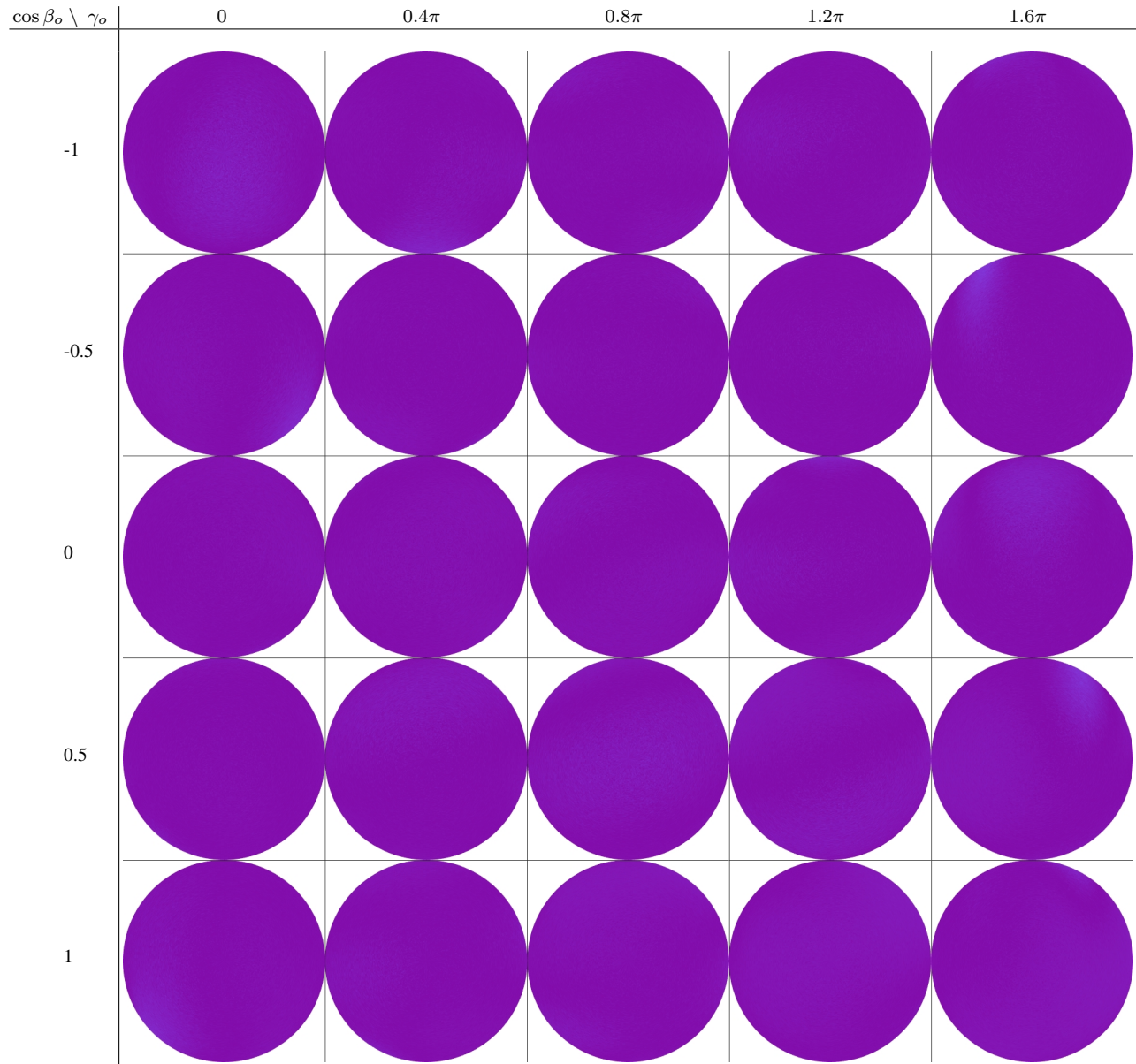


Figure 22: Back-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on black sand grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

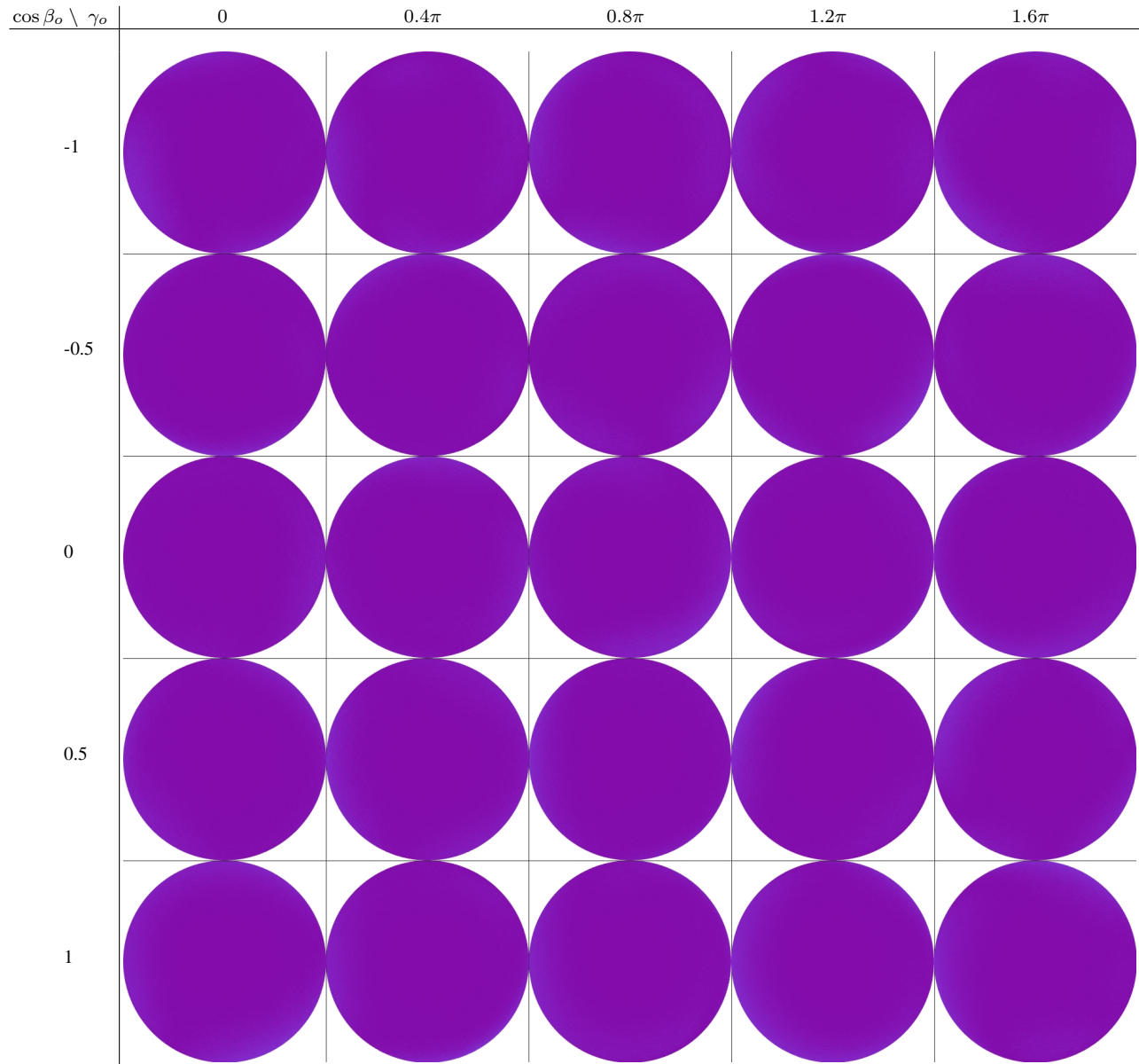


Figure 23: Front-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on flour grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

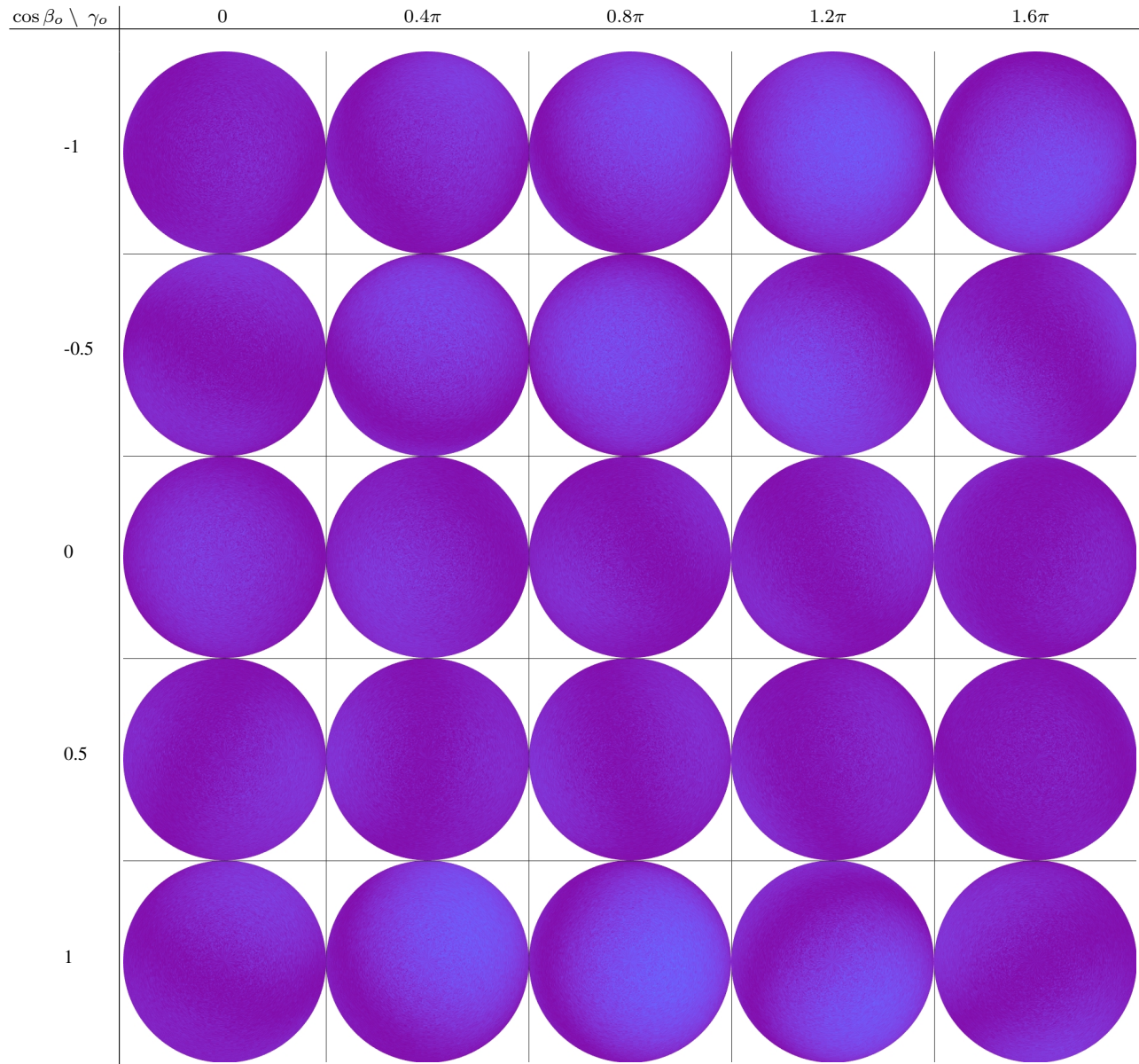


Figure 24: Back-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on flour grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

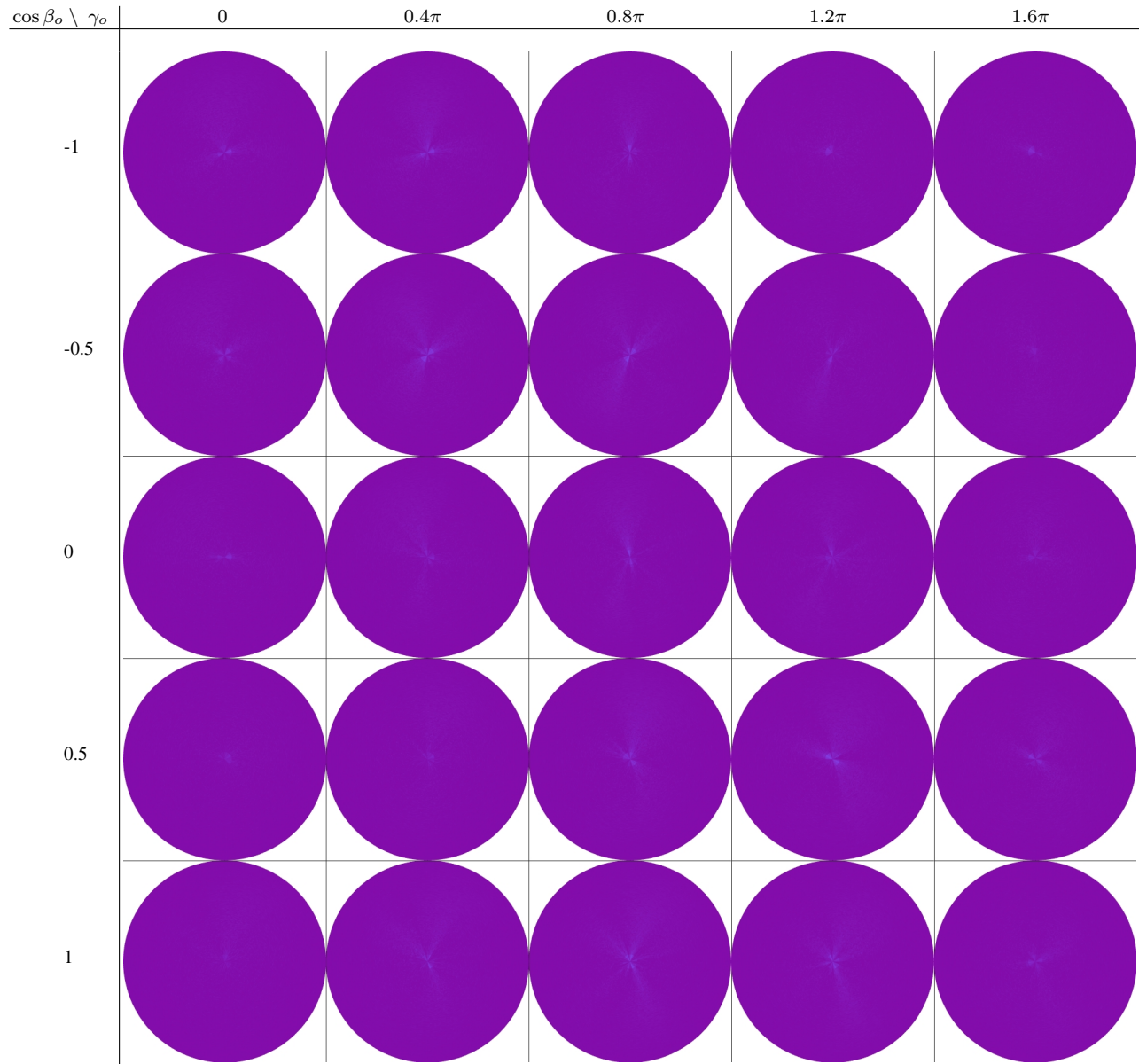


Figure 25: Front-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on hairball grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

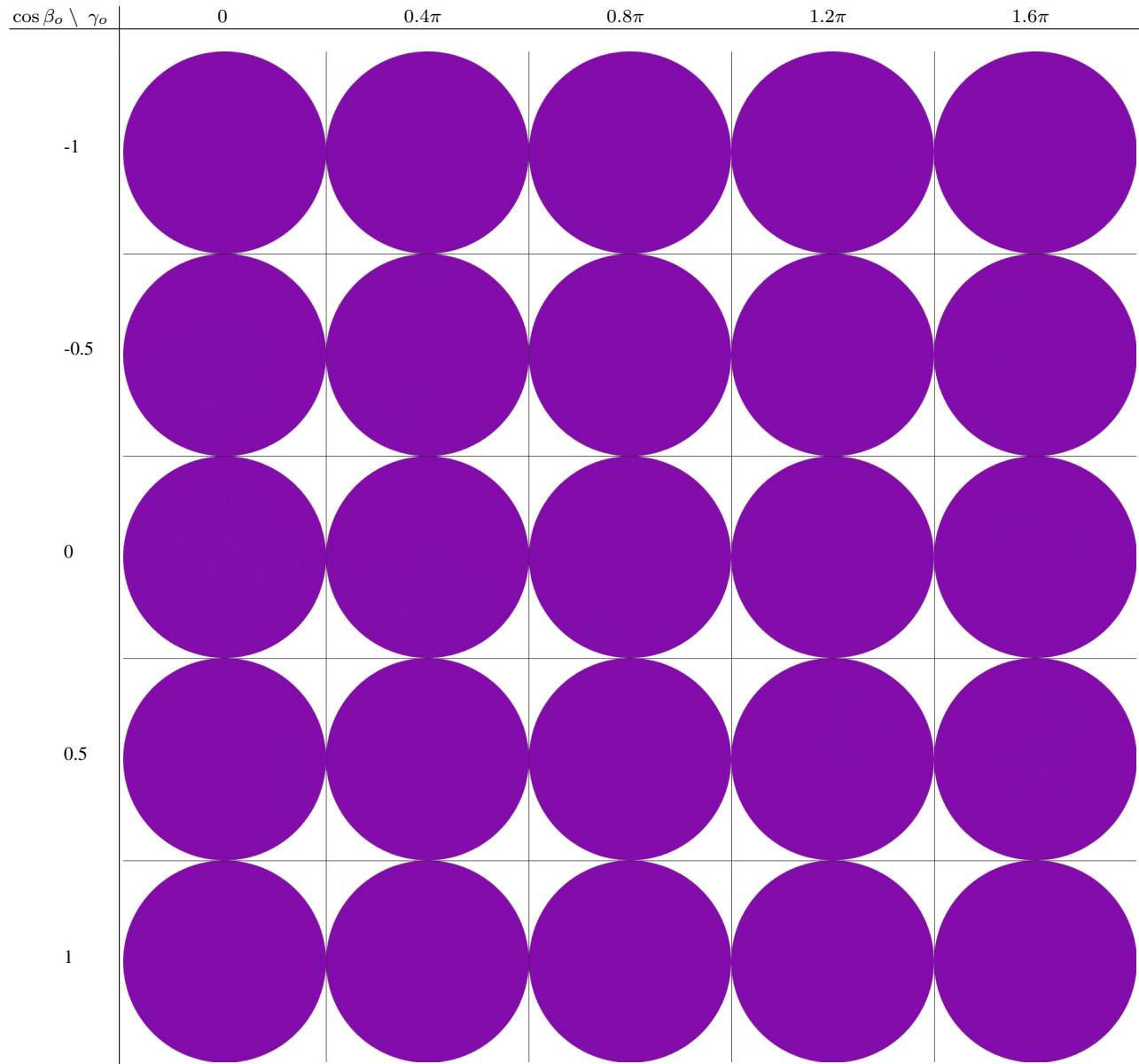


Figure 26: Back-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on hairball grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

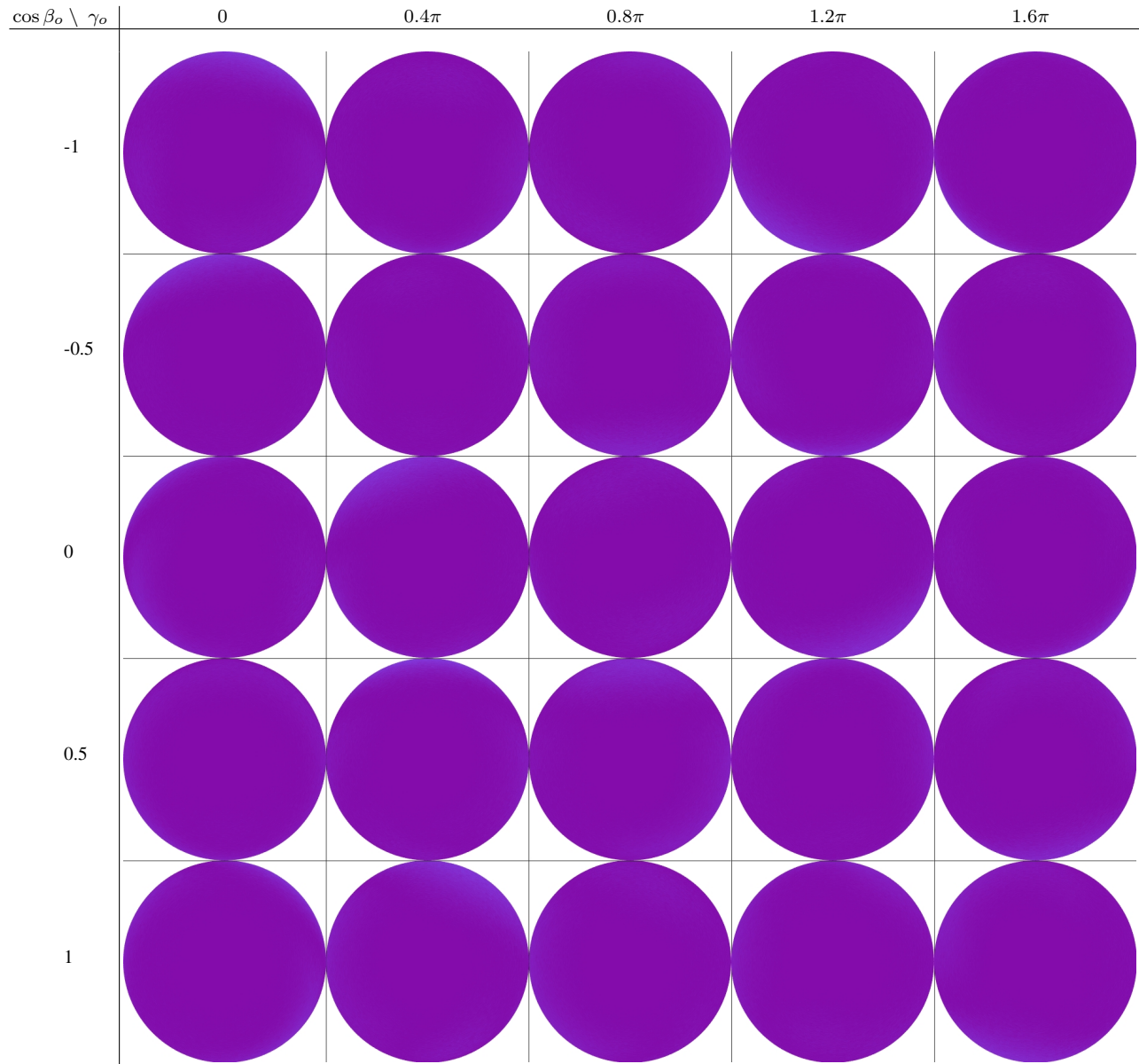


Figure 27: Front-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on cinnamon grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

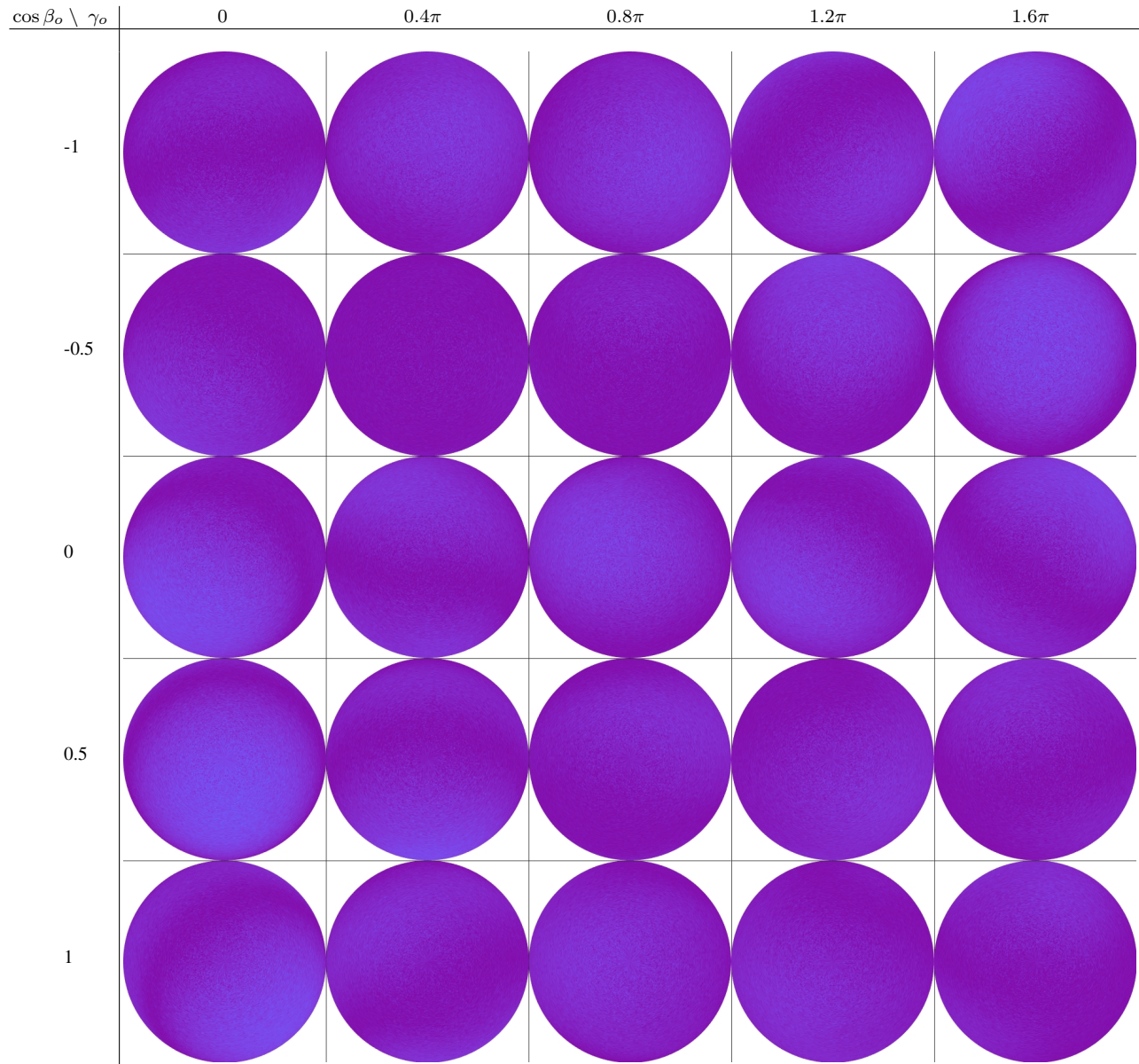


Figure 28: Back-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on cinnamon grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

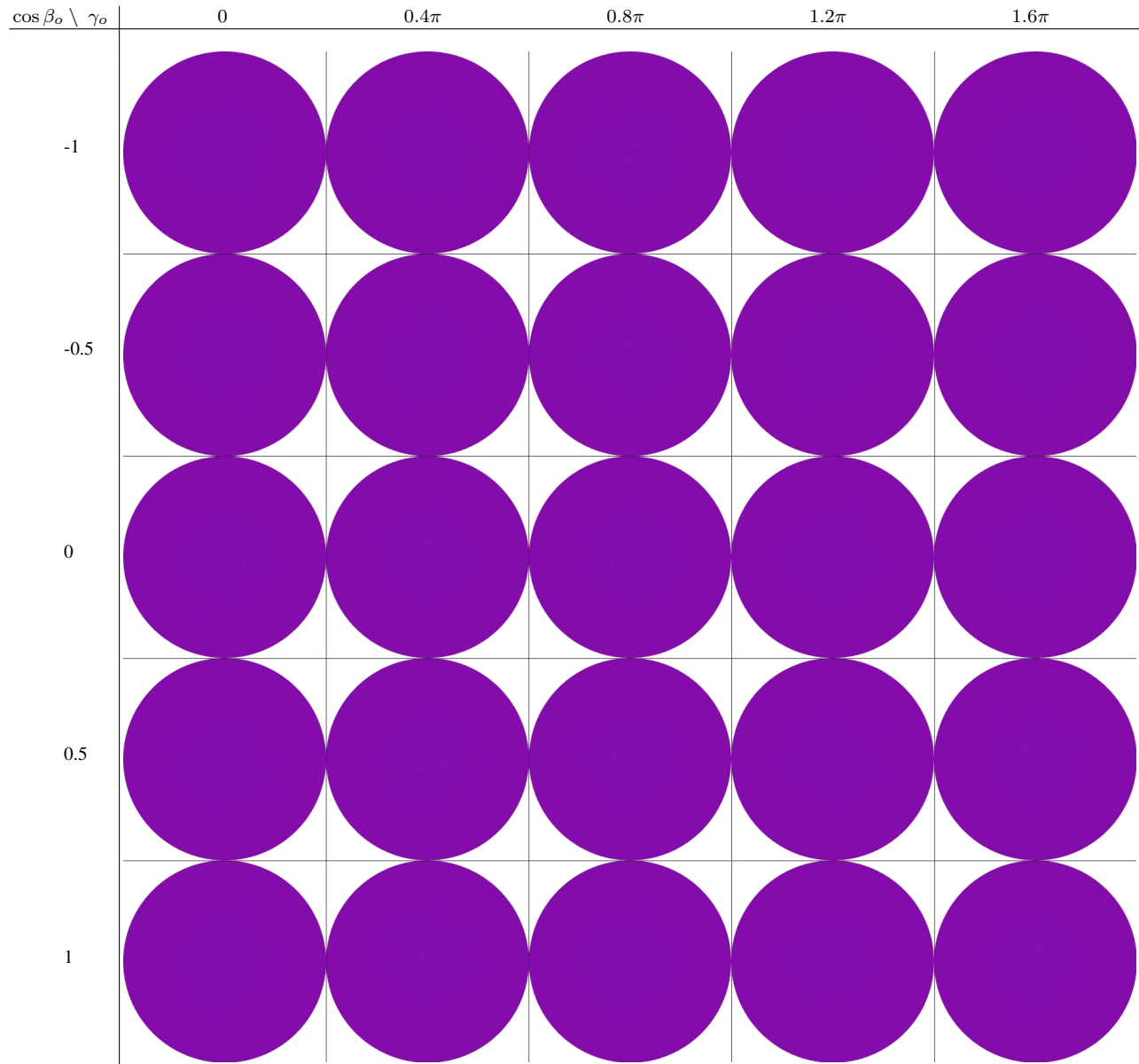


Figure 29: Front-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on dielectric sphere grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

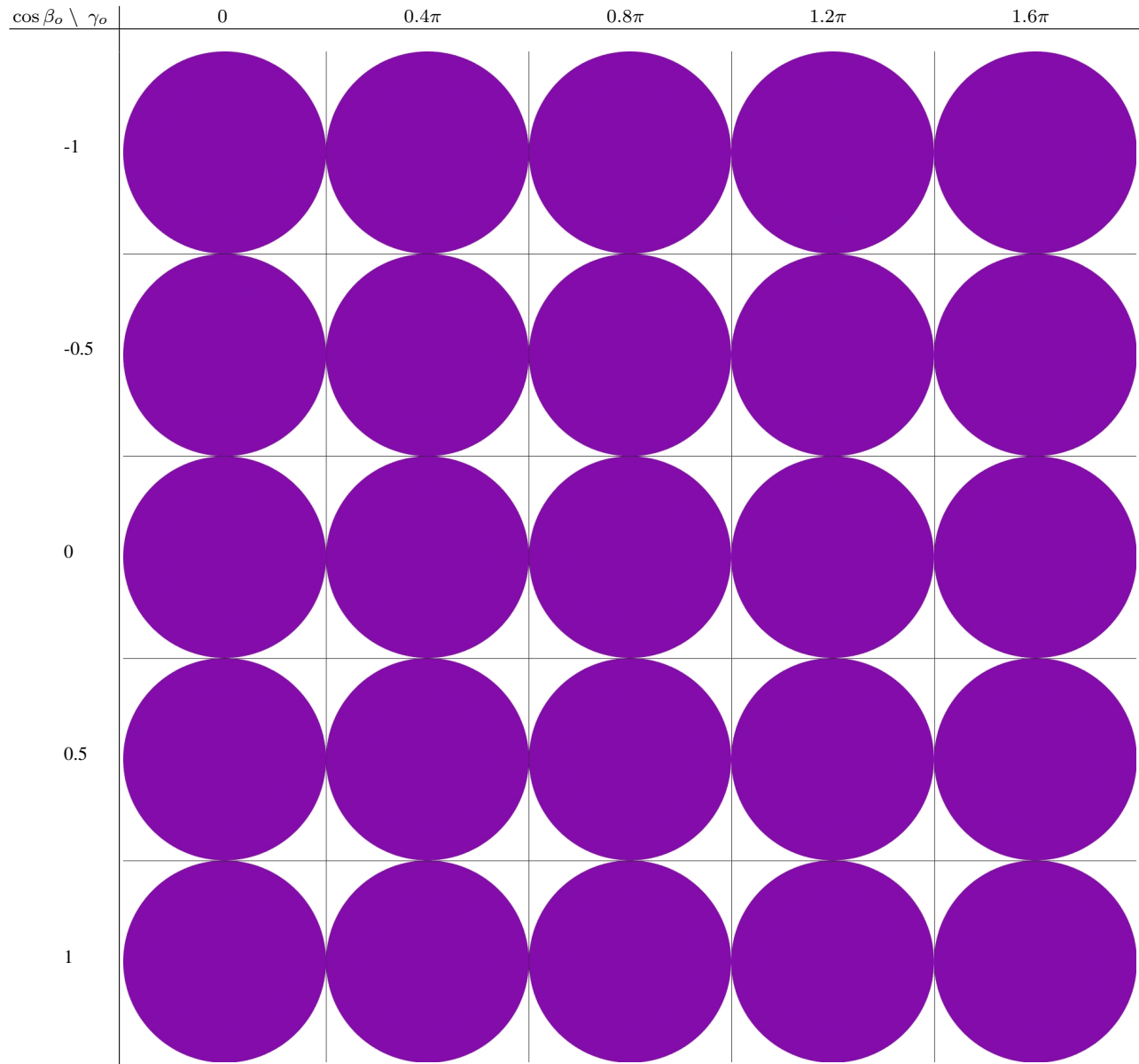


Figure 30: Back-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on dielectric sphere grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.



Figure 31: Front-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on diffuse sphere grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.

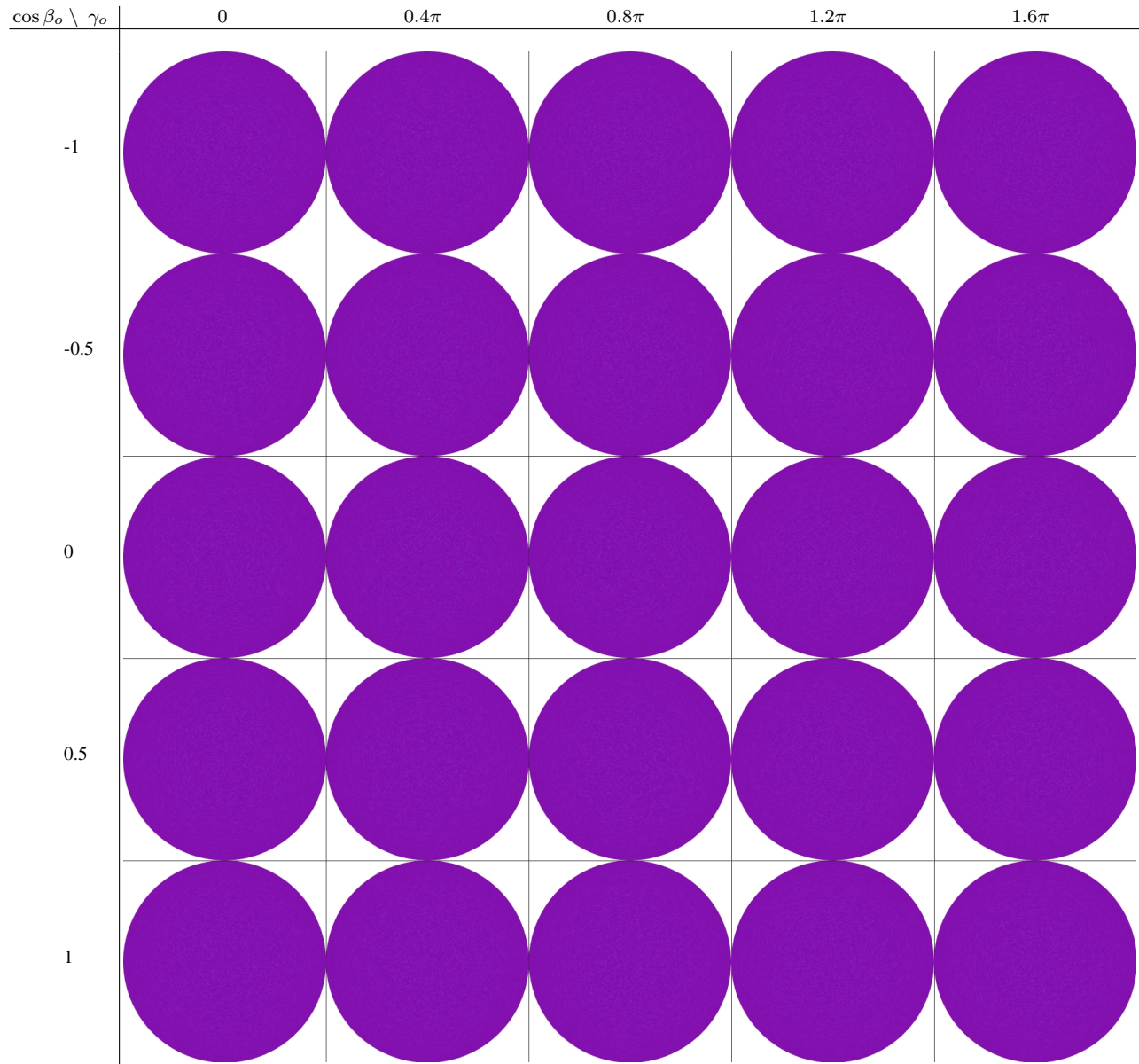


Figure 32: Back-facing directional GSDF approximation error $e_{f_g}(\vec{\omega}_i, \vec{\omega}_o)$ on diffuse sphere grains. We visualize this 4-dimensional error function by plotting slices through $\cos \beta_o$ (vertical) and γ_o (horizontal). The individual circular heatmaps encode $\sin \beta_i$ as the distance from their center and γ_i as their rotational component.