Unzip the **File**

```
import zipfile
import os

# Path to the zip file
zip_path = '/content/dataset.zip'
output_dir = '/content/dataset/'

# Unzip the file
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(output_dir)

print(f"Files extracted to {output_dir}")
```

```
Files extracted to /content/dataset/
```

List the Subfolders (Image Categories)

```
import os

# Path to the dataset containing subfolders
dataset_dir = '/content/dataset/dataset/'

# List all subfolders (categories)
categories = [folder for folder in os.listdir(dataset_dir) if os.path.isdir(os.path.join(dataset_dir, folde
print(f"Found {len(categories)} categories: {categories}")
```

```
Found 47 categories: ['fibrous', 'dotted', 'potholed', 'crystalline', 'chequered', 'zigzagged', 'cobweb
```

Count the Number of Images in Each Category

```
# Count the number of images in each category
for category in categories:
    category_path = os.path.join(dataset_dir, category)
    num_images = len([img for img in os.listdir(category_path) if img.endswith('.jpg')])
    print(f"Category '{category}' contains {num_images} images.")
```

```
Category 'fibrous' contains 120 images.
Category 'dotted' contains 120 images.
Category 'potholed' contains 120 images.
Category 'crystalline' contains 120 images.
Category 'chequered' contains 120 images.
Category 'zigzagged' contains 120 images.
Category 'cobwebbed' contains 120 images.
Category 'banded' contains 120 images.
Category 'woven' contains 120 images.
Category 'grooved' contains 120 images.
Category 'crosshatched' contains 120 images.
Category 'lined' contains 120 images.
Category 'studded' contains 120 images.
```

```
Category 'honeycombed' contains 120 images.
Category 'spiralled' contains 120 images.
Category 'scaly' contains 120 images.
Category 'pitted' contains 120 images.
Category 'cracked' contains 120 images.
Category 'grid' contains 120 images.
Category 'knitted' contains 120 images.
Category 'polka-dotted' contains 120 images.
Category 'stained' contains 120 images.
Category 'lacelike' contains 120 images.
Category 'wrinkled' contains 120 images.
Category 'matted' contains 120 images.
Category 'bubbly' contains 120 images.
Category 'sprinkled' contains 120 images.
Category 'paisley' contains 120 images.
Category 'porous' contains 120 images.
Category 'waffled' contains 120 images.
Category 'perforated' contains 120 images.
Category 'pleated' contains 120 images.
Category 'marbled' contains 120 images.
Category 'freckled' contains 120 images.
Category 'veined' contains 120 images.
Category 'flecked' contains 120 images.
Category 'smeared' contains 120 images.
Category 'meshed' contains 120 images.
Category 'braided' contains 120 images.
Category 'swirly' contains 120 images.
Category 'interlaced' contains 120 images.
Category 'gauzy' contains 120 images.
Category 'stratified' contains 120 images.
Category 'striped' contains 120 images.
Category 'blotchy' contains 120 images.
Category 'bumpy' contains 120 images.
Category 'frilly' contains 120 images.
```

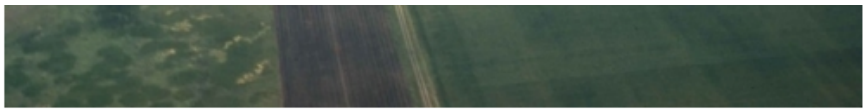
Load and Display an Example Image from Each Category

```
from PIL import Image
import matplotlib.pyplot as plt

# Load and display one image from each category
for category in categories:
    category_path = os.path.join(dataset_dir, category)
    image_path = os.path.join(category_path, os.listdir(category_path)[0])

    # Load the image
    img = Image.open(image_path)

    # Display the image
    plt.imshow(img)
    plt.title(f"Category: {category}")
    plt.axis('off')
    plt.show()
```

Category: fibrous

Category: dotted

Category: potholed



Category: crystalline

Category: chequered

Category: zigzagged

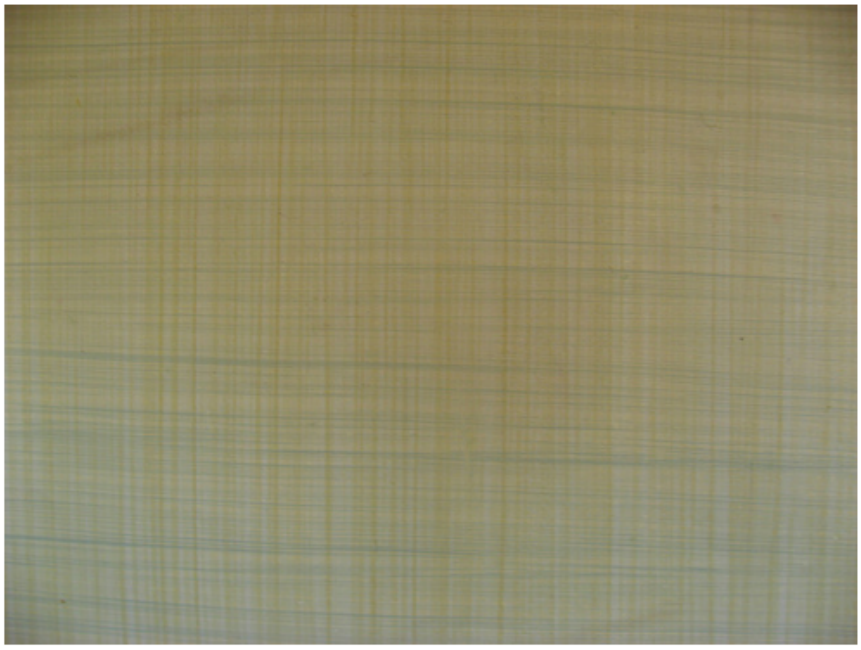
Category: cobwebbed

Category: banded

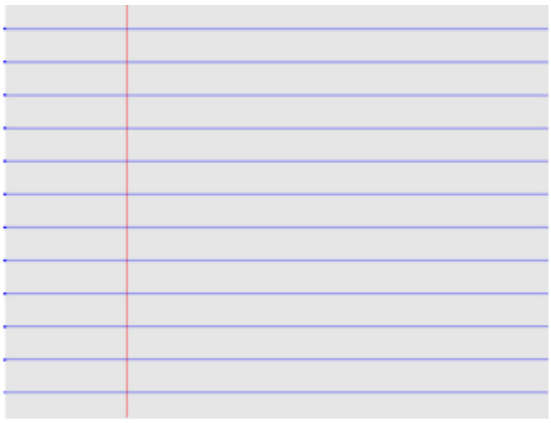
Category: woven

Category: grooved

Category: crosshatched

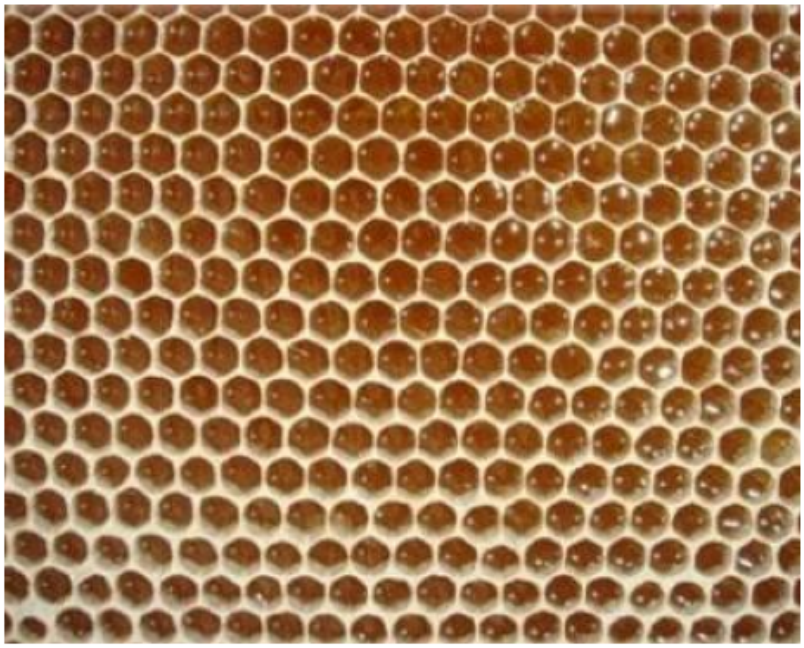


Category: lined

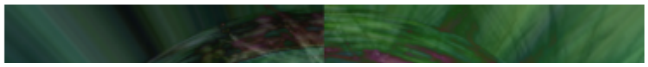


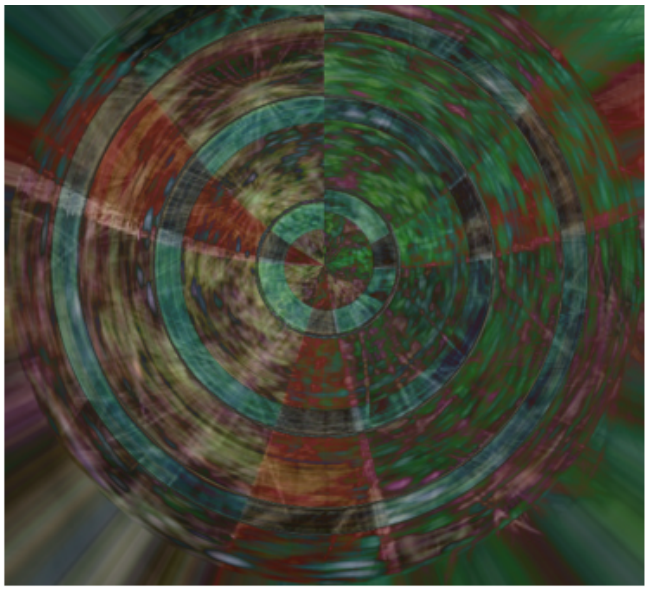
Category: studded

Category: honeycombed



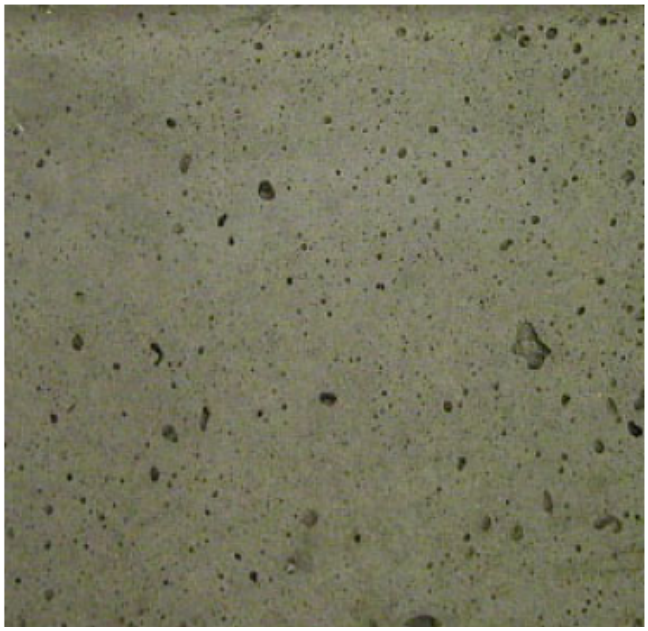
Category: spiralled



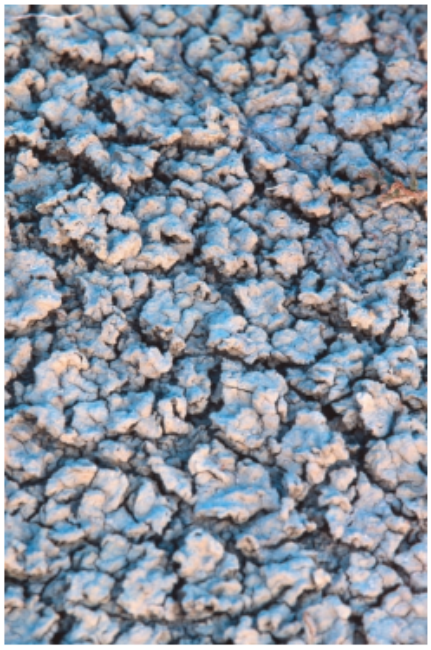


Category: scaly

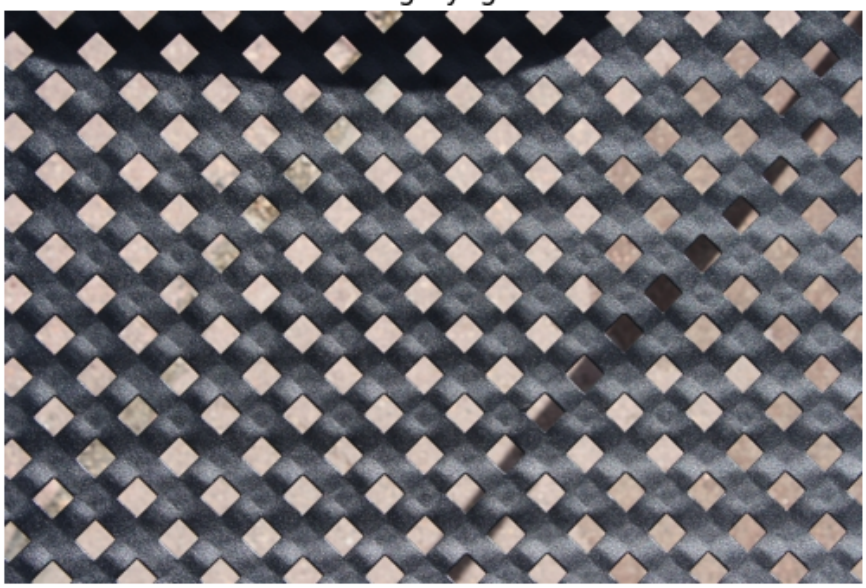
Category: pitted



Category: cracked

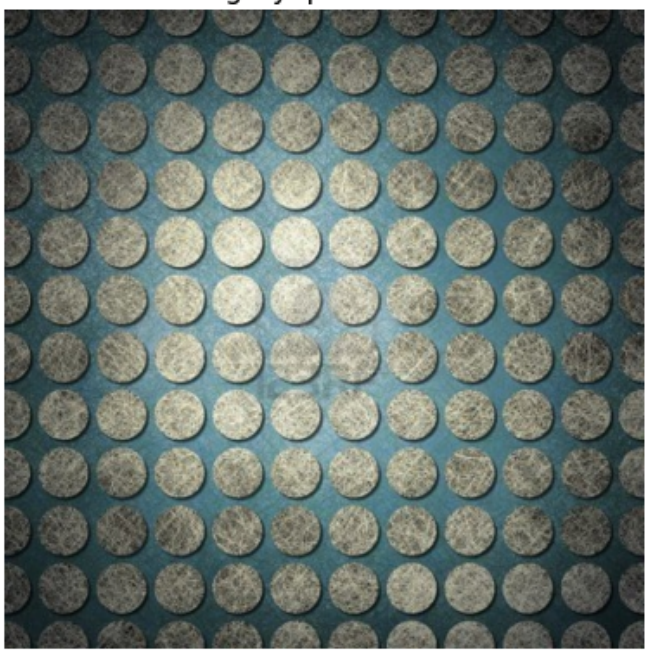


Category: grid



Category: knitted

Category: polka-dotted



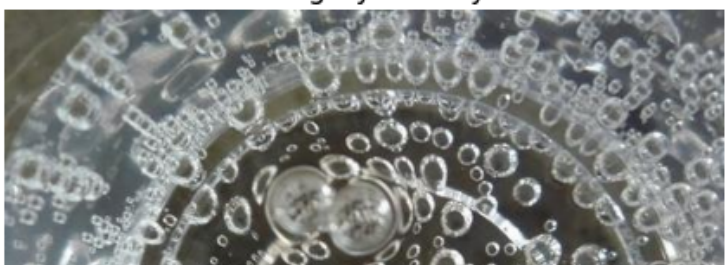
Category: stained

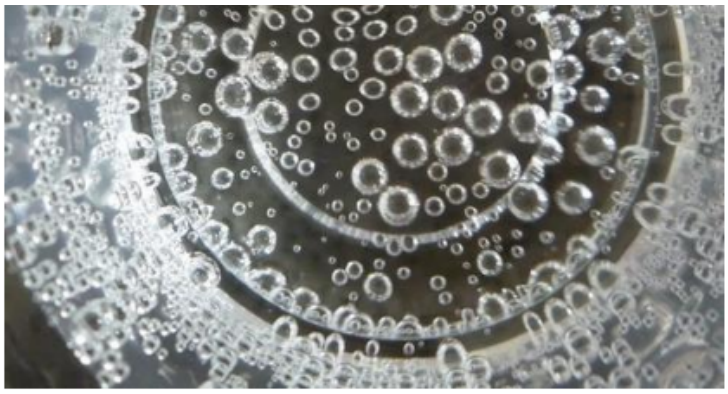
Category: lacelike

Category: wrinkled

Category: matted

Category: bubbly



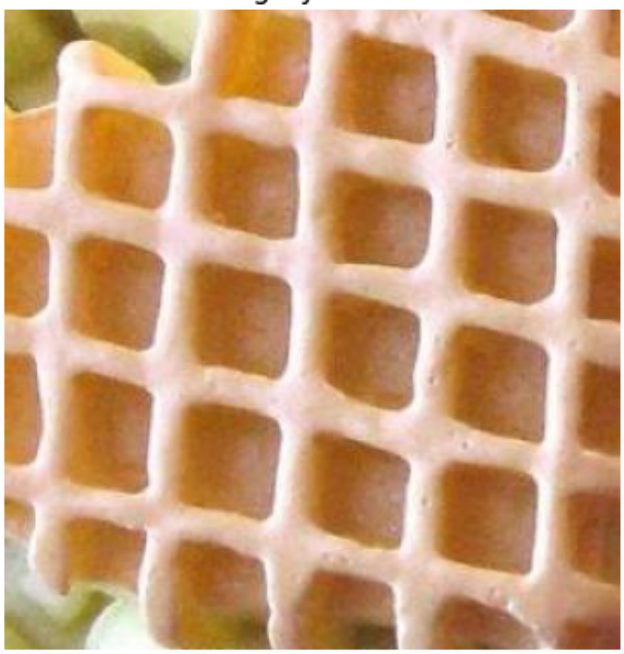


Category: sprinkled

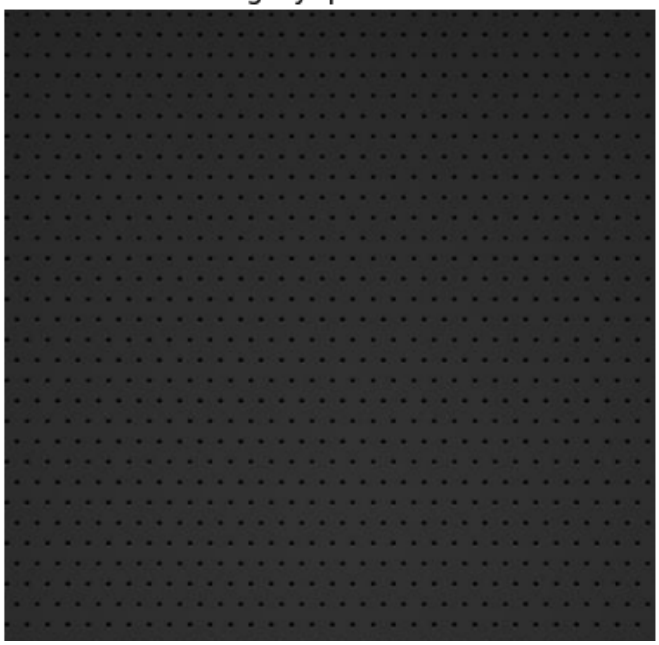
Category: paisley

Category: porous

Category: waffled



Category: perforated

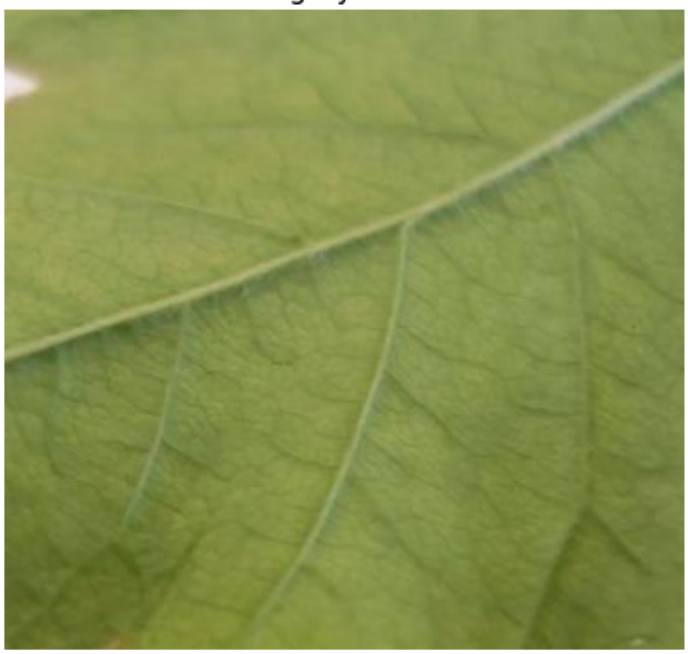


Category: pleated

Category: marbled

Category: freckled

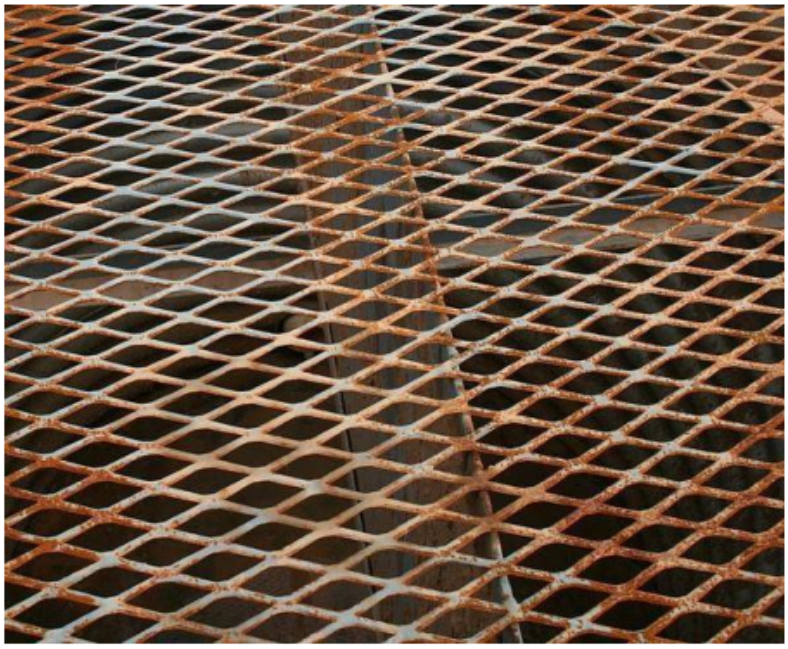
Category: veined



Category: flecked

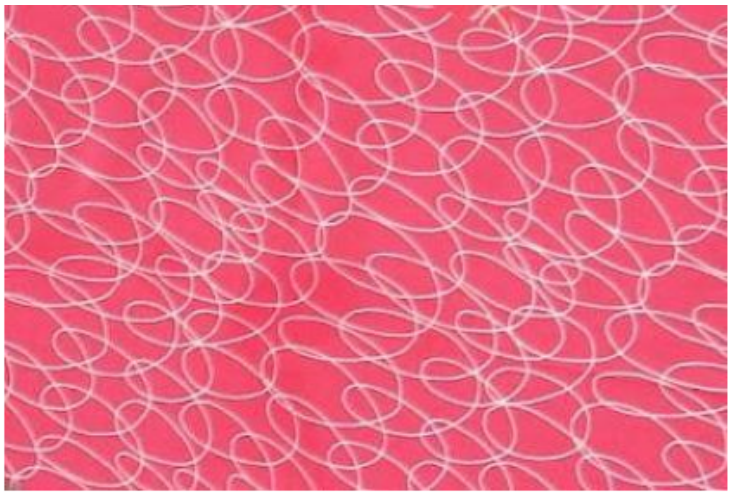
Category: smeared

Category: meshed

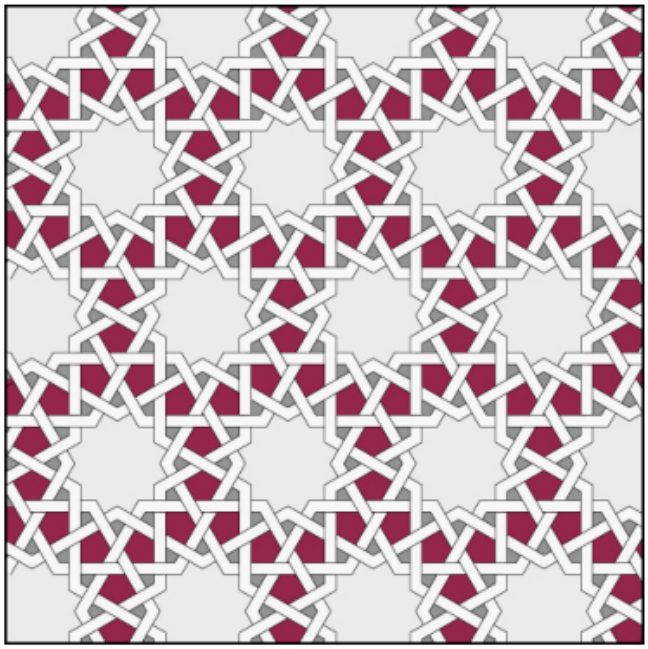


Category: braided

Category: swirly



Category: interlaced



Category: gauzy

Category: stratified

Category: striped

Category: blotchy

Category: bumpy

Category: frilly

```
from PIL import Image
import os

# Path to the dataset containing subfolders (categories)
dataset_dir = '/content/dataset/dataset/'

# Define the target size for resizing
target_size = (256, 256)  # Make sure this is defined

# Create a new directory for resized images
resized_dir = '/content/dataset/resized_dataset/'
if not os.path.exists(resized_dir):
    os.makedirs(resized_dir)

# Loop through the categories and save the resized images in a new folder
for category in os.listdir(dataset_dir):
    category_path = os.path.join(dataset_dir, category)

    if os.path.isdir(category_path):
        new_category_path = os.path.join(resized_dir, category)
        os.makedirs(new_category_path, exist_ok=True)

        # Loop over each image in the category
        for img_name in os.listdir(category_path):
            if img_name.endswith('.jpg'):
                img_path = os.path.join(category_path, img_name)

                # Open and resize the image
                img = Image.open(img_path)
                img_resized = img.resize(target_size)

                # Save the resized image to the new directory
                img_resized.save(os.path.join(new_category_path, img_name))

print("All images have been resized and saved to the new directory.")
```

```
All images have been resized and saved to the new directory.
```

```
import numpy as np
from PIL import Image
import os
from scipy.signal import convolve2d

# Define original Law's masks
def laws_mask(mask_type):
    if mask_type == 'L5':
        return np.array([[1,  4,  6,  4, 1],
                         [4, 16, 24, 16, 4],
                         [6, 24, 36, 24, 6],
                         [4, 16, 24, 16, 4],
                         [1,  4,  6,  4, 1]])  # Level mask
    elif mask_type == 'E5':
        return np.array([[1, -4,  0,  4, -1],
                         [0,  0,  0,  0,  0],
                         [0,  0,  0,  0,  0],
                         [0,  0,  0,  0,  0],
                         [0,  0,  0,  0,  0]])  # Edge mask
    elif mask_type == 'S5':
```