```
import numpy as np
import rasterio
from rasterio.windows import Window
from tqdm import tqdm

def resample_using_reference_grid_with_nodata(
    tif1_path, tif2_path, reference_1km_tif, output_path,
    src_res=30, target_res=1000
):
    factor = target_res // src_res

    with rasterio.open(tif1_path) as src1, \
         rasterio.open(tif2_path) as src2, \
         rasterio.open(reference_1km_tif) as ref:

        ref_transform = ref.transform
        ref_height, ref_width = ref.height, ref.width
        profile = ref.profile
        profile.update(dtype=rasterio.uint8, count=1, compress='lzw', nodata=0)

        nodata1 = src1.nodata
        nodata2 = src2.nodata

        out_array = np.zeros((ref_height, ref_width), dtype=np.uint8)

        assigned_count = 0   # 统计赋值为 1 的像元数量


        for i in tqdm(range(ref_height), desc="Processing"):
            for j in range(ref_width):
                # 当前像素中心坐标

                x, y = ref_transform * (j + 0.5, i + 0.5)


                # 转换为 30m 图像的像素行列

                col_start, row_start = ~src1.transform * (x, y)
                col_start, row_start = int(col_start), int(row_start)
                col_end, row_end = col_start + factor, row_start + factor


                # 边界检查

                if row_start < 0 or col_start < 0 or row_end > src1.height or
col_end > src1.width:
                    continue
```

```
            # 提取窗口数据
            window = Window(col_start, row_start, factor, factor)
            block1 = src1.read(1, window=window)
            block2 = src2.read(1, window=window)

            # 构建有效掩膜
            valid_mask = np.ones_like(block1, dtype=bool)
            if nodata1 is not None:
                valid_mask &= (block1 != nodata1)
            if nodata2 is not None:
                valid_mask &= (block2 != nodata2)

            valid_block1 = block1[valid_mask]
            valid_block2 = block2[valid_mask]

            if valid_block1.size == 0:
                continue

            # 条件 1：tif2 中 1 值 ≥ 50%
            tif2_count_1 = np.sum(valid_block2 == 1)
            if tif2_count_1 / valid_block2.size < 0.5:
                continue

            # 条件 2：tif1 中在 tif2==1 的区域中，1 值 ≥ 50%
            mask_tif2_1 = valid_block2 == 1
            if np.sum(mask_tif2_1) == 0:
                continue

            tif1_count_1 = np.sum(valid_block1[mask_tif2_1] == 1)
            if tif1_count_1 / np.sum(mask_tif2_1) >= 0.5:
                out_array[i, j] = 1
                assigned_count += 1

print(f"✓ 总共赋值为 1 的像元数量: {assigned_count}")

# 写出新 TIF
with rasterio.open(output_path, 'w', **profile) as dst:
```

```
                dst.write(out_array, 1)

    print(f"✅ 重采样完成，输出路径: {output_path}")



resample_using_reference_grid_with_nodata(
    tif1_path='E:/YJS/TIF/2000_2021rubber/2021.tif',
    tif2_path='E:/next/lucu/DQ2021_f_30m.tif',
    reference_1km_tif='E:/next/01 渔网/ikm_0.tif',
    output_path='E:/next/20251013/rubber_forest_05_1km_filled.tif'
)
```

```
import numpy as np
import fiona
import rasterio
from rasterio.windows import from_bounds, Window
from rasterio.features import geometry_mask
from tqdm import tqdm


# 输入/输出路径

src_path = 'E:/next/ndvi/30m_ndvi_q.tif'

ref_path = 'E:/next/ndvi/还原_1km 按 F_R_1 掩膜提取.tif'

shp_path = 'E:/next/ndvi/去除部分/shp/1km_qc01.shp'          # 你的 shp 文件

dst_path = 'E:/next/ndvi/尺度效应/ndvi_1km_cd_shp.tif'


# 定义目标函数及其二阶导数

def f(x):
    return x

def f_dd(x):
    return 0


# —— 第一步：读参考栅格，获取空间信息 ——

with rasterio.open(ref_path) as ref:
    ref_transform = ref.transform
    ref_width = ref.width
    ref_height = ref.height
    ref_crs = ref.crs
    ref_dtype = ref.dtypes[0]


# —— 第二步：读 shp，生成几何掩膜 ——

# 用 fiona 读取所有 geometry，并转换到参考栅格 CRS

with fiona.open(shp_path, 'r') as shp:

    # 如果 shp 的 CRS 与 ref_crs 不一致，需要先转换：

    shp_crs = shp.crs_wkt
    shapes = [feature["geometry"] for feature in shp]
```

```
# 在参考栅格网格上做 mask：invert=True 表示掩膜内为 True
mask_geom = geometry_mask(
    shapes,
    out_shape=(ref_height, ref_width),
    transform=ref_transform,
    invert=True
)

# 取出要处理的像元行列号
valid_rows, valid_cols = np.where(mask_geom)

# —— 第三步：对 30m 数据做二阶校正，只遍历 shp 区域 ——
with rasterio.open(src_path) as src:
    src_transform = src.transform
    src_crs = src.crs

    profile = {
        'driver': 'GTiff',
        'height': ref_height,
        'width': ref_width,
        'count': 1,
        'dtype': 'float32',
        'crs': ref_crs,
        'transform': ref_transform,
        'nodata': np.nan
    }

    with rasterio.open(dst_path, 'w', **profile) as dst:
        total = len(valid_rows)
        with tqdm(total=total, desc="Processing valid cells", unit="cell") as pbar:
            for row, col in zip(valid_rows, valid_cols):
                # 计算 1×1 目标像元的地理边界
                x_min, y_max = ref_transform * (col, row)
                x_max = x_min + ref_transform.a
                y_min = y_max + ref_transform.e

                # 在 30m 栅格上切窗读取
                window = from_bounds(
                    left=x_min, bottom=y_min,
```

```
            right=x_max, top=y_max,
            transform=src_transform
        )
        data = src.read(
            1, window=window,
            boundless=True, fill_value=np.nan
        ).astype(float)

        # 计算二阶校正值
        valid = data[~np.isnan(data)]
        if valid.size == 0:
            out_val = np.nan
        else:
            mu = valid.mean()
            sigma2 = valid.var()
            out_val = f(mu) + 0.5 * f_dd(mu) * sigma2

        # 写入结果栅格
        dst.write(
            np.array([[out_val]], dtype='float32'),
            1,
            window=Window(col, row, 1, 1)
        )
        pbar.update(1)

print(f"✅ 共对 shp 区域内 {total} 个像元完成二阶校正，结果保存在：{dst_path}")
```