EE604 H2
Binary Mask Generation

LOHIT P TALAVAR
210564
lohitpt21@iitk.ac.in

August 14, 2025

Contents

1 Results (visual)

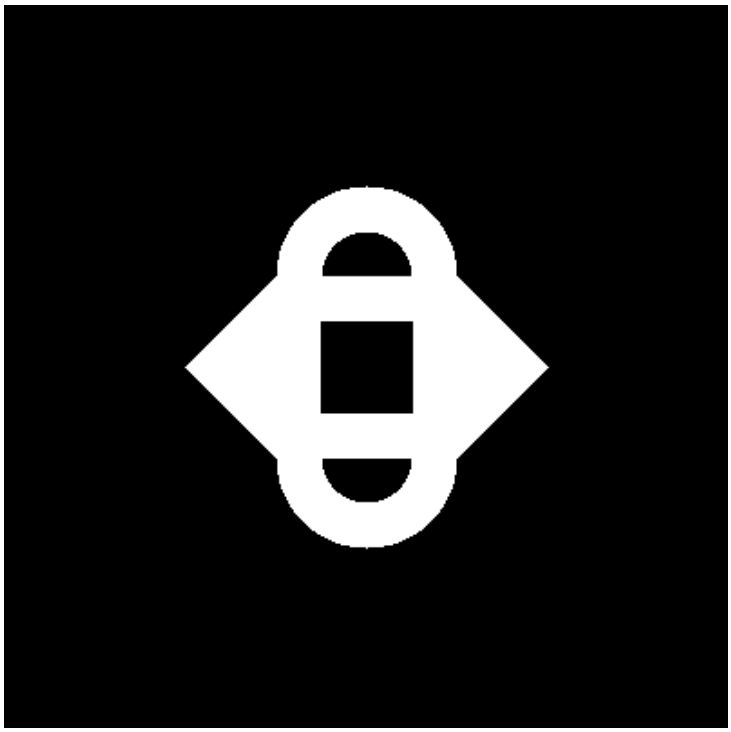

Figure 1: Generated binary mask saved as `mask_binary.png`.

2 Key code (make_mask.py)

```python
#!/usr/bin/env python3
"""
make_mask.py
Generate a binary mask and save as mask_binary.png.
"""
import argparse
import numpy as np
from PIL import Image

def clamp(v, lo, hi):
    return max(lo, min(v, hi))

def make_mask(r=50, img_size=None):
    if img_size is None:
        img_size = 8 * r
    img_size = max(img_size, 4 * r + 10)
    mask = np.zeros((img_size, img_size), dtype=np.uint8)
    cx, cy = img_size // 2, img_size // 2

    outer_x1, outer_x2 = cx - r, cx + r
    outer_y1, outer_y2 = cy - r, cy + r
    inner_x1, inner_x2 = cx - r // 2, cx + r // 2
    inner_y1, inner_y2 = cy - r // 2, cy + r // 2

    # Fill outer rectangle
    x0 = clamp(outer_x1, 0, img_size - 1)
    x1 = clamp(outer_x2, 0, img_size - 1)
    y0 = clamp(outer_y1, 0, img_size - 1)
    y1 = clamp(outer_y2, 0, img_size - 1)
    for y in range(y0, y1 + 1):
        mask[y, x0:x1 + 1] = 255

    # Cut out inner rectangle
    ix0 = clamp(inner_x1, 0, img_size - 1)
    ix1 = clamp(inner_x2, 0, img_size - 1)
    iy0 = clamp(inner_y1, 0, img_size - 1)
    iy1 = clamp(inner_y2, 0, img_size - 1)
    for y in range(iy0, iy1 + 1):
        mask[y, ix0:ix1 + 1] = 0

    # Left triangle ear
    for y in range(y0, y1 + 1):
        if y <= cy:
            dy = y - outer_y1
        else:
            dy = outer_y2 - y
        dx = int(round((dy / max(1, r)) * r))
        sx = clamp(outer_x1 - dx, 0, img_size - 1)
        ex = clamp(outer_x1, 0, img_size - 1)
        mask[y, sx:ex + 1] = 255

    # Right triangle ear
    for y in range(y0, y1 + 1):
        if y <= cy:
            dy = y - outer_y1
        else:
            dy = outer_y2 - y
        dx = int(round((dy / max(1, r)) * r))
        sx = clamp(outer_x2, 0, img_size - 1)
        ex = clamp(outer_x2 + dx, 0, img_size - 1)
        mask[y, sx:ex + 1] = 255
```

```
62
63      # Top hollow semicircle (centered at (cx, outer_y1))
64      rad = r
65      rad_inner = max(1, r // 2)
66      ty0 = clamp(outer_y1 - rad, 0, img_size - 1)
67      ty1 = clamp(outer_y1, 0, img_size - 1)
68      tx0 = clamp(outer_x1, 0, img_size - 1)
69      tx1 = clamp(outer_x2, 0, img_size - 1)
70      for y in range(ty0, ty1 + 1):
71          for x in range(tx0, tx1 + 1):
72              eq_outer = ((x - cx) ** 2) / (rad ** 2) + ((y - outer_y1) ** 2) / (rad
                    ** 2)
73              eq_inner = ((x - cx) ** 2) / (rad_inner ** 2) + ((y - outer_y1) ** 2) /
                     (rad_inner ** 2)
74              if eq_outer <= 1.0 and eq_inner >= 1.0:
75                  mask[y, x] = 255
76
77      # Bottom hollow semicircle (centered at (cx, outer_y2))
78      by0 = clamp(outer_y2, 0, img_size - 1)
79      by1 = clamp(outer_y2 + rad, 0, img_size - 1)
80      bx0 = clamp(outer_x1, 0, img_size - 1)
81      bx1 = clamp(outer_x2, 0, img_size - 1)
82      for y in range(by0, by1 + 1):
83          for x in range(bx0, bx1 + 1):
84              eq_outer = ((x - cx) ** 2) / (rad ** 2) + ((y - outer_y2) ** 2) / (rad
                    ** 2)
85              eq_inner = ((x - cx) ** 2) / (rad_inner ** 2) + ((y - outer_y2) ** 2) /
                     (rad_inner ** 2)
86              if eq_outer <= 1.0 and eq_inner >= 1.0:
87                  mask[y, x] = 255
88
89      return mask
90
91  if __name__ == "__main__":
92      parser = argparse.ArgumentParser()
93      parser.add_argument("--r", type=int, default=50, help="scale parameter r")
94      parser.add_argument("--out", type=str, default="mask_binary.png", help="output
            filename")
95      args = parser.parse_args()
96
97      r = max(1, args.r)
98      mask = make_mask(r=r)
99      Image.fromarray(mask).save(args.out)
100
101     # print proof (optional)
102     fg = int(np.count_nonzero(mask))
103     total = mask.size
104     coords = np.argwhere(mask)
105     if coords.size > 0:
106         ymin, xmin = coords.min(axis=0)
107         ymax, xmax = coords.max(axis=0)
108         width = int(xmax - xmin + 1)
109         height = int(ymax - ymin + 1)
110     else:
111         ymin = xmin = width = height = 0
112     coverage = 100.0 * fg / total if total else 0.0
113     print(f"Saved image to: {args.out}")
114     print(f"r = {r}")
115     print(f"Image size: {mask.shape[1]} x {mask.shape[0]}")
116     print(f"Foreground pixels: {fg}")
117     print(f"Bounding box (xmin, ymin, width, height): ({xmin}, {ymin}, {width}, {
            height})")
118     print(f"Coverage: {coverage:.3f}%")
```

Listing 1: make mask.py