EE604 - H4

Lohit P Talavar
210564

September 2, 2025

1 Python implementation

Below is the full Python script used to run the synthetic tests and to apply the morphological operations automatically on `image.png`.

```python
# morphology_tests.py
# -*- coding: utf-8 -*-
"""
Run morphological erosion and dilation tests and apply them to image.png.

Outputs multiple PNGs in the working directory.
If you see ImportError about libGL, install headless OpenCV:
    pip install opencv-python-headless
"""

import cv2
import numpy as np
from typing import Set, Tuple

Pixel = Tuple[int, int]
ImageSize = Tuple[int, int]

# -------------------------
# Core set-based morphology
# -------------------------

def erode(image_pixels: Set[Pixel], image_size: ImageSize, se_pixels: Set[Pixel
    ]) -> Set[Pixel]:
    rows, cols = image_size
    out = set()
    for r in range(rows):
        for c in range(cols):
            ok = True
            for (dr, dc) in se_pixels:
                rr, cc = r + dr, c + dc
                if not (0 <= rr < rows and 0 <= cc < cols and (rr, cc) in
                    image_pixels):
                    ok = False
                    break
            if ok:
                out.add((r, c))
    return out

def dilate(image_pixels: Set[Pixel], se_pixels: Set[Pixel]) -> Set[Pixel]:
    out = set()
    if not image_pixels or not se_pixels:
        return out
    for (r, c) in image_pixels:
        for (dr, dc) in se_pixels:
            out.add((r + dr, c + dc))
    return out

# -------------------------
# Utilities
# -------------------------

def image_to_set(binary_img: np.ndarray) -> Set[Pixel]:
    coords = np.argwhere(binary_img != 0)
    return set((int(r), int(c)) for r, c in coords)

def structuring_element_to_set(struct_elem: np.ndarray, origin: Pixel) -> Set[
    Pixel]:
    rows, cols = struct_elem.shape
    or_r, or_c = origin
    if not (0 <= or_r < rows and 0 <= or_c < cols):
        raise ValueError("origin must be within SE bounds")
    offsets = set()
    for r in range(rows):
        for c in range(cols):
            if struct_elem[r, c] != 0:
                offsets.add((r - or_r, c - or_c))
    return offsets

def set_to_image(pixel_set: Set[Pixel], shape: ImageSize) -> np.ndarray:
    img = np.zeros(shape, dtype=np.uint8)
    for (r, c) in pixel_set:
        if 0 <= r < shape[0] and 0 <= c < shape[1]:
            img[r, c] = 1
    return img

def set_to_tight_image(pixel_set: Set[Pixel]) -> Tuple[np.ndarray, Tuple[int,
    int]]:
    if not pixel_set:
        return np.zeros((1,1), dtype=np.uint8), (0,0)
    rows = [r for r,_ in pixel_set]; cols = [c for _,c in pixel_set]
    rmin, rmax = min(rows), max(rows); cmin, cmax = min(cols), max(cols)
    h, w = rmax - rmin + 1, cmax - cmin + 1
    img = np.zeros((h, w), dtype=np.uint8)
    for (r, c) in pixel_set:
        img[r - rmin, c - cmin] = 1
    return img, (rmin, cmin)

def create_structuring_element(shape: str = 'square', size: int = 3) -> np.
    ndarray:
    if size <= 0 or size % 2 == 0:
        raise ValueError("size must be a positive odd integer")
    if shape == 'square':
        return np.ones((size, size), dtype=np.uint8)
    if shape == 'cross':
        se = np.zeros((size, size), dtype=np.uint8)
        mid = size // 2
        se[mid, :] = 1; se[:, mid] = 1
        return se
    raise ValueError("shape must be 'square' or 'cross'")

# -------------------------
# Tests + image application
# -------------------------

def synthetic_tests_and_save():
    # TEST 1
    image_size_1 = (10, 10)
    image_pixels_1 = {(4,4),(4,5),(4,6),(5,4),(5,5),(5,6),(6,4),(6,5),(6,6)}
    se1 = create_structuring_element('square', 3)
    se1_set = structuring_element_to_set(se1, (1,1))
    dil1 = dilate(image_pixels_1, se1_set)
    ero1 = erode(image_pixels_1, image_size_1, se1_set)
    print("Test1 - eroded pixels:", sorted(ero1))
    cv2.imwrite("testcase1_eroded.png", set_to_image(ero1, image_size_1)*255)
    dil_img1, top_left1 = set_to_tight_image(dil1)
    cv2.imwrite("testcase1_dilated_tight.png", dil_img1*255)
    print("Test1 dilation top-left:", top_left1)

    # TEST 2
    image_size_2 = (15, 15)
    image_pixels_2 = {(r,7) for r in range(2,12)}
    se2 = create_structuring_element('cross', 3)
    se2_set = structuring_element_to_set(se2, (1,1))
    dil2 = dilate(image_pixels_2, se2_set)
    ero2 = erode(image_pixels_2, image_size_2, se2_set)
    print("Test2 - eroded pixels (expected empty):", sorted(ero2))
    cv2.imwrite("testcase2_eroded.png", set_to_image(ero2, image_size_2)*255)
    dil_img2, top_left2 = set_to_tight_image(dil2)
    cv2.imwrite("testcase2_dilated_tight.png", dil_img2*255)
    print("Test2 dilation top-left:", top_left2)

    # TEST 3
    image_size_3 = (20,20)
    image_pixels_3 = {(5,5),(10,10),(15,15)}
    se3 = create_structuring_element('square', 5)
    se3_set = structuring_element_to_set(se3, (2,2))
    dil3 = dilate(image_pixels_3, se3_set)
    ero3 = erode(image_pixels_3, image_size_3, se3_set)
    print("Test3 - eroded pixels (expected empty):", sorted(ero3))
    cv2.imwrite("testcase3_eroded.png", set_to_image(ero3, image_size_3)*255)
    dil_img3, top_left3 = set_to_tight_image(dil3)
    cv2.imwrite("testcase3_dilated_tight.png", dil_img3*255)
    print("Test3 dilation top-left:", top_left3)

def run_on_image_file(image_path: str = "image.png", resize_to: ImageSize =
    (100,100)):
    print(f"\nApplying morphological ops to '{image_path}' (resized to {
        resize_to})")
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    if img is None:
        print(f"Error: '{image_path}' not found in current directory. Skipping
            image test.")
        return
    img_resized = cv2.resize(img, (resize_to[1], resize_to[0]))
    _, binary_img = cv2.threshold(img_resized, 127, 1, cv2.THRESH_BINARY)
    image_set = image_to_set(binary_img)
    se = create_structuring_element('square', 3)
    se_set = structuring_element_to_set(se, (1,1))
    dil = dilate(image_set, se_set)
    ero = erode(image_set, resize_to, se_set)
    cv2.imwrite("original_image_4.png", binary_img * 255)
    cv2.imwrite("eroded_image_4.png", set_to_image(ero, resize_to) * 255)
    dil_tight, offset = set_to_tight_image(dil)
    cv2.imwrite("dilated_image_4_tight.png", dil_tight * 255)
    cv2.imwrite("dilated_image_4.png", set_to_image(dil, resize_to) * 255)
    print("Saved: original_image_4.png, eroded_image_4.png, dilated_image_4.png
        , dilated_image_4_tight.png")
    print("Dilation tight-crop top-left offset in original coords:", offset)

# helper re-used
```

```python
def image_to_set(binary_img: np.ndarray):
    coords = np.argwhere(binary_img != 0)
    return set((int(r), int(c)) for r, c in coords)

if __name__ == "__main__":
    print("--- Running Synthetic Tests ---")
    synthetic_tests_and_save()
    print("\n--- Running Image Test on 'image.png' ---")
    run_on_image_file("image.png", resize_to=(100,100))
    print("\nAll done. Check PNG files in the working directory.")
```

2 Results and Generated Images

2.1 Synthetic Testcases

[b]0.45

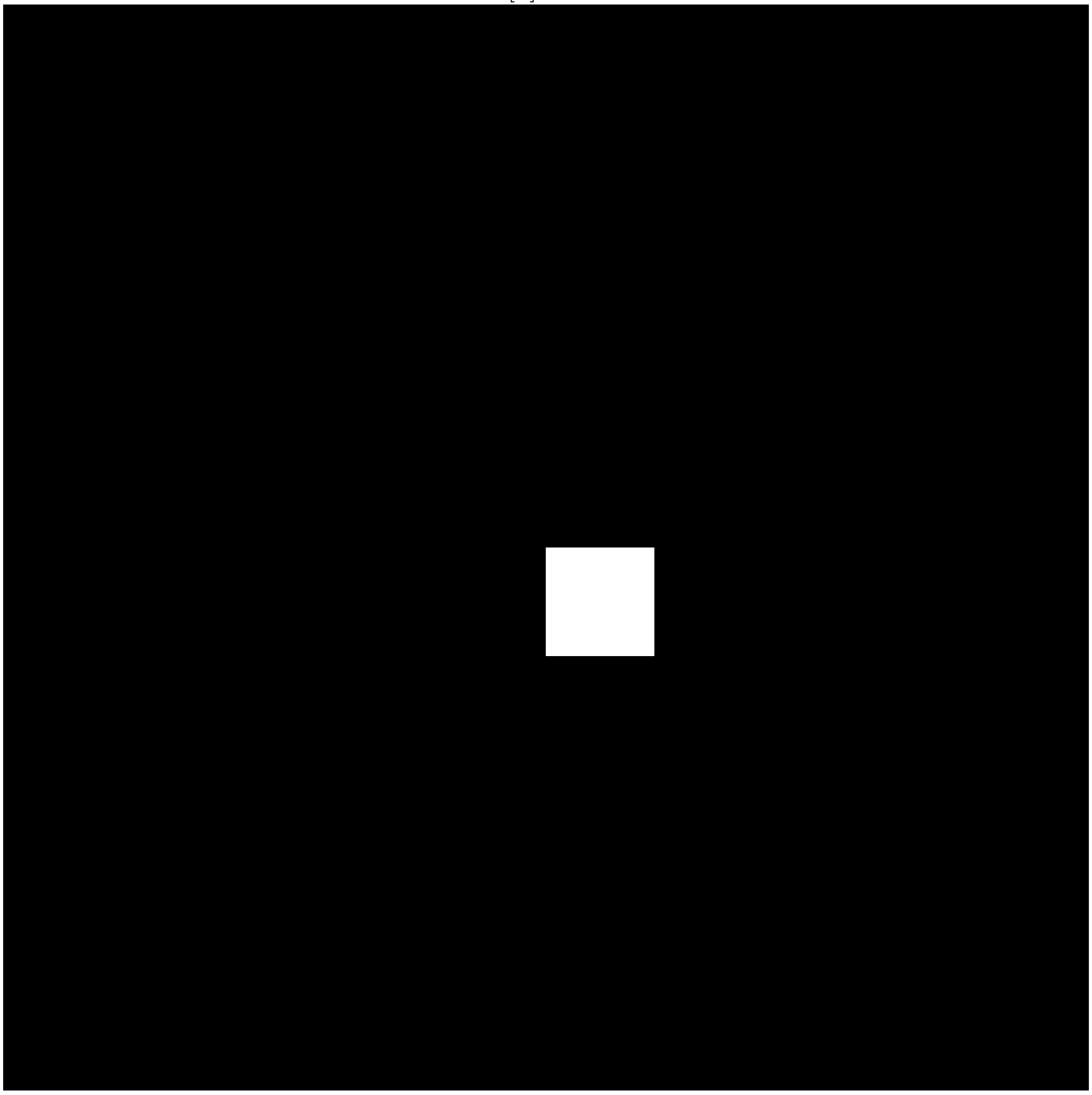


Figure 1: Test 1 — eroded (3x3 square SE)

[b]0.45

[b]0.45

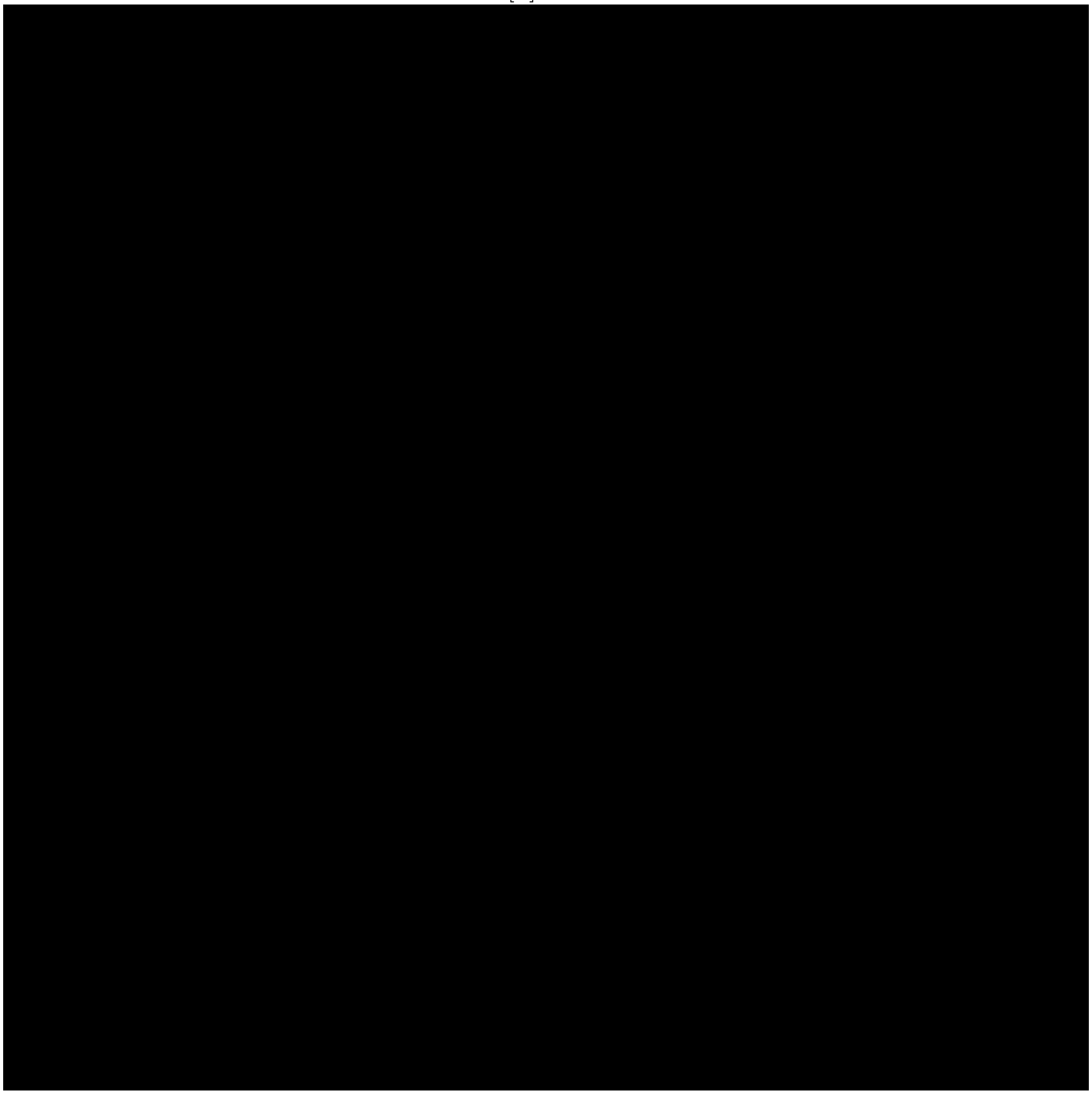
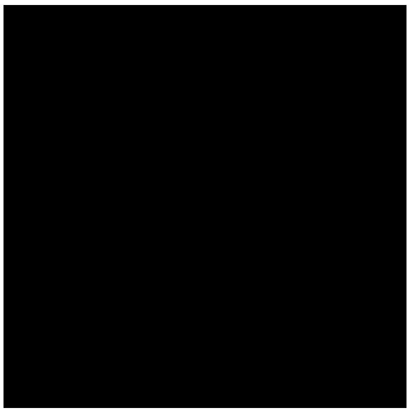
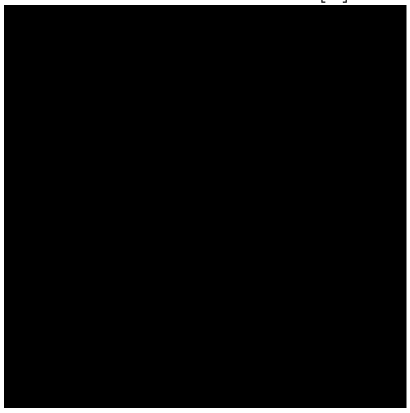


Figure 4: Test 2 — eroded (3x3 cross SE)

[b]0.45





[b]0.45

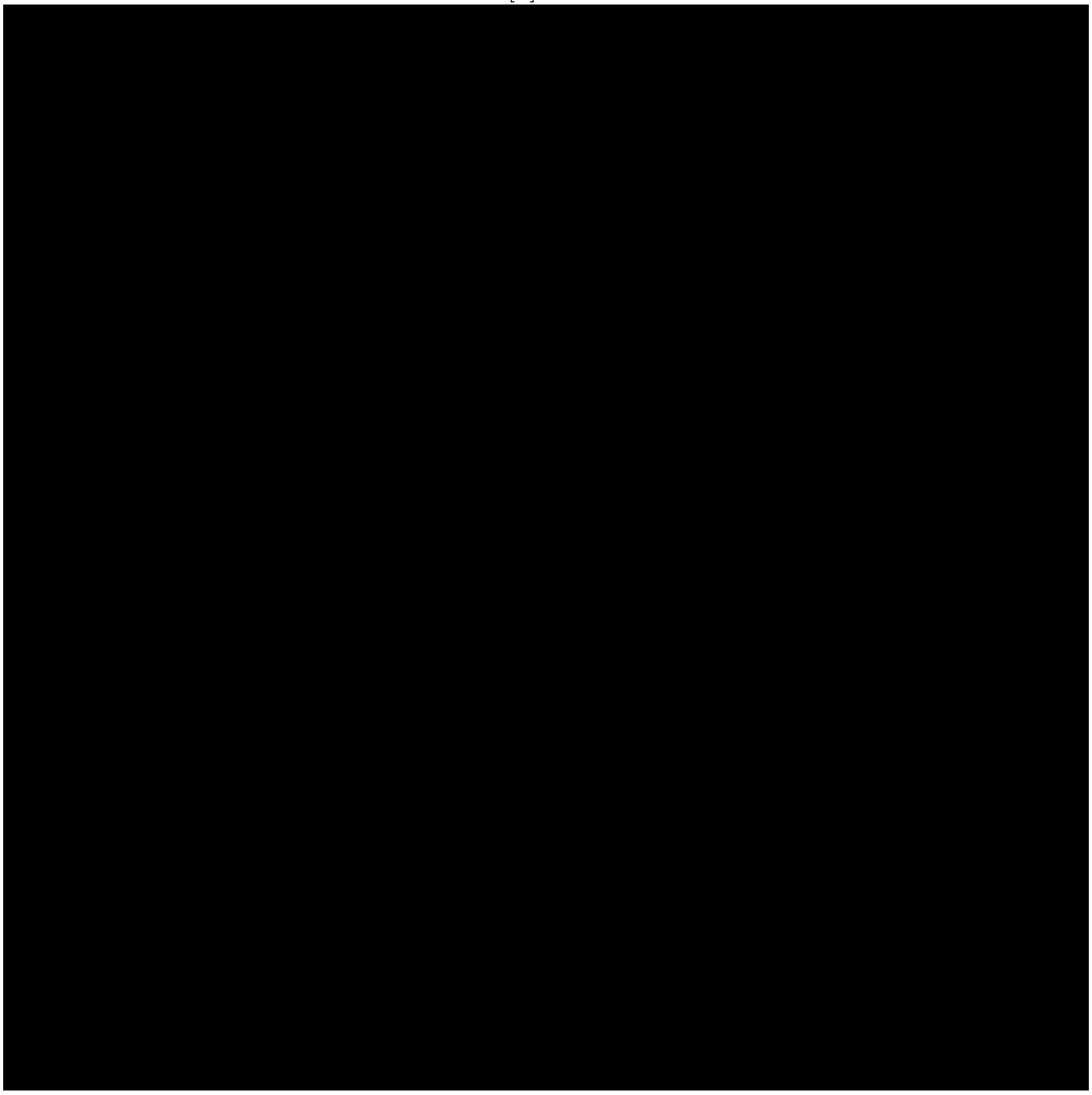
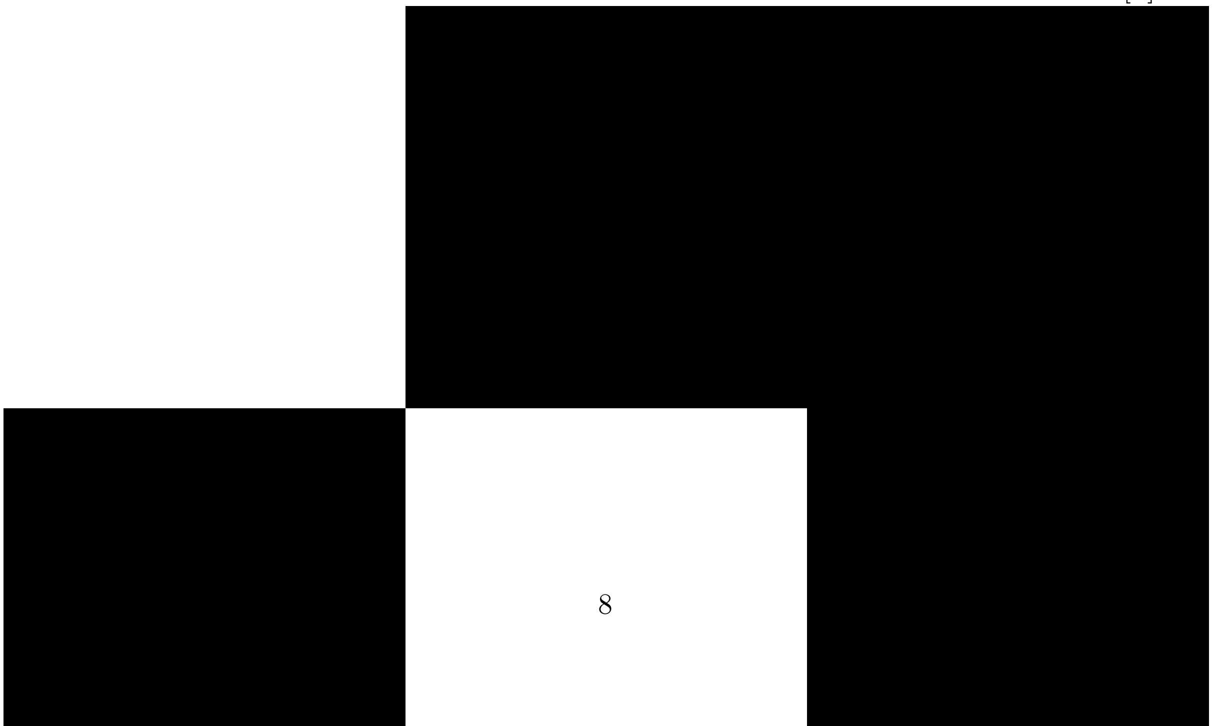


Figure 7: Test 3 — eroded (5x5 square SE)

[b]0.45



2.2 Real Image: `image.png`

[b]0.32

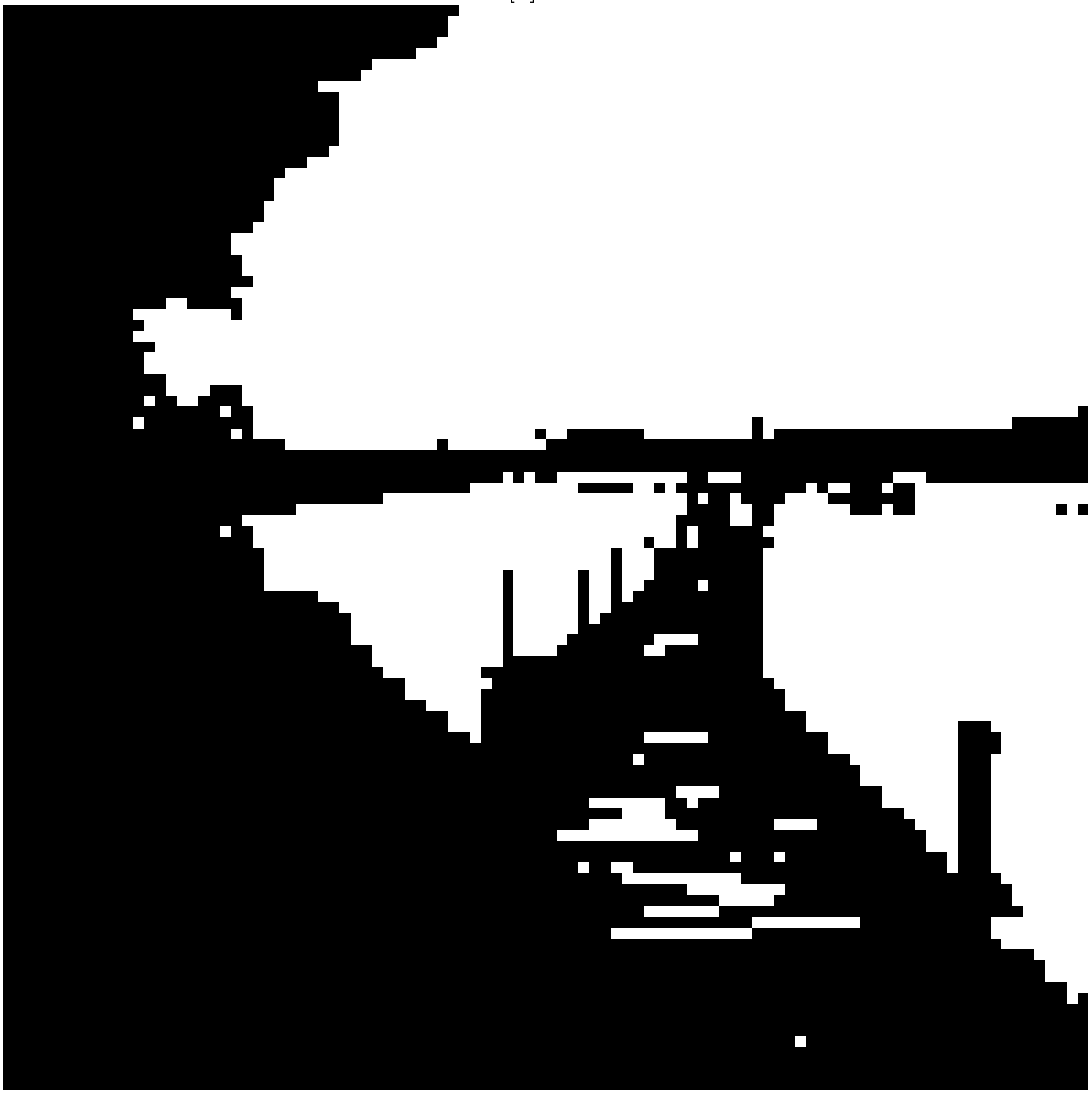
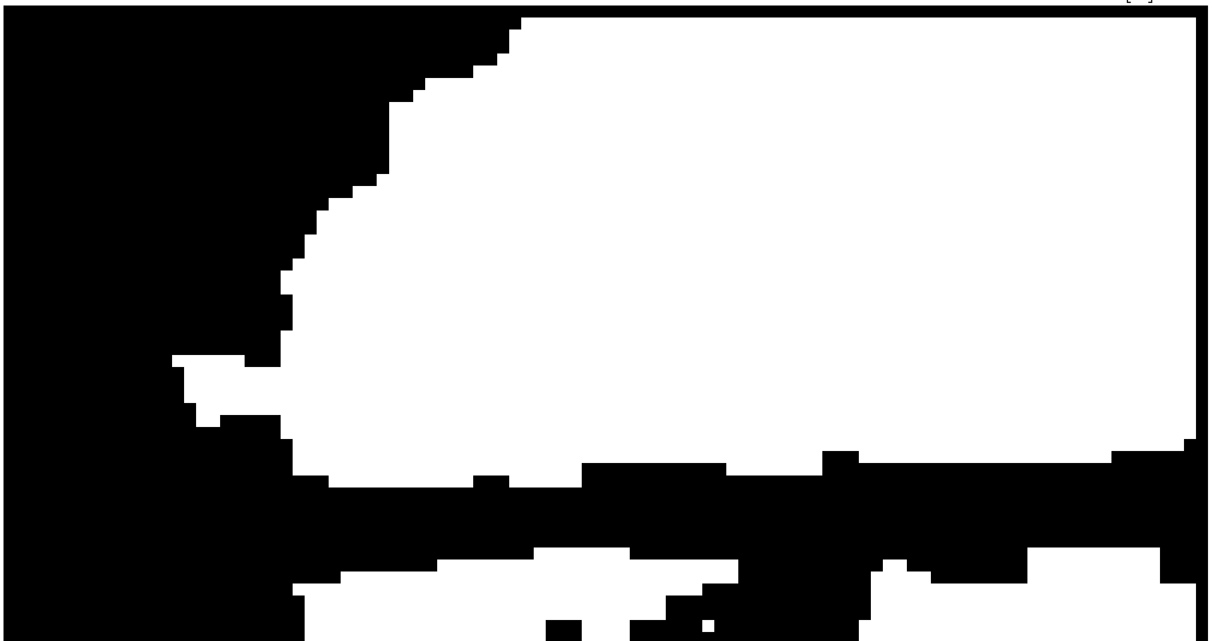


Figure 10: Original (thresholded)

[b]0.32



Captions and labels Each subfigure has an individual caption and label so you can reference them in the text: e.g. “see Figure 3 (Test 1 dilation)” or “see Figure 14 for the results on `image.png`.”