EE604 H1

LOHIT P TALAVAR
210564
lohitpt21@iitk.ac.in

August 10, 2025

Contents

1 Results and short code snippets

1.1 Transpose

Figure 1: Transpose of the original image (rows ↔ columns).

```
# For each (i,j) in HxW assign pixel -> (j,i)
out = zeros((W,H,3), dtype=uint8)
for i in range(H):
  for j in range(W):
    out[j,i,0] = arr[i,j,0]
    out[j,i,1] = arr[i,j,1]
    out[j,i,2] = arr[i,j,2]
```

Listing 1: Manual transpose snippet

1.2 Rotate clockwise by 90°

Figure 2: Original image rotated 90 degrees clockwise.

```
# (i, j) -> (j, H-1-i)
out = zeros((W,H,3), dtype=uint8)
for i in range(H):
  for j in range(W):
    ni = j
    nj = H - 1 - i
    out[ni,nj,:] = arr[i,j,:]
```

Listing 2: Manual 90° CW rotation snippet

1.3 Grayscale (average over channels)

Figure 3: Grayscale obtained by integer average of R, G, B.

```
# avg = (r + g + b) // 3
out = zeros((H,W), dtype=uint8)
for i in range(H):
  for j in range(W):
    r,g,b = arr[i,j,0], arr[i,j,1], arr[i,j,2]
    out[i,j] = (int(r) + int(g) + int(b)) // 3
```

Listing 3: Manual average grayscale snippet

1.4 Flip horizontally

Figure 4: Horizontally flipped (mirror) image.

```
# mirror columns: (i,j) -> (i, W-1-j)
out = zeros_like(arr)
for i in range(H):
  for j in range(W):
    out[i, W-1-j, :] = arr[i,j,:]
```

Listing 4: Manual horizontal flip snippet

1.5 Change logo color to green

Figure 5: Recolor red-like pixels to pure green by threshold mask.

```
# simple red threshold: r >= 100, g <= 100, b <= 100
mask = zeros((H,W), dtype=bool)
for i in range(H):
  for j in range(W):
    r,g,b = arr[i,j,:]
    if (r >= 100) and (g <= 100) and (b <= 100):
      mask[i,j] = True

out = arr.copy()
for i in range(H):
  for j in range(W):
    if mask[i,j]:
      out[i,j,:] = [0,255,0]
```

Listing 5: Manual recolor-to-green snippet

1.6 2x2 tiled image (large image)

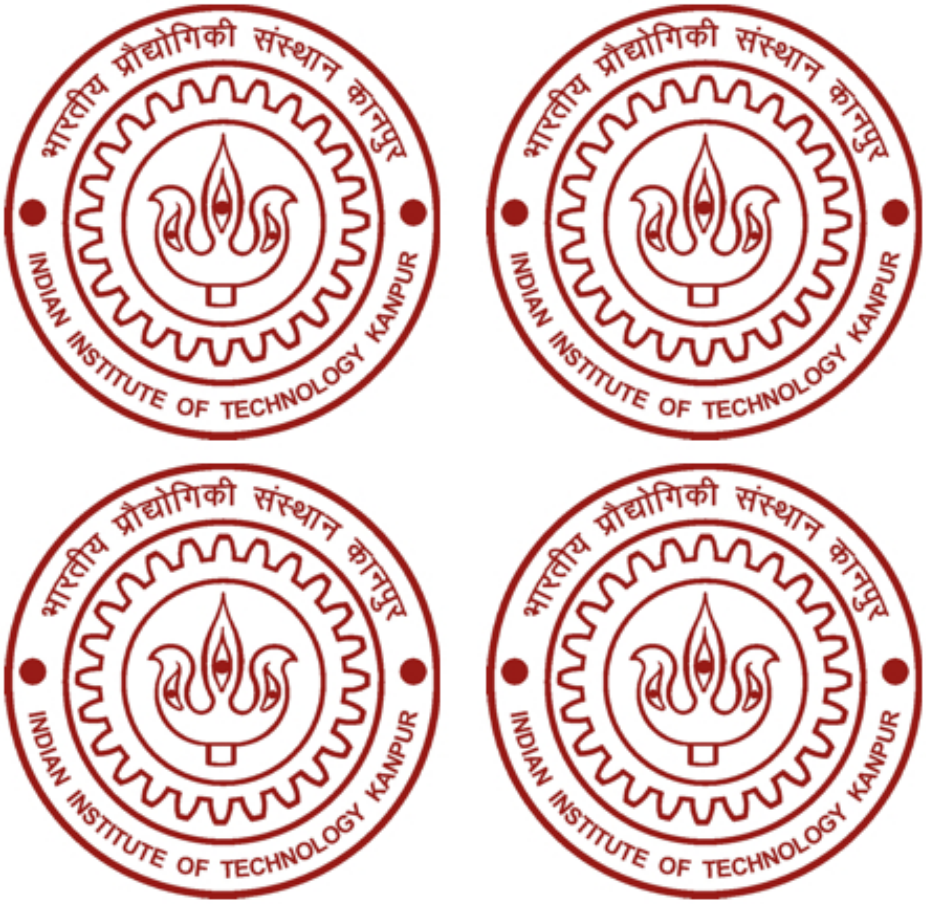

Figure 6: A 2x2 grid; each cell contains the original logo.

```
# place each pixel into four quadrants
out = zeros((2*H,2*W,3), dtype=uint8)
for i in range(H):
  for j in range(W):
    out[i, j, :] = arr[i, j, :]
    out[i, j+W, :] = arr[i, j, :]
    out[i+H, j, :] = arr[i, j, :]
    out[i+H, j+W, :] = arr[i, j, :]
```

Listing 6: Manual 2x2 tile snippet

1.7 Binary version (logo white, background black)

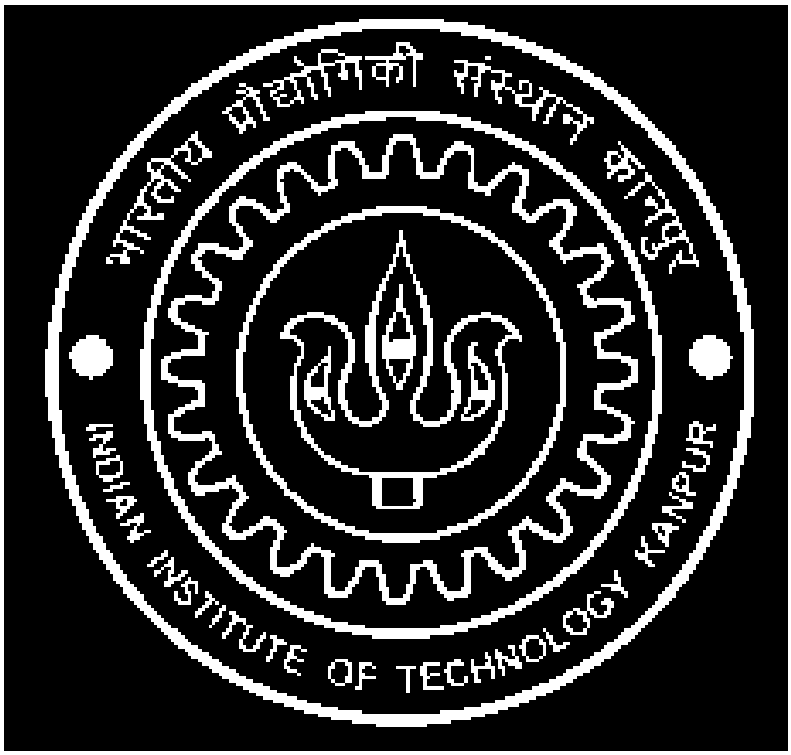

Figure 7: Binary mask: white for logo pixels, black for background.

```
# using previously computed mask (boolean)
out = zeros((H,W), dtype=uint8)
for i in range(H):
  for j in range(W):
    out[i,j] = 255 if mask[i,j] else 0
```

Listing 7: Manual binary mask snippet

1.8 Removed “Three-eyed Trishul”

Figure 8: Trishul area removed by circular mask and small eye wipes (set to white).

```
# main circle center (cx,cy), radius R:
for i in range(H):
  for j in range(W):
    dx = j - cx; dy = i - cy
    if dx*dx + dy*dy <= R*R:
      out[i,j,:] = [255,255,255]

# small eye circles: repeat for each eye center with small radius
```

Listing 8: Manual trishul removal snippet

2 Full Python source

```
"""
process_redlogo_no_builtin_ops.py

Performs required image operations WITHOUT using high-level built-in array ops.
Allowed libs: numpy (for allocation and dtype), PIL only for reading/writing images.
All image transformations are implemented with explicit loops.

Outputs (saved in 'outputs/' folder):
 - redlogo_transpose.png
 - redlogo_rot90_cw.png
 - redlogo_grayscale_avg.png
 - redlogo_flip_horizontal.png
 - redlogo_green_logo.png
 - redlogo_2x2_grid.png
 - redlogo_binary.png
 - redlogo_no_trishul.png
"""

import os
import numpy as np
from PIL import Image

# ---------- Settings ----------
INPUT_PATH = "redlogo.jpg"
OUT_DIR = "outputs"
os.makedirs(OUT_DIR, exist_ok=True)

# color thresholds for detecting red logo (tweak if needed)
RED_MIN = 100
GREEN_MAX = 100
BLUE_MAX = 100

# Trishul removal params (tweak to match your image)
TRISHUL_CENTER_X = 115  # x coordinate (columns)
TRISHUL_CENTER_Y = 109  # y coordinate (rows)
TRISHUL_RADIUS = 45

# ---------- Helpers ----------
def load_image_as_array(path):
    img = Image.open(path).convert("RGB")
    arr = np.array(img, dtype=np.uint8)
    return arr

def save_array_as_image(arr, path, mode="RGB"):
    # arr shape: HxW or HxWx3
    Image.fromarray(arr, mode=mode).save(path)
    print("Saved", path)

# ---------- Implementations without high-level built-ins ----------

def transpose_image_manual(arr):
    H, W = arr.shape[0], arr.shape[1]
    out = np.zeros((W, H, 3), dtype=np.uint8)  # transposed shape
    for i in range(H):
        for j in range(W):
            # place pixel (i,j) -> (j,i)
            out[j, i, 0] = arr[i, j, 0]
            out[j, i, 1] = arr[i, j, 1]
            out[j, i, 2] = arr[i, j, 2]
    return out
```

```python

def rotate_90_cw_manual(arr):
    # rotation: (i, j) -> (j, H-1-i)
    H, W = arr.shape[0], arr.shape[1]
    out = np.zeros((W, H, 3), dtype=np.uint8)
    for i in range(H):
        for j in range(W):
            ni = j
            nj = H - 1 - i
            out[ni, nj, 0] = arr[i, j, 0]
            out[ni, nj, 1] = arr[i, j, 1]
            out[ni, nj, 2] = arr[i, j, 2]
    return out

def grayscale_avg_manual(arr):
    H, W = arr.shape[0], arr.shape[1]
    out = np.zeros((H, W), dtype=np.uint8)
    for i in range(H):
        for j in range(W):
            r = int(arr[i,j,0])
            g = int(arr[i,j,1])
            b = int(arr[i,j,2])
            s = r + g + b
            avg = s // 3   # integer average
            out[i,j] = avg
    return out

def flip_horizontal_manual(arr):
    H, W = arr.shape[0], arr.shape[1]
    out = np.zeros_like(arr)
    for i in range(H):
        for j in range(W):
            out[i, W - 1 - j, 0] = arr[i, j, 0]
            out[i, W - 1 - j, 1] = arr[i, j, 1]
            out[i, W - 1 - j, 2] = arr[i, j, 2]
    return out

def logo_mask_manual(arr, rmin=RED_MIN, gmax=GREEN_MAX, bmax=BLUE_MAX):
    H, W = arr.shape[0], arr.shape[1]
    mask = np.zeros((H, W), dtype=np.bool_)
    for i in range(H):
        for j in range(W):
            r = int(arr[i,j,0])
            g = int(arr[i,j,1])
            b = int(arr[i,j,2])
            if (r >= rmin) and (g <= gmax) and (b <= bmax):
                mask[i,j] = True
    return mask

def recolor_to_green_manual(arr, mask):
    H, W = arr.shape[0], arr.shape[1]
    out = np.zeros_like(arr)
    for i in range(H):
        for j in range(W):
            if mask[i,j]:
                out[i,j,0] = 0
                out[i,j,1] = 255
                out[i,j,2] = 0
            else:
                out[i,j,0] = arr[i,j,0]
                out[i,j,1] = arr[i,j,1]
                out[i,j,2] = arr[i,j,2]
    return out
```

```
124
125 def tile_2x2_manual(arr):
126     H, W = arr.shape[0], arr.shape[1]
127     out = np.zeros((H*2, W*2, 3), dtype=np.uint8)
128     for i in range(H):
129         for j in range(W):
130             # top-left
131             out[i, j, :] = arr[i,j,:]
132             # top-right
133             out[i, j + W, :] = arr[i,j,:]
134             # bottom-left
135             out[i + H, j, :] = arr[i,j,:]
136             # bottom-right
137             out[i + H, j + W, :] = arr[i,j,:]
138     return out
139
140 def binary_logo_manual(mask):
141     # mask: boolean HxW
142     H, W = mask.shape
143     out = np.zeros((H, W), dtype=np.uint8)
144     for i in range(H):
145         for j in range(W):
146             out[i,j] = 255 if mask[i,j] else 0
147     return out
148
149 def remove_trishul_manual(arr, center_x=TRISHUL_CENTER_X, center_y=TRISHUL_CENTER_Y
    , radius=TRISHUL_RADIUS):
150     H, W = arr.shape[0], arr.shape[1]
151     out = np.zeros_like(arr)
152     # start by copying original
153     for i in range(H):
154         for j in range(W):
155             out[i,j,0] = arr[i,j,0]
156             out[i,j,1] = arr[i,j,1]
157             out[i,j,2] = arr[i,j,2]
158     radius2 = radius * radius
159     # remove main circular region
160     for i in range(H):
161         for j in range(W):
162             dx = j - center_x
163             dy = i - center_y
164             dist2 = dx*dx + dy*dy
165             if dist2 <= radius2:
166                 out[i,j,0] = 255
167                 out[i,j,1] = 255
168                 out[i,j,2] = 255
169     # remove three small eyes by small circular wipes (offsets relative to center)
170     eye_offsets = [(-8, -6), (0, -9), (8, -6)]   # tweak if necessary
171     eye_radius = max(3, radius // 8)
172     eye_r2 = eye_radius * eye_radius
173     for ex_off, ey_off in eye_offsets:
174         ex = center_x + ex_off
175         ey = center_y + ey_off
176         for i in range(H):
177             for j in range(W):
178                 dx = j - ex
179                 dy = i - ey
180                 if dx*dx + dy*dy <= eye_r2:
181                     out[i,j,0] = 255
182                     out[i,j,1] = 255
183                     out[i,j,2] = 255
184     return out
185
```

```python
# ---------- Main ----------
def main():
    if not os.path.exists(INPUT_PATH):
        raise FileNotFoundError("Place 'redlogo.jpg' in the same folder as this script or update INPUT_PATH.")
    arr = load_image_as_array(INPUT_PATH)
    H, W = arr.shape[0], arr.shape[1]
    print("Loaded image:", INPUT_PATH, "size:", W, "x", H)

    # 1) Transpose
    trans = transpose_image_manual(arr)
    save_array_as_image(trans, os.path.join(OUT_DIR, "redlogo_transpose.png"))

    # 2) Rotate 90 CW
    rot90 = rotate_90_cw_manual(arr)
    save_array_as_image(rot90, os.path.join(OUT_DIR, "redlogo_rot90_cw.png"))

    # 3) Grayscale (average)
    gray = grayscale_avg_manual(arr)
    save_array_as_image(gray, os.path.join(OUT_DIR, "redlogo_grayscale_avg.png"), mode="L")

    # 4) Flip horizontally
    flipped = flip_horizontal_manual(arr)
    save_array_as_image(flipped, os.path.join(OUT_DIR, "redlogo_flip_horizontal.png"))

    # 5) Change logo color to green
    mask = logo_mask_manual(arr)
    green = recolor_to_green_manual(arr, mask)
    save_array_as_image(green, os.path.join(OUT_DIR, "redlogo_green_logo.png"))

    # 6) 2x2 grid
    tiled = tile_2x2_manual(arr)
    save_array_as_image(tiled, os.path.join(OUT_DIR, "redlogo_2x2_grid.png"))

    # 7) Binary version
    binary = binary_logo_manual(mask)
    save_array_as_image(binary, os.path.join(OUT_DIR, "redlogo_binary.png"), mode="L")

    # 8) Remove trishul
    removed = remove_trishul_manual(arr)
    save_array_as_image(removed, os.path.join(OUT_DIR, "redlogo_no_trishul.png"))

    print("All done. Outputs are in:", OUT_DIR)

if __name__ == "__main__":
    main()
```

Listing 9: Manual 2x2 tile snippet