A3: locality

Design Document

CSC 411 - Noah Daniels

Marceline Kelly and Nicolas Leffray

October 20, 2023

What invariant properties should hold during the solution of the problem?

- Every pixel in the source image should correspond to one and only one pixel in the destination image.

What are the major components of your program, and what are their interfaces?

Data structures

- `src`: an `Array2` built from the input image.
- `dest`: an `Array2` containing the transformed output image.

Functions

Each operation (except for the trivial "rotate by 0°") will have its own function. Each function will simply accept a source `Array2` and output a transformed `Array2`.

- `rotate90`: translates the pixel at (i, j) to $(h - j - 1, i)$
- `rotate180`: translates the pixel at (i, j) to $(w - i - 1, h - j - 1)$

- `rotate270`: translates the pixel at (i, j) to $(w - i - 1, j)$
- `flip_horizontal`: translates the pixel at (i, j) to $(w - i, j)$
- `flip_vertical`: translates the pixel at (i, j) to $(i, h - j)$
- `transpose`: translates the pixel at (i, j) to (j, i)

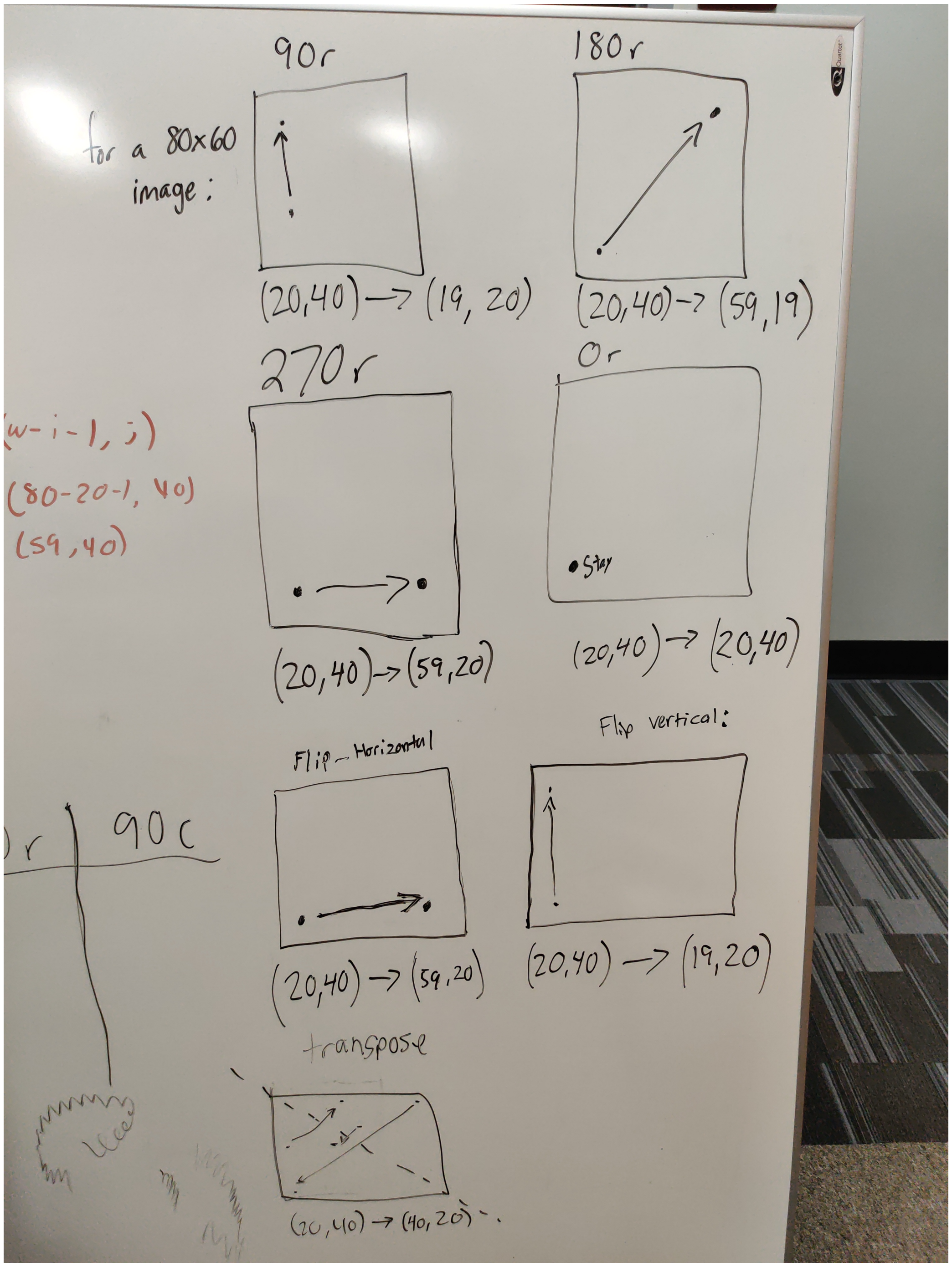
What test cases will you use to convince yourself that your program works?

- Choose a pixel from the source image using a given coordinate set. Then, use the appropriate formula to determine the corresponding index in the transformed image. After performing the transformation, test if the brightness value at each of the coordinate sets is the same.
  - For maximum efficacy, each pixel should have a unique brightness value in the source image. This prevents cases where each pixel happens to have the same brightness, despite the image being transformed incorrectly.
  - Of course, each operation would have its own test case that incorporates this idea.

What arguments will you use to convince a skeptical audience that your program works?

Each transformation involves a sequence of arithmetic operations to produce an output image. These operations are easy to independently verify by anyone reviewing the program. Further, the program will incorporate a suite of test cases to computationally verify the output of each transformation. These test cases verify that the primary invariant holds true for each transformation.

Visual explanation of transform functions


90r
180r
for a 80x60 image:
(20,40) → (19, 20)
(20,40) → (59,19)
270r
0r
(w-i-1, j)
(80-20-1, 40)
(59,40)
Stay
(20,40) → (59,20)
(20,40) → (20,40)
Flip - Horizontal
Flip vertical:
r
90c
(20,40) → (59,20)
(20,40) → (19,20)
transpose
(20,40) → (40,20)