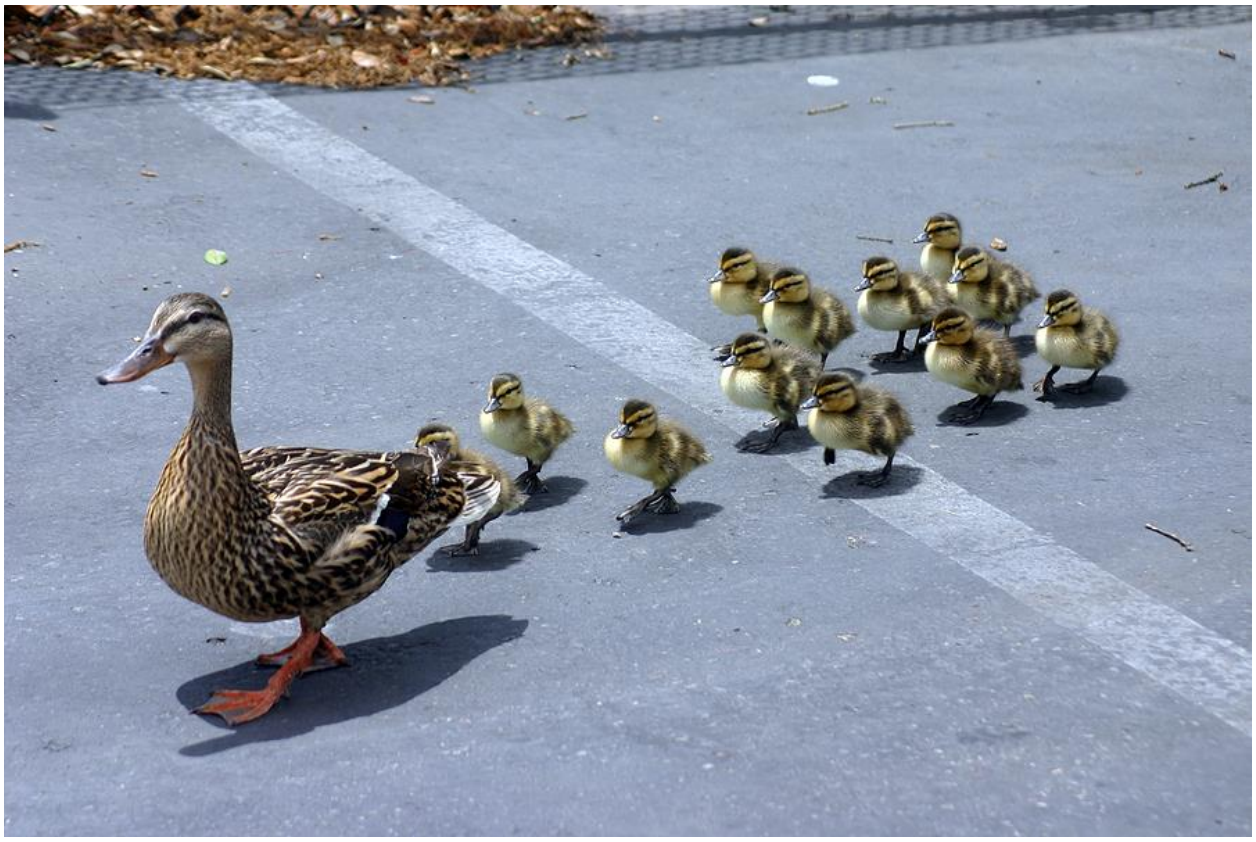
Computer Vision Methods for Medical & Biomedical Images Project

Haejong Dong
2292191
s2haejong@gmail.com

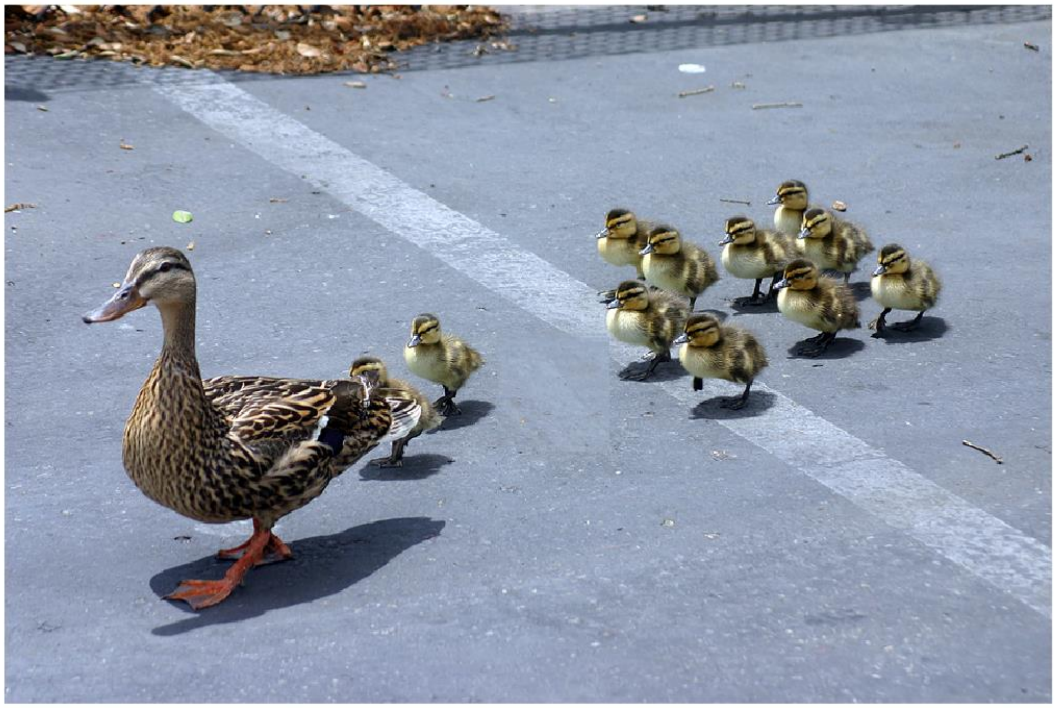
Section 1

I have built an in-printing application for the project work.
Some features I found while testing.

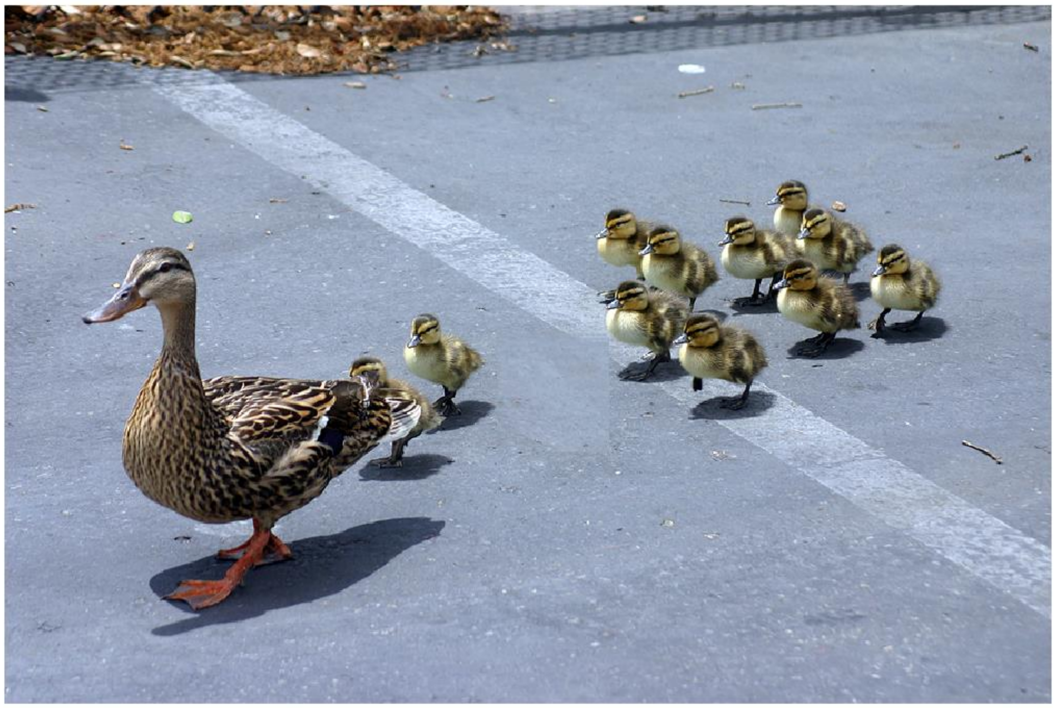
- The application doesn't work for every images
- Result often to be better with small size images
- I didn't follow what's known as state of the arts technique for this task
- I found was that it is very hard to make so that it works generally. Even using the same image I had to modify the code to operate the processing at different part of the image
- Some cases code works with less conditions some others not.
- Filling a large area was harder than a small area

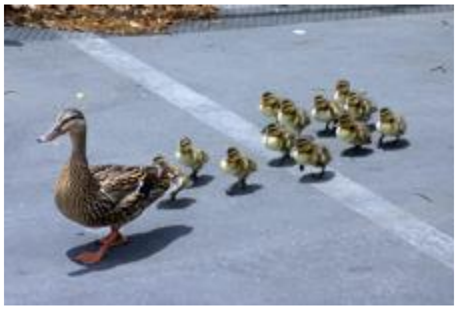

(original image)

Result 1.

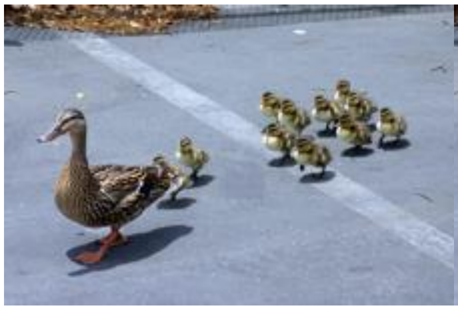


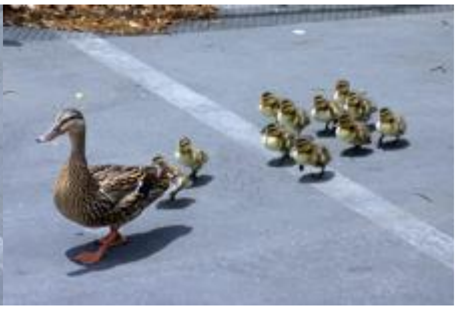
Result 2



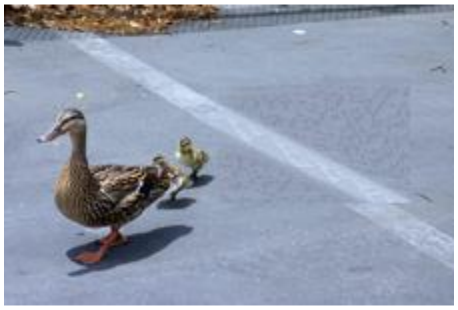

(original, small sized)

Result 1. Result 2.





Result 3.



How it works

1. First it takes 6 parameters.
    a. Value that measures size of area that it's going to search to find similar pattern
    b. Threshold value for the condition when selecting pixel values from a selected grid, it computes if the pixel values with which it intends to replace is relevant to the context of the original grid.
    c. Steps that it wishes to jump over at each computing when searching a similar pattern, to reduce computing time
    d. Weighted centric, I observed when context of the image switch to another, for instance, when the grid approaches to part of a new object that is missing its original shape, it happened to be appeared through the pixel gradually thus in the algorithm used for this application had most of times failed to match that point

timely. So I added this condition that if the grid finds a very different pixel value at the center of it, I assume that it is a sign of changing context so the condition allows having more weight on that specific value to influence following-up values

e. Weight value in which context that is seem to be getting darker
f. Weight value in which context that is seem to be getting brighter

2. Second, there are five additional functions that work together with the main function
    a. isOutBoundary() : measures if the selected search area is going to collide with image boundary
    b. isRGBZero() : check if there is 0 value across RGB within the grid
    c. isRelevant() : find if the pixel values in the selected grid are not too arbitrary
    d. isInsideGrid() : check if the grid spans the empty grid. This is used generally to find to identify if current position is inside of the empty grid or not.
    e. ifWeighted() : check if the context is about to change (if so, to give more weight)

3. When an image with empty (RGB = 255 or 0) grid to be filled is loaded, first method will fill the empty grid with all 0 values to differentiate from normal values.
4. And start searching until the searching-grid finds the empty grid, isRGBZero() or isInsideGrid() methods are used for this.
5. When it finds, isOutBoundary() measures with which sides of the image boundaries would collide with pattern-search-area then the area is allocated accordingly by the given parameter
6. Then start searching in that area to find similar pattern with the grid that needs to fill the empty value
7. The searching-grid ignore area that is upon another empty grid or area where has been filled lately to avoid repeating the same pattern by referring grid one pixel next to.
8. The selection is done based on subtraction between grids one that is searching value and the other that is found
9. Once it found a relevant grid, stores the data until better one found.
10. When searching is done through the search area, finally replace the empty value by the value stored until that moment.
11. And repeat until the end of empty grid
12. After all is said and done, a restored image is produced as an output image