Object detection and counting in images based on template object

Team Members:
Roshan Prashant Bara
Shreyansh Singh
Rohit Janbandhu

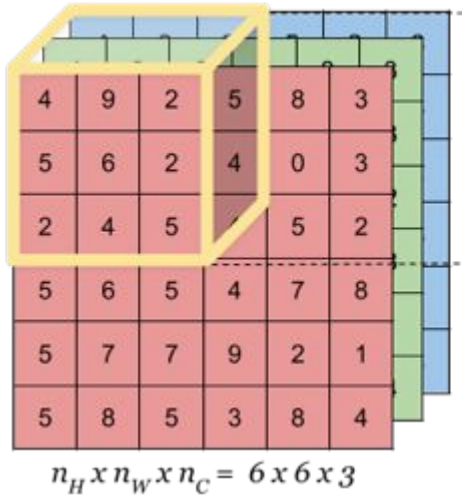


Image is a 2D signal represented as f(x, y), where x and y are coordinates of a 2D plane.

The range of values that f(x, y) outputs is [0, 255]. 255 makes pixel white and 0 makes it black.

Each RGB image is stored in an array of shape (x dims, y dims, 3) where x dims and y dims are number of pixels in image horizontally and vertically respectively and 3 layers are R, G, B layers.

Block Matching Algorithm

- In Block Matching Algorithm we divide an image into multiple macroblocks.
- Then we match each of them with a reference block.
- For best matching we can:-
  - maximise the correlation function
  - minimise the Mean Squared Error (MSE)
  - minimise the Mean Absolute Difference (MAD)
- The most basic idea is to compare it will all the blocks and find the best matching, but this is computationally very expensive.
- For optimization we can define some search window and search will only be done inside it.
- This can be further optimized by selective searching. There are many algorithms implementing this like Three Step Search, Diamond Search, etc.

Implementation Details of the Code for Milestone 2:

- We first read the image file and object file in GRAYSCALE mode using CV2 library.
- Then using the mse function, we compute the error_matrix.
- The mse function calculates the Mean Squared Error by comparing the image and object file in a sliding window with the dimensions of the sliding window being the same as object dimensions and stores it in error_matrix and each position in the error_matrix corresponds to the left top pixel of the sliding window.
- Now , having got the error_matrix, we select the pixel having mean squared error less than or equal to threshold in a region of the image and mark the boundary of the matching region. We also take care that overlapping regions are not marked by selecting the minimum MSE from a low MSE cluster in the neighbourhood of matching region.
- The matching regions bounded by a black rectangle on the image is then displayed in a separate window.
- The number of matching regions in the image file is then displayed in the console.

Sample Output 1

```
PS C:\Users\Roshan\Desktop\ELL_Project Milestone 2> python .\main.py
Enter image filename: t1.png
Enter object filename: t2.png
Number of matching objects in image:  2
```

t1.png

t2.png

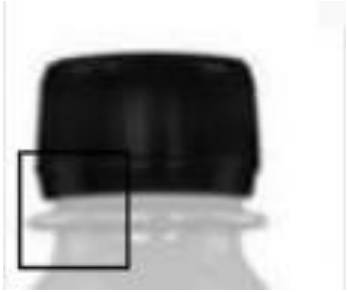
Image file

Object file

Output

Image file

Object file



Output

Image file

Object file

Output

Image file

Object file

Output

Gaussian Pyramid

- An image pyramid is a collection of images- all arises from a single original image - that are successively down-samples until some desired stopping point is reached.
- Gaussian pyramid is one of the common kind of image pyramid
- Higher level (Low resolution) in a Gaussian Pyramid is formed by removing consecutive rows and column in Lower level (higher resolution) image. Then each pixel in higher level is formed by the contribution from pixels in underlying level with gaussian weights. By doing so, a M x N image becomes a M/2 x N/2 image.
- So its area refuses to ¼ the original area.
- The same pattern continuous as we go upper in pyramid
- Similarity while expanding area becomes 4 times in each level.

Implementation Details of the Code for Milestone 3:

- We first read the image file and object file in the GRAYSCALE mode using the CV2 library.
- Then we use the getflag function to get the list of potential matchings.
- We then use MSE function to generate the error matrix and based on the error matrix, we find the potential matchings.
- The getflag is called for the original image, twice downsized object and four-times downsized object. Similarly to capture the larger dimension matchings the getflags function is run using twice upsized object and four-times upsized object
- Then using the remove_duplicate function, we get the final list of non overlapping coordinates.
- The remove_duplicate function compares the the overlapping regions and selects the minimum MSE from a low MSE cluster in the neighbourhood of matching region.
- The matching regions bounded by a black rectangle on the image is then displayed in a separate window.
- The number of matching regions in the image file is then displayed in the console.

Sample output 1

```
OUTPUT   DEBUG CONSOLE   PROBLEMS   TERMINAL

PS C:\Users\ROHIT\Desktop\ELL_Project Milestone 3> python -u "c:\Users\ROHIT\Desktop\ELL_Project Milestone 3\newmain (1).py"
Enter image filename: gg.jpeg
Enter object filename: gg2.png
Number of matching objects in image:  3
```

gg.jpeg

gg2.png

Output

Sample output 2

```
OUTPUT   DEBUG CONSOLE   PROBLEMS   TERMINAL

PS C:\Users\ROHIT\Desktop\ELL_Project Milestone 3> python -u "c:\Users\ROHIT\Desktop\ELL_Project Milestone 3\newmain (1).py"
Enter image filename: gg.jpeg
Enter object filename: gg3.png
Number of matching objects in image:  3
PS C:\Users\ROHIT\Desktop\ELL_Project Milestone 3>
```

gg.jpeg

gg3.png

Output