Background Subtraction

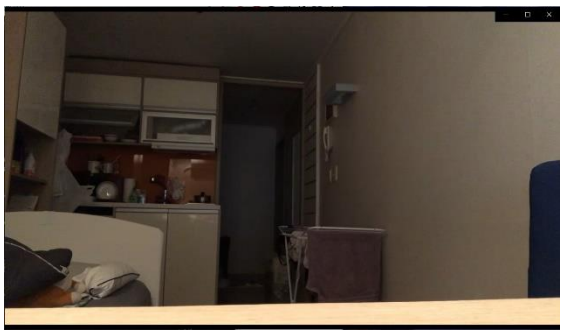
20221024 민정현

https://github.com/jeonghyeoni/TP1_digital_matting

Explanation of the algorithm

Simply put, by obtaining the absolute difference between the background and each frame, pixels with a difference less than the threshold are considered as the background, and greater than the threshold as the foreground.

Of course, the camera should be fixed so that the background does not change throughout the video.

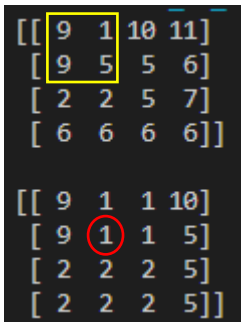


Background image

a frame of the video

Also, I applied erosion and dilation to all frames of the video.

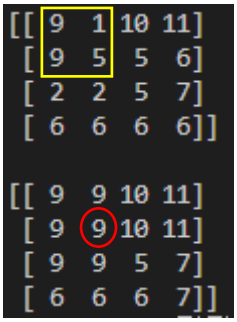
Erosion and Dilation



The upper array is a 4x4 size image, and the lower array is the result of eroding the image with a 2x2 kernel.

cv2.erosion() uses a kernel to scan the image and changes the value of the pixels to the smallest number in the kernel.

"It is useful for removing small white noises and used to detach two connected objects."[1]



As opposed to cv2.erosion(), cv2.dilation() selects the largest value among the numbers in the kernel.

"In cases like noise removal, erosion is followed by dilation. Because, erosion removes white noises, but it also shrinks our object. So we dilate it. Since noise is gone, they won't come back, but our object area increases. It is also useful in joining broken parts of an object."[2]

[1,2] Pratima Upadhyay ,"Erosion and Dilation of images using OpenCV in python",<GreeksforGreeks>(2022.7.29)(2022.11.30 connect) https://www.geeksforgeeks.org/erosion-dilation-images-using-opencv-python/

Process of the project

```
lower10 = final_difference[-1]<=10
frame2[lower10] = im[lower10] # im is new background
```

All subsequent results are the result of replacing a pixel with a difference less than 10 with a pixel in the new image as code above.

1. Calculate the absolute difference

There is a lot of noise in the results because the brightness of the background keeps changing slightly.

2. Apply Gaussian blur effect

Removing small noises was an easy part. Just apply blur to the background image and each frame, small noises are easily removed.

3. blur + erosion(mask: 3x3) three times

Erosion was performed to remove noise from the background without damaging the foreground too much.

4. blur + erosion(3x3) three times + dilation(3x3) ten times

Dilate on the previous attempt to reattach the erased part from the foreground

5. blur + erosion(3x3) three times + dilation(3x3~9x9) once each + dilation(9x9) three times

There are still parts that have not been filled in the foreground, so I have increased the size of the mask and filled it more.

6. blur + erosion(3x3) three times + dilation(3x3~12x12) once each

I think it's not enough yet, so I filled it up more.

7. blur + erosion(3x3) three times + dilation(3x3~12x12) once each + erosion(10x10, 7x7, 4x4) three times each

I erode again to erase the border part included in the foreground.
However, as you can see in the sixth picture, there was a square-shaped area where the face was erased. A very small dot that was not found in the previous attempt caused this result.

8. blur + erosion(3x3) three times + dilation(3x3~12x12) once each + dilation(5x5) once + erosion(10x10, 7x7, 4x4, 5x5) three times each

I supplemented the dilations to fill in the small dots.

9. blur + erosion(3x3) twice, (5x5) once + dilation(3x3~12x12) once each + dilation(7x7) three times + erosion(10x10, 7x7, 4x4) three times each, (5x5) seven times

Although erosion eliminates noise, some of the foreground is erased, and dilation fills the foreground, but the background is also filled, so the appropriate kernel size and the appropriate number of iterations were needed. The process of finding it was the hardest part.

Although the results were improved to some extent by applying erosion and dilation, it took a lot of time to find the appropriate kernel size and number of applications by continuously repeating code execution. And as a result, this is hard coding.

To solve this problem, I applied erosion to pixels with a difference of less than a threshold and applied dilation to pixels with a difference of more than a threshold, but in the end, the results were similar to before.

I thought about solving the problem through CNN or DNN, but I didn't try because I didn't have enough time and my head didn't follow me. I want to try it when I have time personally next time.

Applying another video to my code

blur + erosion(7x7, 5x5, 3x3) once + dilation(5x5) three times

Original video | Result

Video source: https://github.com/PeterL1n/BackgroundMattingV2

To apply my algorithm to other videos, I once again had to set the appropriate kernel size and number of iterations.

However, this video had a clear contrast between the background and the foreground, and the background was cleaner than the video I took, so it took much less effort and the result is much better.

Result Comparison against other's model

I put the video I took into Peter L1n's Background Matting V2 model.

Peter L1n's Background Matting V2 mine


S