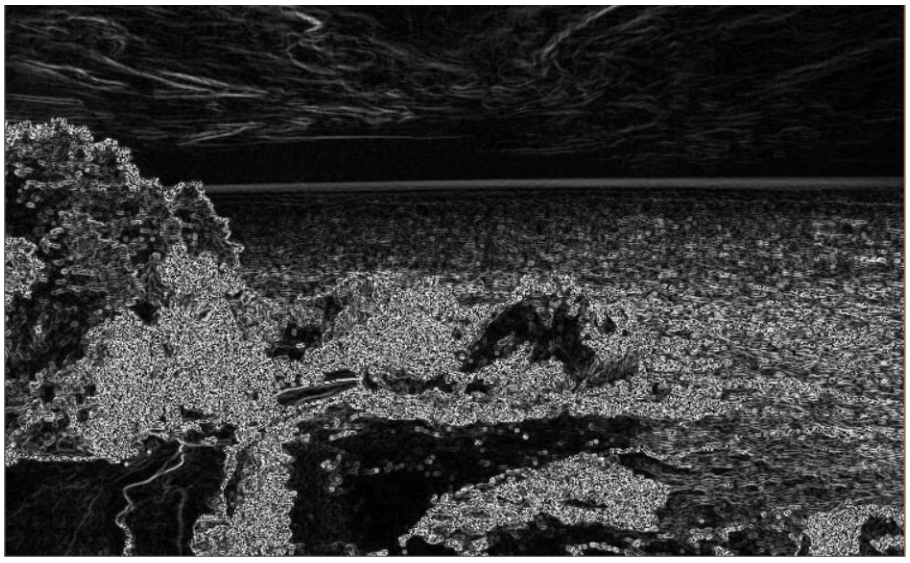
Seam Carving and Template Matching

Siddharth Khera (2015MT60567)
Ankur Sharma (2015CS50278)

Test Image (751 x 500)

Energy Image

Energy function is a simple energy function implemented by summing up the absolute modulus of the **sobel operator** applied along the x and y axis. There can be other energy functions too which could have been used too but this one works almost most of the times.

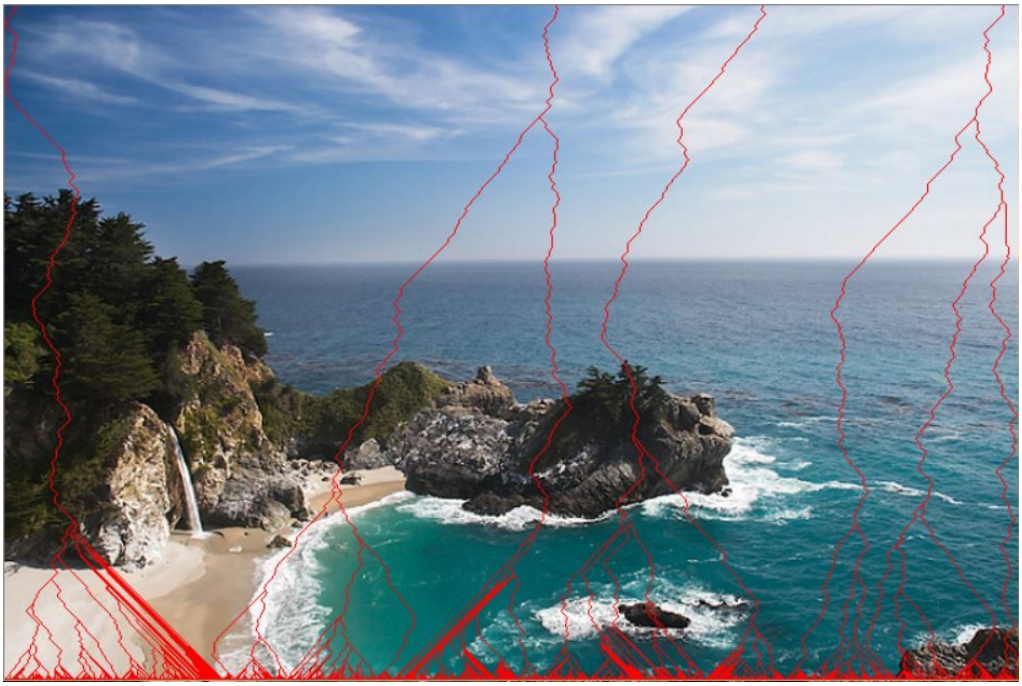


HIGHLIGHTED SEAMS

By constructing the dp and path table, we were able to identify seams along horizontal or vertical direction, and the same were highlighted as shown below.

Horizontal Seams

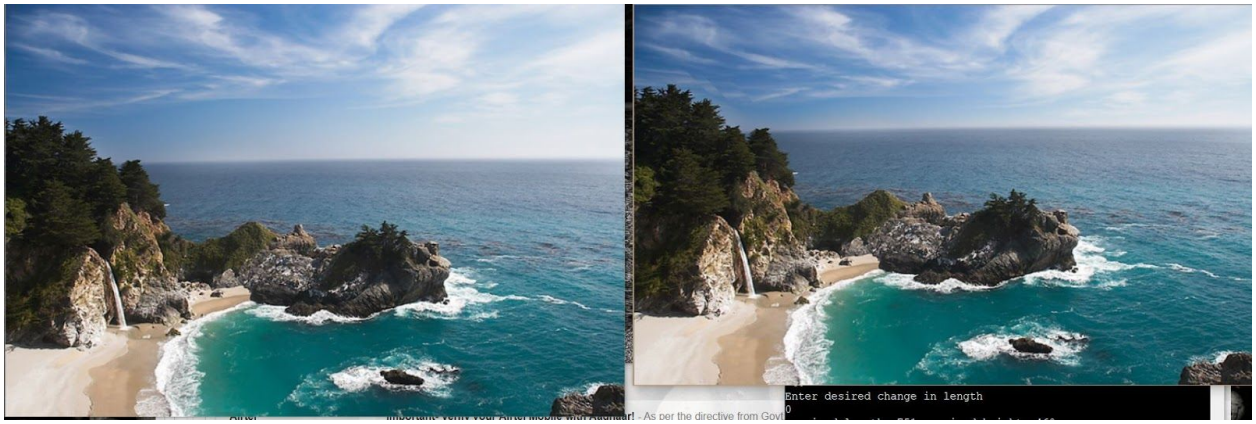
Vertical Seams



SEAM DELETION

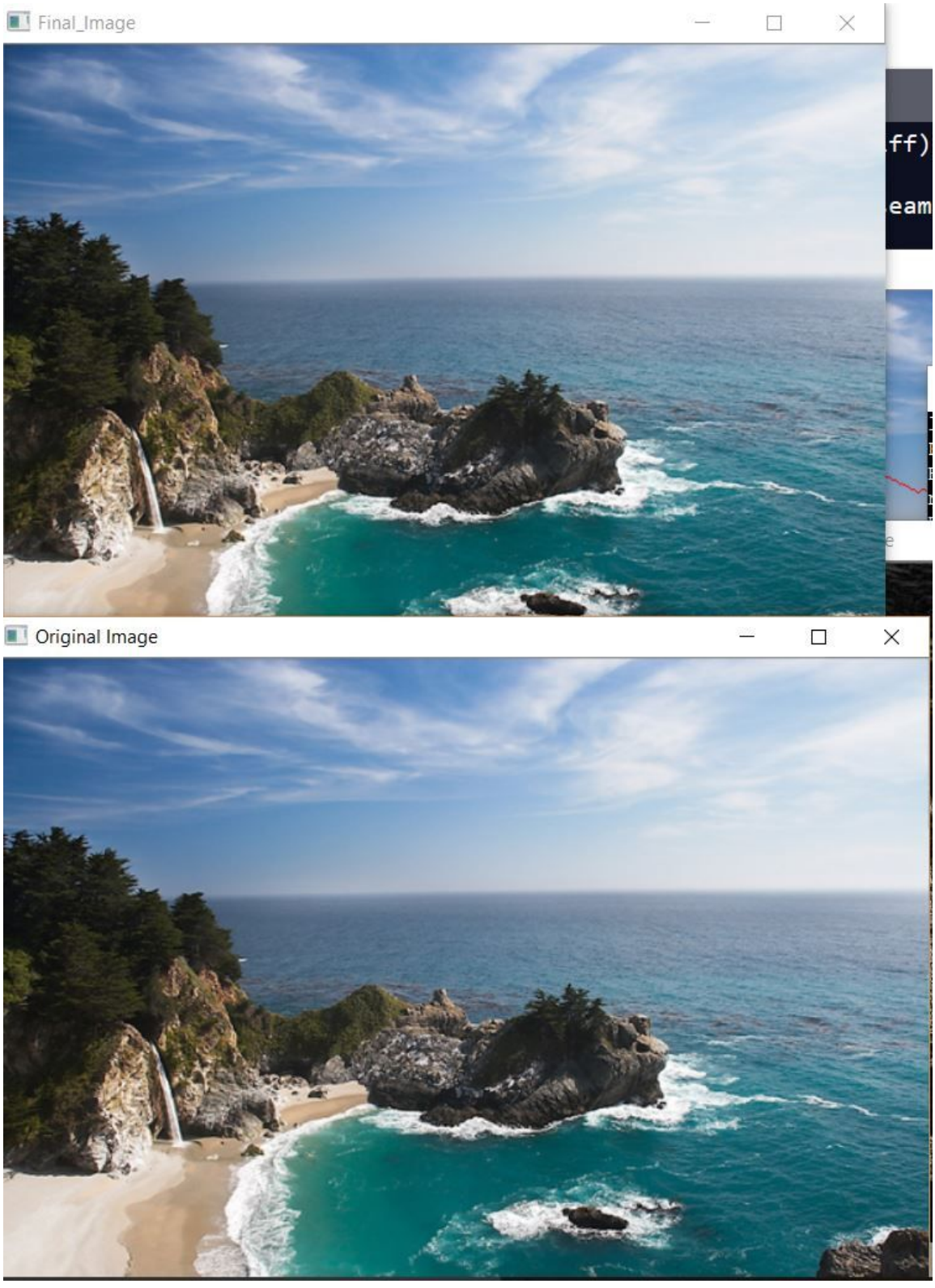
Horizontal Seam Removal (711 x 500)

```
(openCVenv) C:\Users\Siddharth Khera\PycharmProjects\Assignment-SeamCarving>python source.py
length= 751 height= 500
Do you want to remove objects from the image? [y/n]
Enter
n
Enter desired change in height
-40
Enter desired change in length
0
required length= 751 required height= 460
Reducing Image Height. 40 operations required                    1 2 3 4 5 6 7 8 9 10 11 12 13 14
 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
```



Vertical Seam Removal (751 x 460)

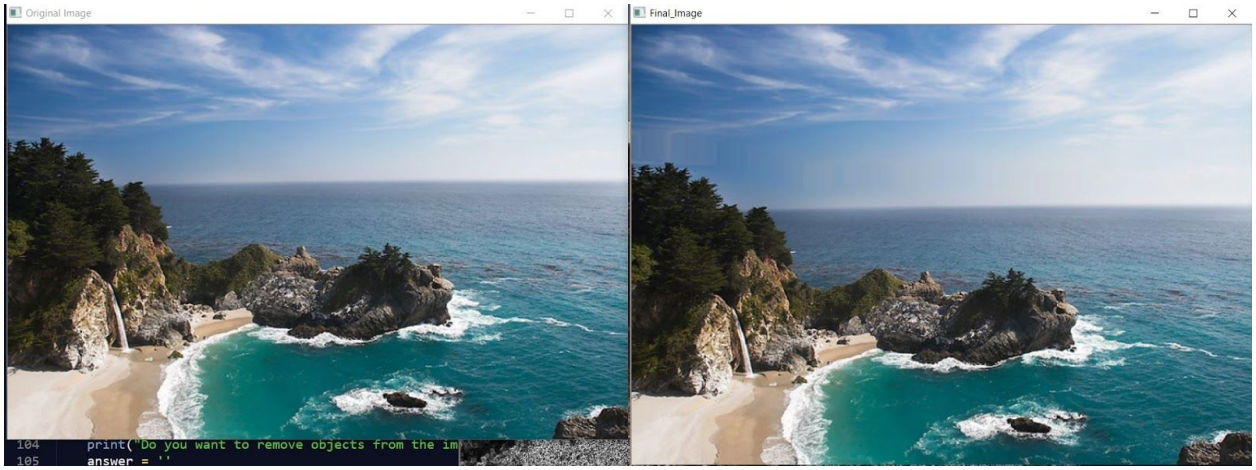
```
(openCVenv) C:\Users\Siddharth Khera\PycharmProjects\Assignment-SeamCarving>python source.py
length= 751 height= 500
Do you want to remove objects from the image? [y/n]
Enter
n
Enter desired change in height
0
Enter desired change in length
-40
required length= 711 required height= 500
Reducing Image Length. 40 operations required
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36 37 38 39 40
```


Final_Image
Original Image

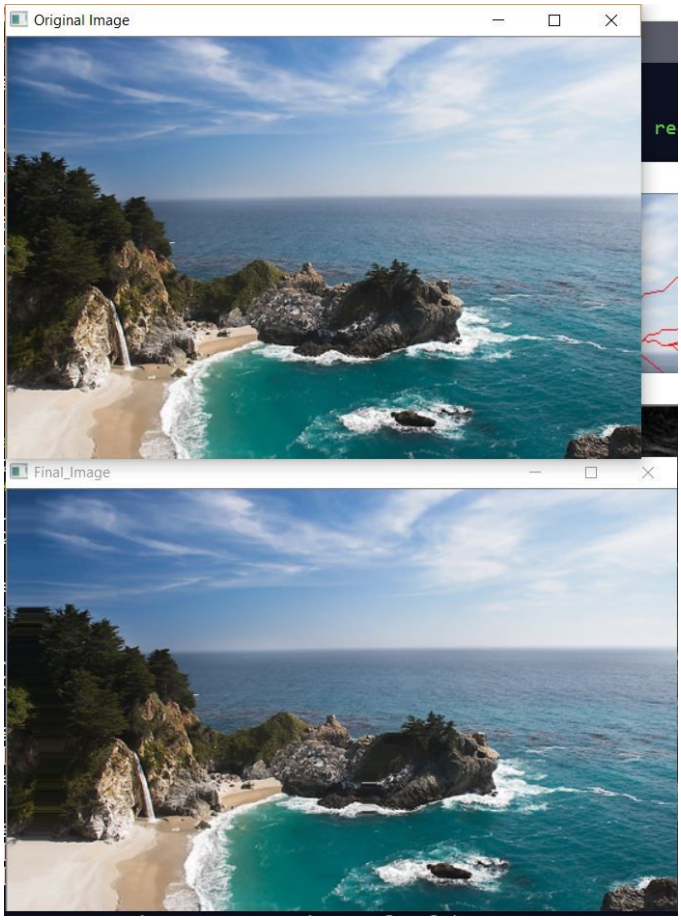
SEAM INSERTION

There are cases of visual artifacts in the inserted image, when most of the seams converge into one converged team, and hence we realise that choosing a better batch size or maybe improving the energy function can be one possible alternative way out of the problem. However, for some cases the conventional method of inserting seam also works decently.

HORIZONTAL SEAM INSERTION (BATCH SIZE: 30, SEAMS INSERTED: 30)



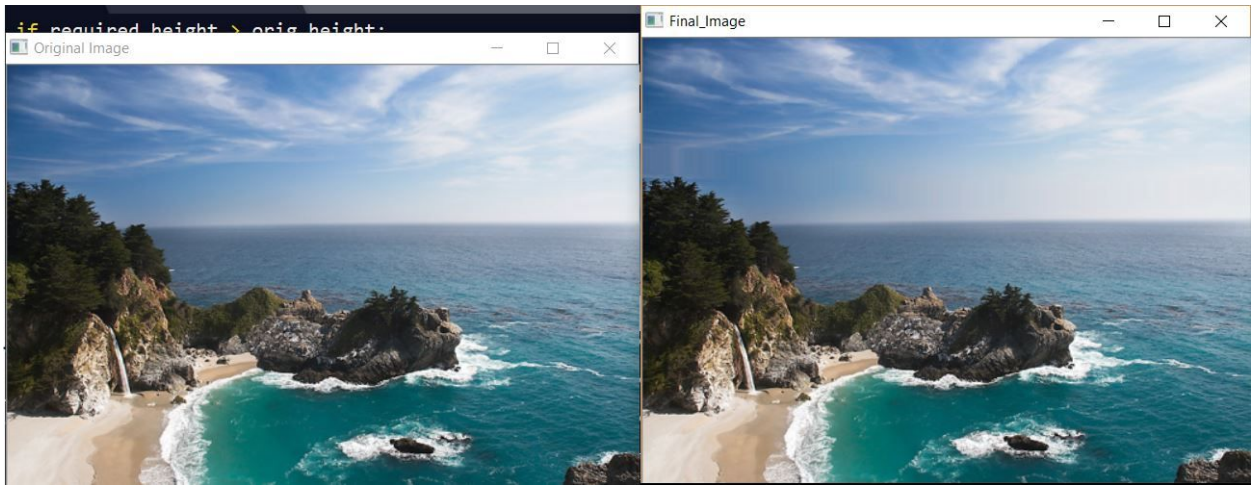
VERTICAL SEAM INSERTION(BATCH SIZE: 20, SEAMS INSERTED: 30)



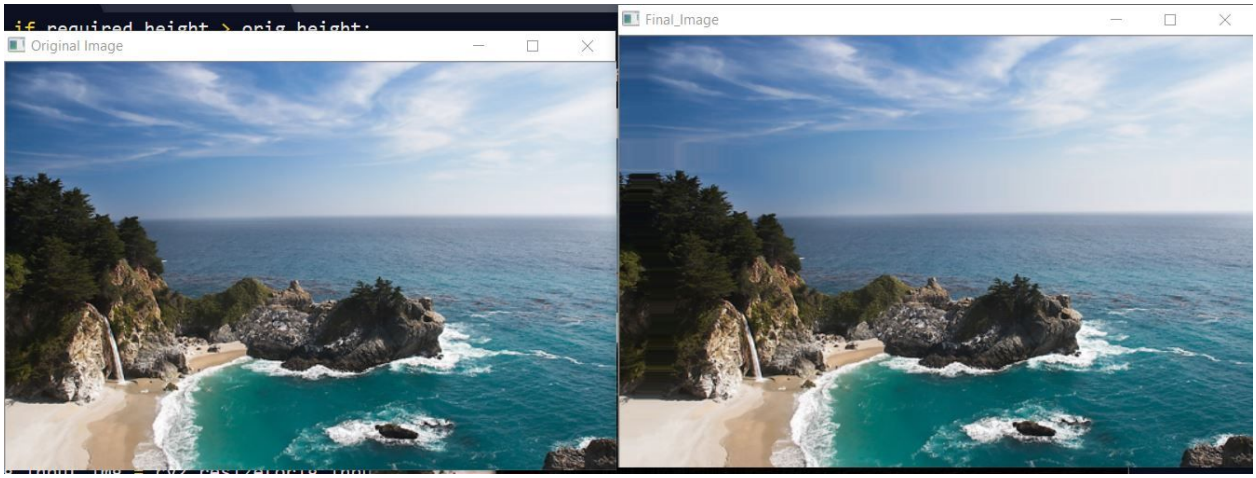
RETARGETING IMAGE SIZE (CHANGING DIMENSIONS)

751 x 500 -> 739 x 488 (Both are decreased by 12)

751 x 500 -> 771 x 480 (Height increased by 20, Length decreased by 20)

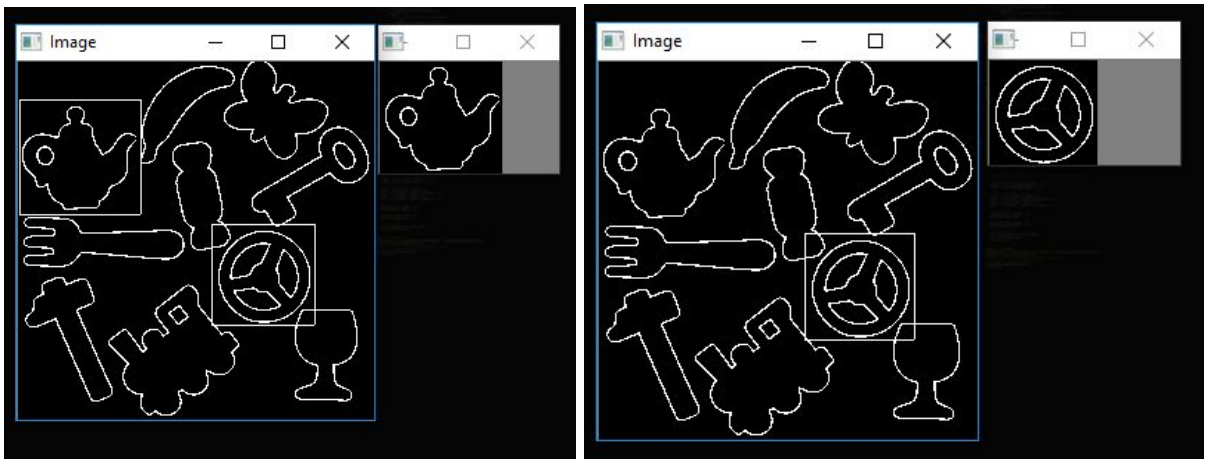


751 x 500 -> 771 x 520 (Both are increased by 20)



OBJECT REMOVAL

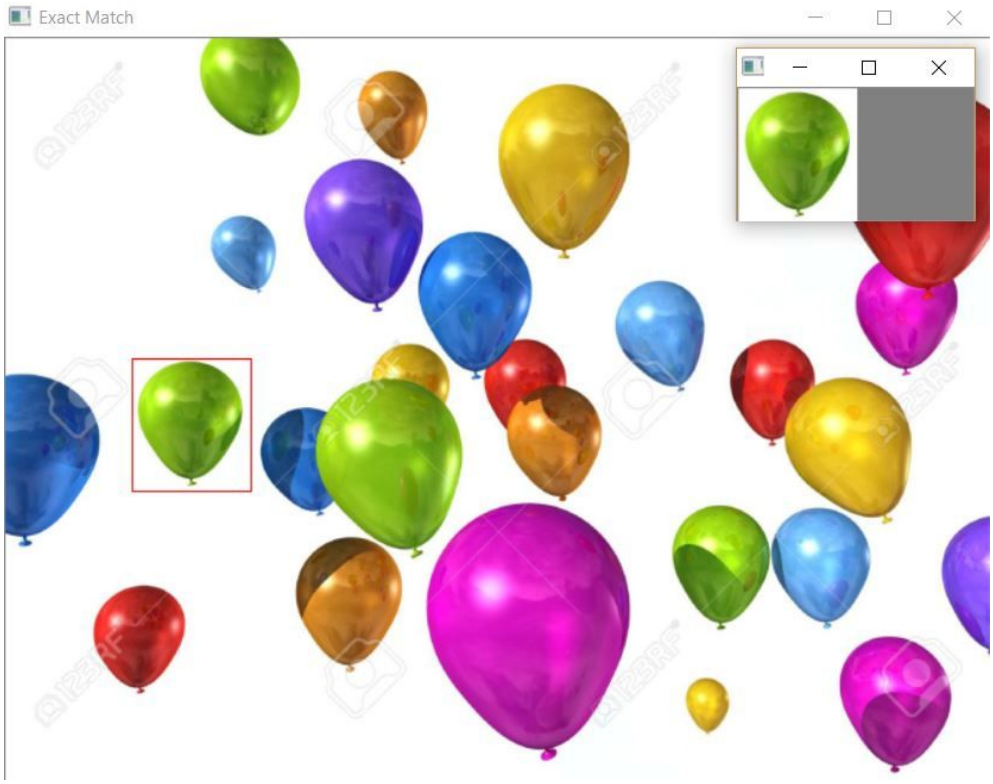
1.) Generalised Hough Transform :

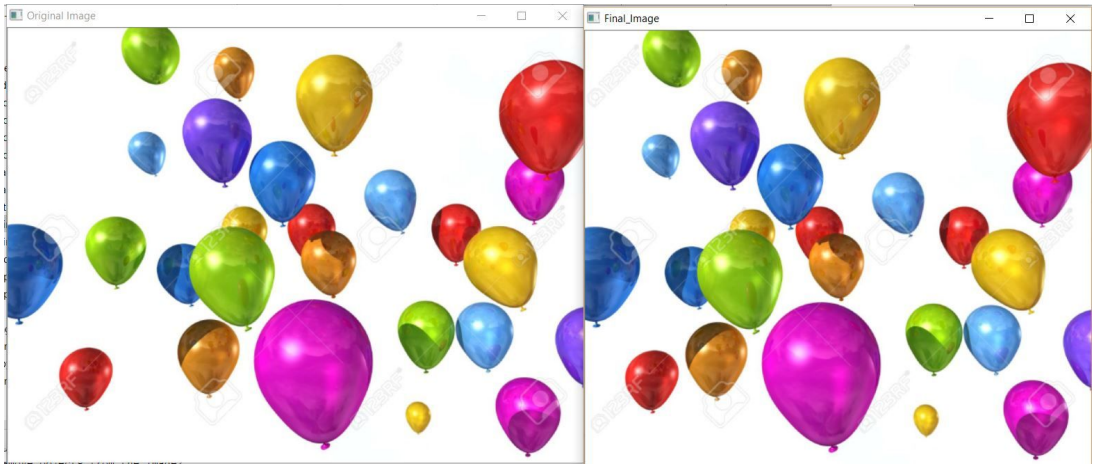


The code for GHT has been modified from an existing code on github to make this cv2 compatible. We tried testing GHT on a number of sample figures but the inaccuracy and inconsistency in the results of GHT make it unscalable, and hence we adopted the Phase correlation method for the same.

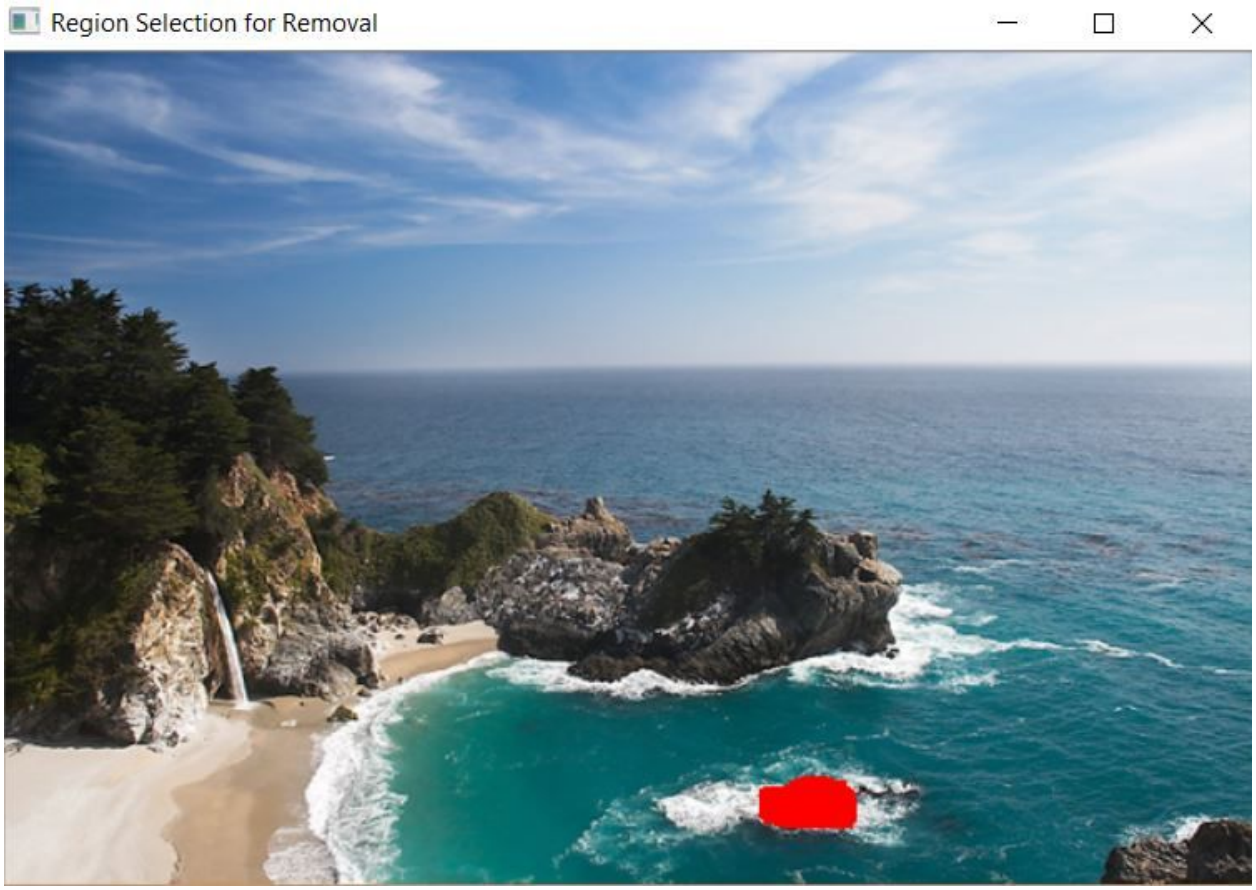
2.) Finding correlations in the Fourier Domain :

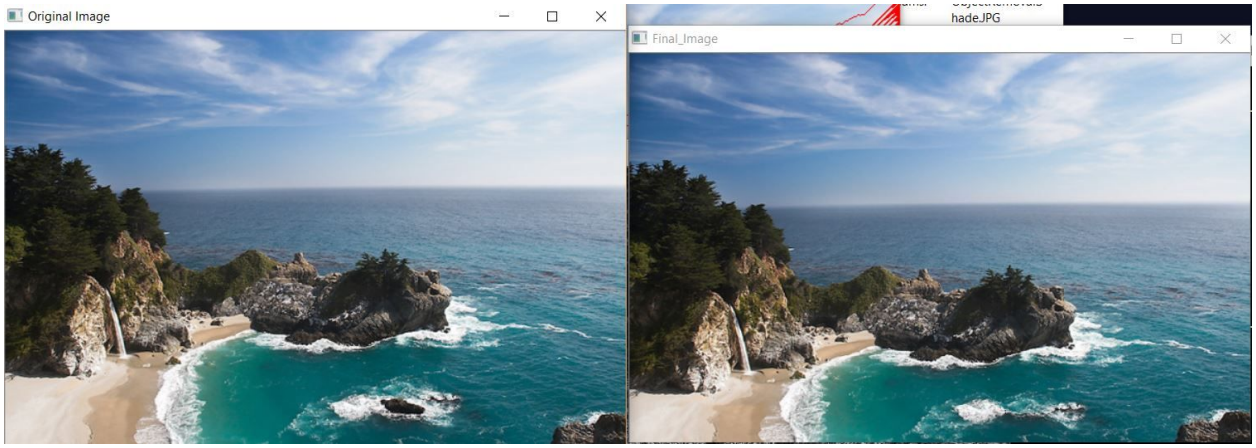
This thing is implemented by finding phase correlation by computing cross power spectrum of two signals in the fourier domain, and taking the maximum of the computed value of the real component of inverse fourier transform.





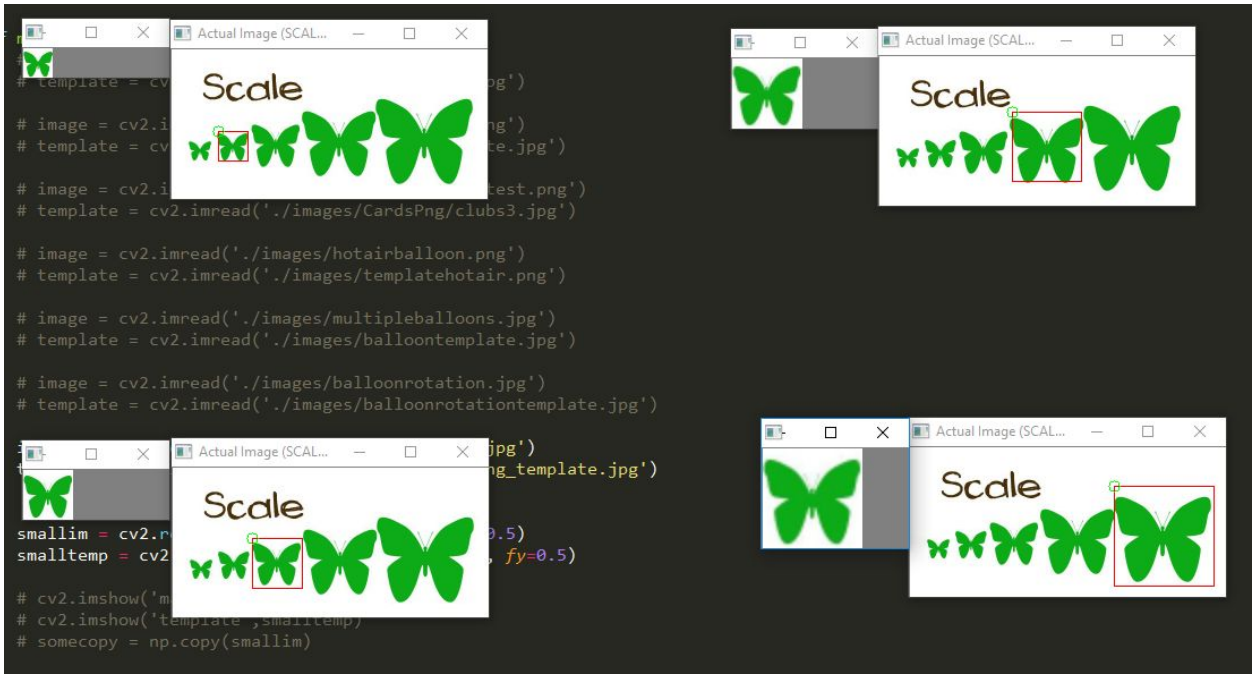
3.) Manual Interface: We have designed a manual user-friendly interface in which the user can manually mark the object(red pixels) to remove by holding and dragging the mouse over the region to select, and the same will be removed using seam carving.





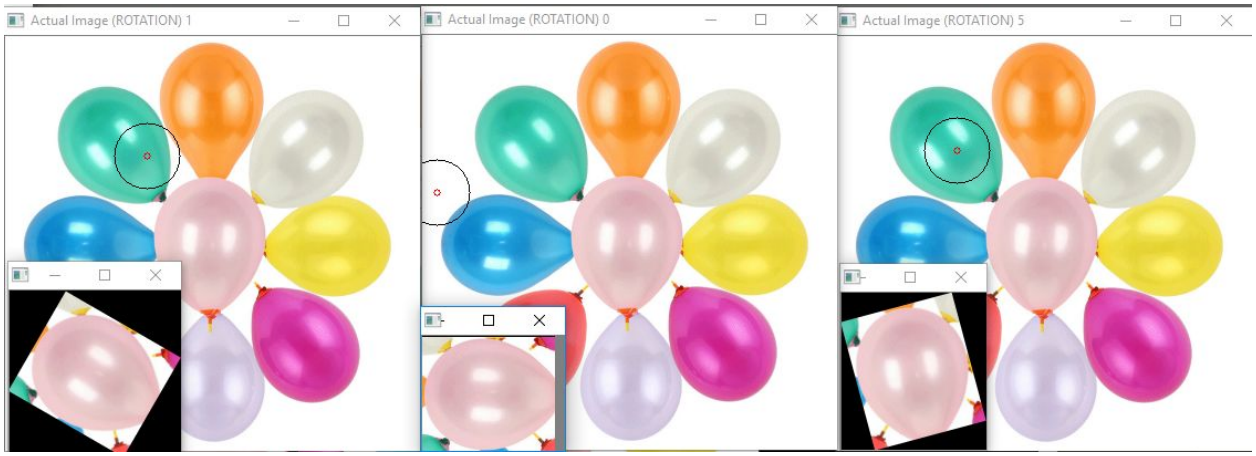
Template Matching in Fourier Domain

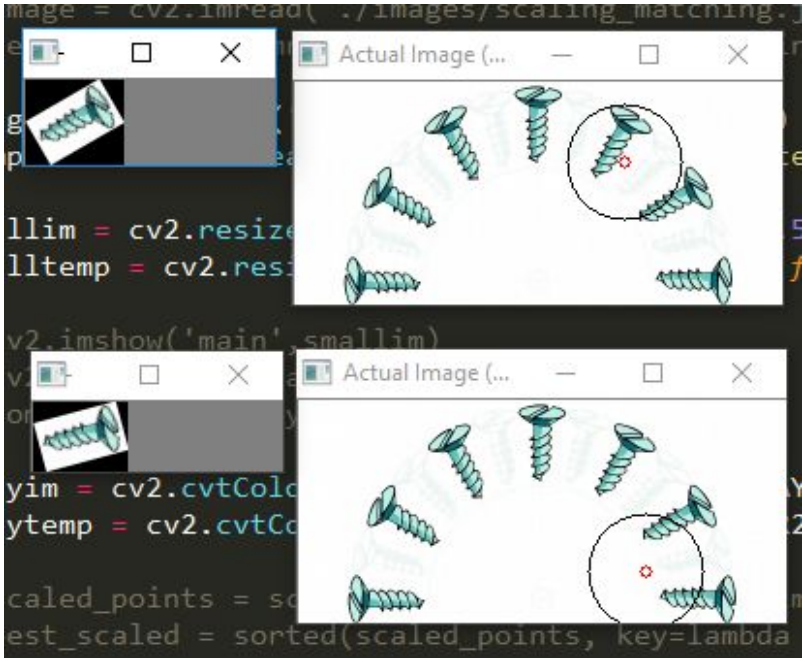
Scaling the template (0.5x to 2x) and then matching each resized template with the original image and showing the best match**.**



Rotating the template (-90 degrees from the vertical to 90 degrees right) and then matching each rotated template with the original image and showing the best match**.**

Here the black circle shows the best possible match for the rotated template in the image shown.




Actual Image (...
Actual Image (...
llim = cv2.resize
lltemp = cv2.resi
v2.imshow('main', smallim)
yim = cv2.cvtColo
ytemp = cv2.cvtCo
caled_points = sc
est_scaled = sorted(scaled_points, key=lambda

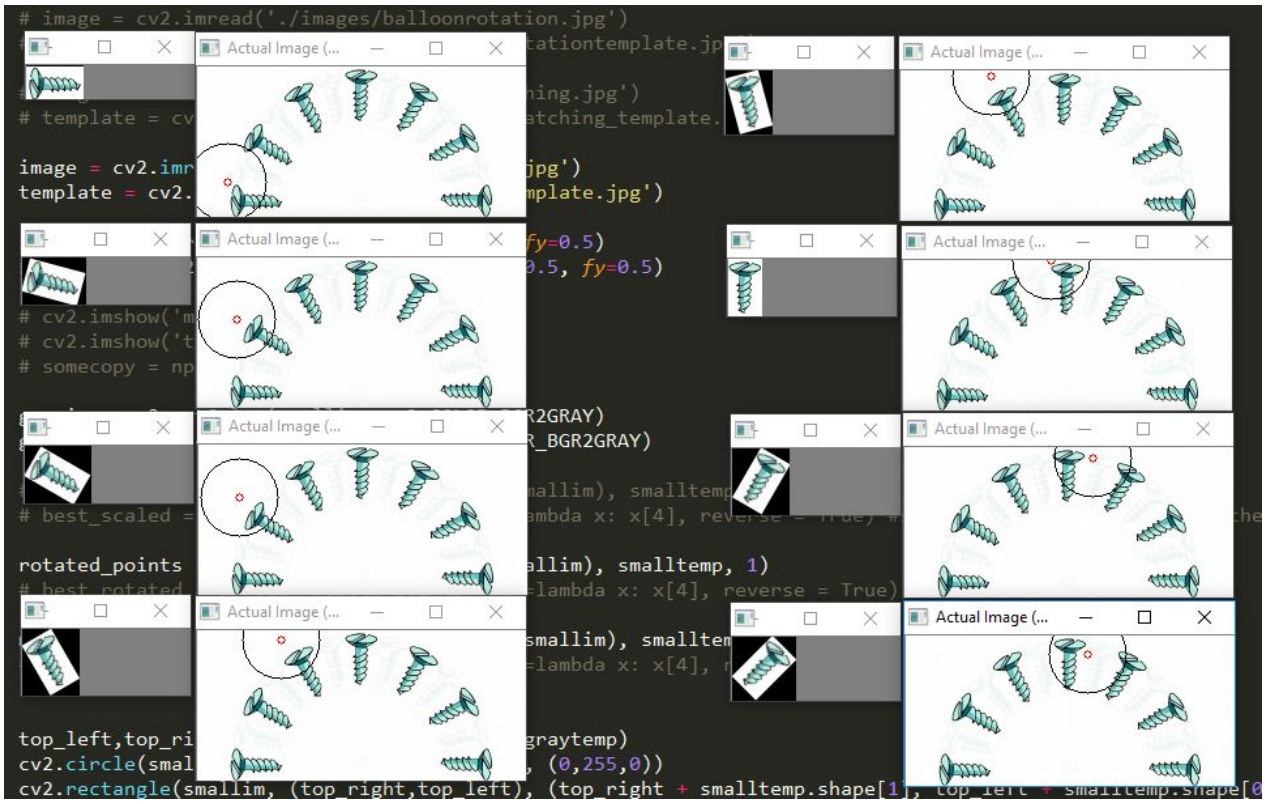
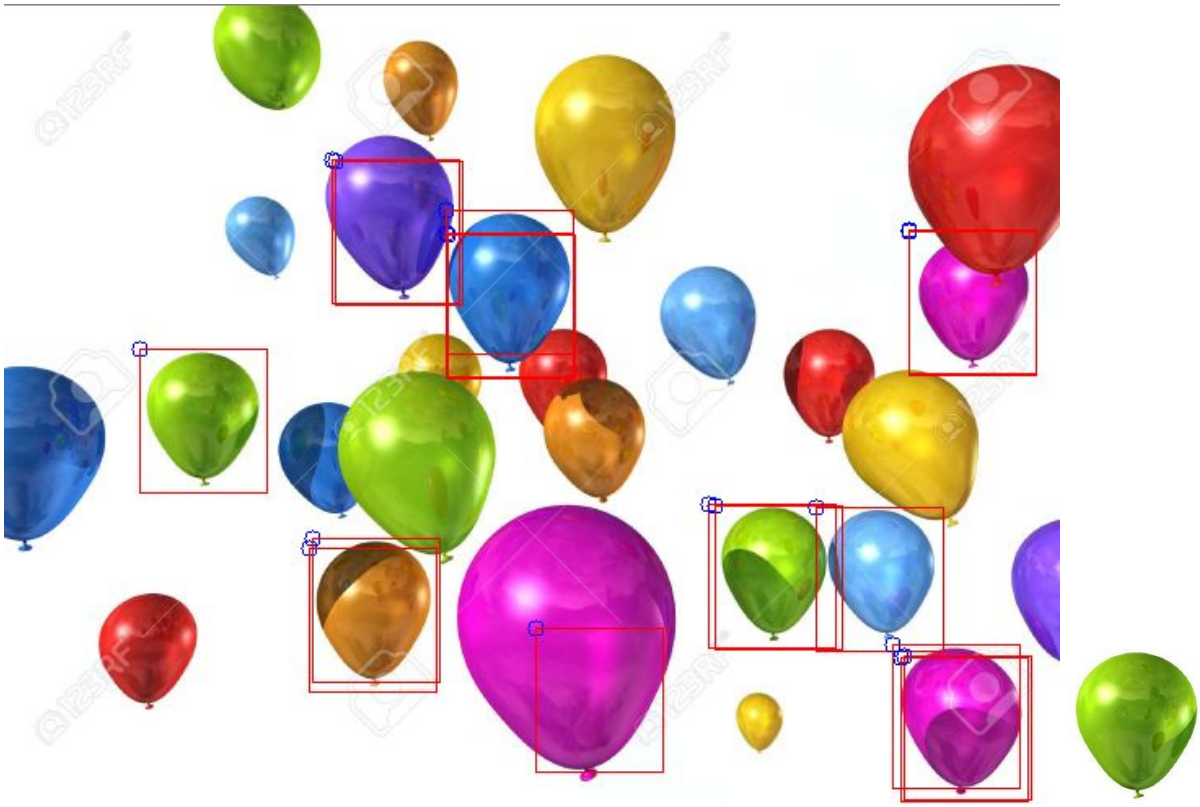


image = cv2.imread('./images/balloonrotation.jpg')
Actual Image (...
# template = cv
image = cv2.imr
template = cv2.
jpg')
mplate.jpg')
fy=0.5)
0.5, fy=0.5)
cv2.imshow('m
cv2.imshow('t
somecopy = np
R2GRAY)
R_BGR2GRAY)
best_scaled =
rotated_points
allim), smalltemp, 1)
lambda x: x[4], reverse = True)
smallim), smalltem
lambda x: x[4],
top_left,top_ri
graytemp)
cv2.circle(smal
(0,255,0))
cv2.rectangle(smallim, (top_right,top_left), (top_right + smalltemp.shape[1],

Finding some best possible matches of the template (NO ROTATION OR SCALING)



Template on the right hand side. Red squares highlight the best matches.

Exact matching