Assignment 2

22 February 2020

Seam Carving (5 points)

The goal of this assignment is to implement from *scatch* the **Seam Carving** method.

Resources:

- Paper on seam carving Seam Carving paper.
- Tutorial Brown university
- Tutorial 2 Andrew blog

1. **Compute the magnitude of image gradients.**
   We will not only compute the gradients but also the energy function in each pixel of the image.

```python
def energy_function(image):
    """Computes energy of the input image.

    Args:
        image: numpy array of shape (H, W, 3)

    Returns:
        out: numpy array of shape (H, W)
    """
    H, W, _ = image.shape
    out = np.zeros((H, W))

    # convert the image to grayscale
    image = color.rgb2gray(image)

    #computing the gradient
    Gx, Gy = np.gradient(image)

    #taking the L1 norme
    out = np.abs(Gx) + np.abs(Gy)

    return out
```

Figure 1: Initial Image

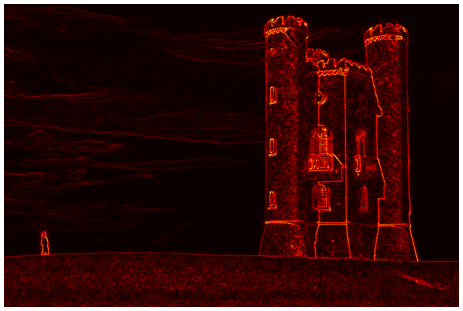


Figure 2: Energy

2. **Find the path with the smallest energy**

   - **Compute the cost on each point**: Once we computed the energy, We will compute the cost for each pixel. The procedure starts from the first line and uses *dynamic programming* to compute the minimal cost from the parents. the code is in the function `compute_costs(I, energy)` in the `seamcarving.py` file. The function also returns a `Path` which indicate the parent for each pixel in order to compute the seam.

     In the Figure 3, we show the vertical and horizontals seams. The figure clearly show that costs are higher near the tower.

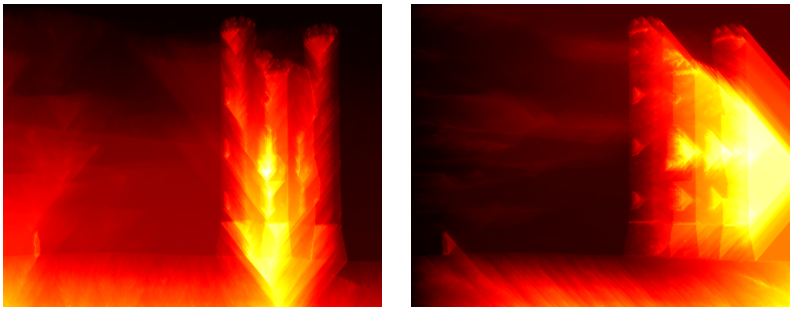
   - **Finding Optimal Seams:** Using the cost we computed in previous item, We could compute the seam with the lowest energy in the Image. Then, we could remove seam by seam until we attain the desired **width** or **height**

     – First, the function `backtrack_seam(paths, end)` where `end` is the pixel with the lowest cost. Either in the last row or

column depending the reducing mode. The function simply track the parents to find the seam.

- Second, we apply the function `remove_seam(Img, seam)` to remove the seam from the Image.
- This function `remove_seam` is repeated until get get the desired Size.

In the Figure 5, we show the resized Image after seam remove in both directions.



(a) Vertical (b) Horizontal

Figure 3: Vertical and Horizontal cost functions

Here is the figure showing the backtracked seam. The seam is traced with the red color.

Figure 4: Vertical Minimal Seam

(a) initiale (b) reduced

Figure 5: Reduced Image

KeyPoint Detection

1. **Implement a function to perform Harris corner detection?**

```python
def harris_corners(img, window_size=3, k=0.04):
    """
    Compute Harris corner response map. Follow the math equation
    R=Det(M)-k(Trace(M)^2).

    Args:
        img: Grayscale image of shape (H, W)
        window_size: size of the window function
        k: sensitivity parameter

    Returns:
        response: Harris response image of shape (H, W)
    """

    H, W = img.shape
    window = np.ones((window_size, window_size))

    response = np.zeros((H, W))

    dx = filters.sobel_v(img)
    dy = filters.sobel_h(img)

    #convolving
    dxc = convolve(dx**2, window)
    dyc = convolve(dy**2, window)
    dxcc= convolve(dx*dy, window)
    #creating the main matrix
    M= np.zeros((2*H, 2*W))

    #storing the main matrix
    M[::2,::2] = dxc
    M[1::2,1::2] = dyc
    M[::2,1::2] = dxcc
    M[1::2,::2] = dxcc

    M= view_as_blocks(M, block_shape=(2,2))

    for i in range(H):
        for j in range(W):
            mat = M[i,j]
            lam,eigenVec =np.linalg.eig(mat)
            response[i,j] = lam.prod()- k*(lam.sum()**2)

    return response
```

2. **Implement Lowe's Scale invariant interest point detection. Let the number of scales per octave be a parameter of your code?**