R06946003 湯忠憲

Homework 1 Image Matching (Detecting Motion Vectors)

Description

- Given two images: trucka.bmp, truckb.bmp
- Detect motions vectors between trucka.bmp and truckb.bmp.
- Use trucka.bmp as the basis, sample it by an 8×8, 11×11, 15×15, 21×21, 31×31 block.

Algorithm:

- Use Block Matching Algorithm
- Sum of Absolute Difference is used as the cost function.
- Segment the image into blocks with a given size, e.g., 8x8. Redundant pixels are ignored. The location of top-left point for each block is recorded.

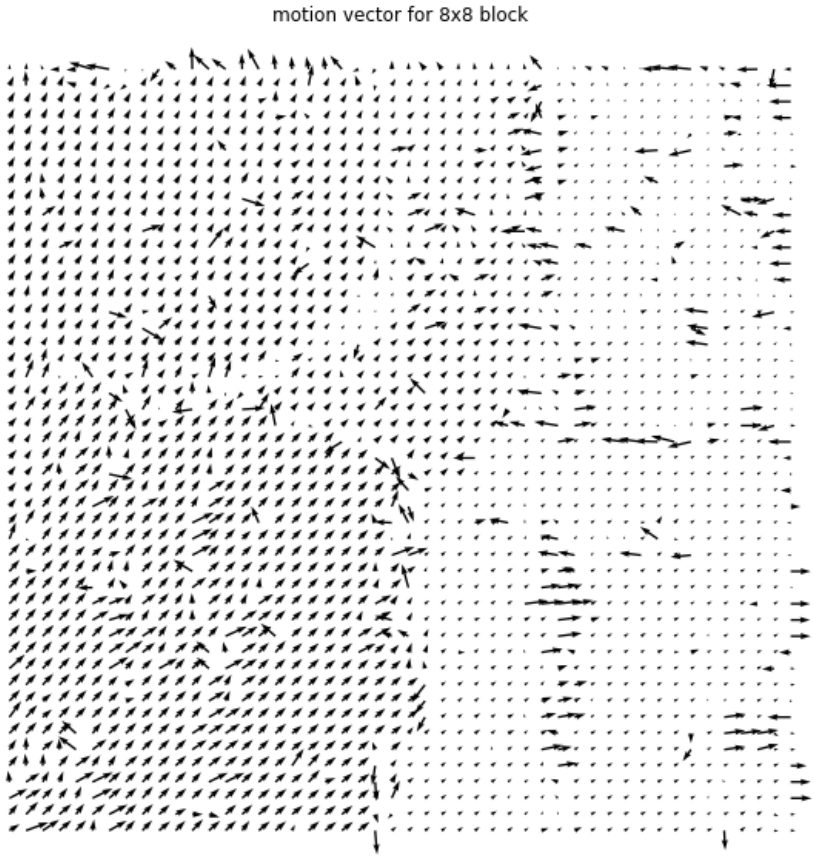
```
def get_block_position(img, windowsize_r=8, windowsize_c=8, stride=8):
    patch_list = []
    position_tup = []
    # Crop out the window
    for r in range(0,img.shape[0] - windowsize_r, stride):
        for c in range(0,img.shape[1] - windowsize_c, stride):
            position_tup.append((r,c))
            window = img[r:r+windowsize_r,c:c+windowsize_c]
            patch_list.append(window)
    return patch_list, position_tup
```

- Define a search range (50 pixels here) and find the block in basis image that can minimize the cost function.

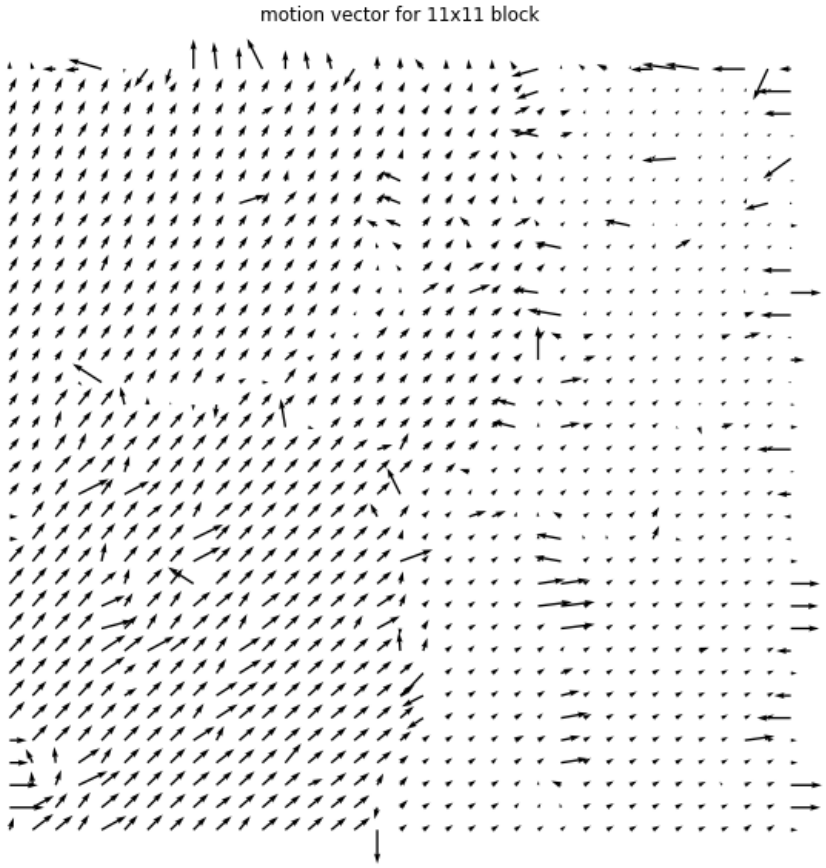
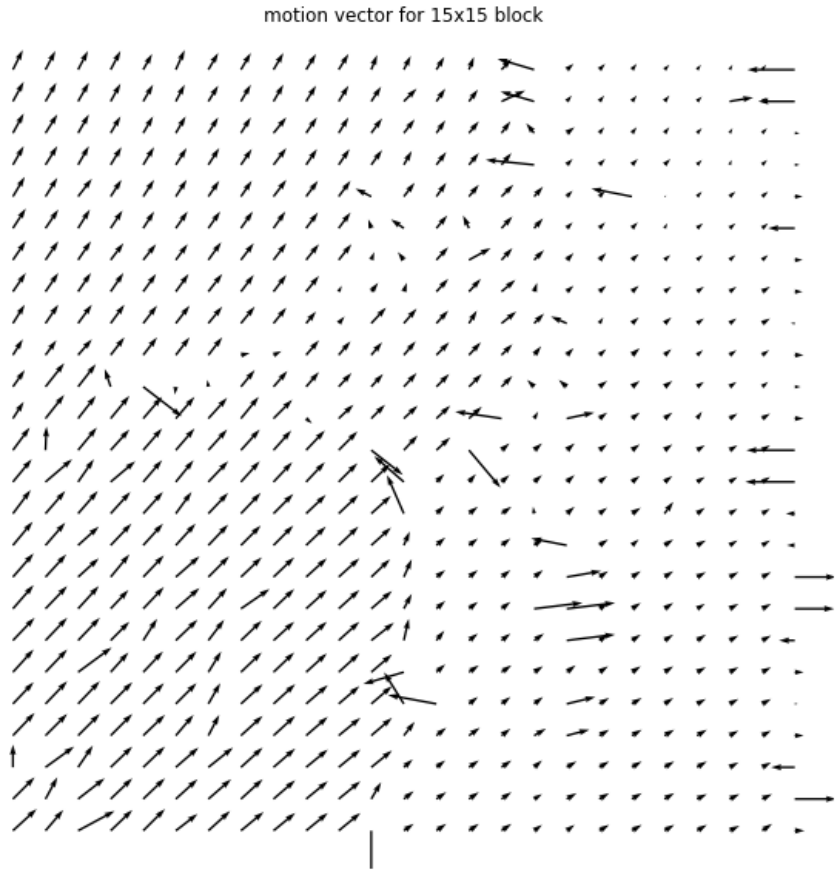
```
def get_motion_and_position(patch_list_a, position_tup_a, patch_list_b, position_tup_b, search_range=50):
    motion_tup = []
    for patch_b, position_b in zip(patch_list_b, position_tup_b):
        cost = 999999
        for patch_a, position_a in zip(patch_list_a, position_tup_a):
            position_distance = ((position_a[0]-position_b[0])**2 + (position_a[1]-position_b[1])**2)**0.5
            if position_distance<=search_range:
                difference = np.sum(abs(patch_a-patch_b))
                if difference <= cost:
                    cost = difference
                    match_position = position_a
        dx, dy = match_position[0]-position_b[0], match_position[1] - position_b[1]
        motion_tup.append((position_b, match_position, (dx, dy)))
    return motion_tup
```

R06946003 湯忠憲

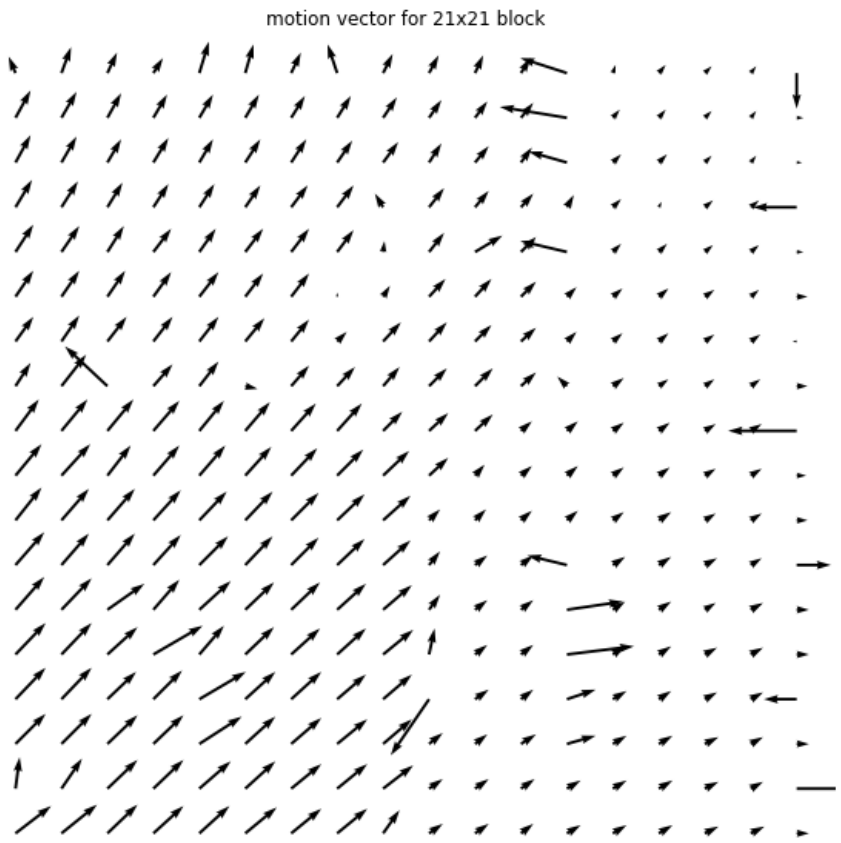
Result:

Block size	Motion vector
8x8	

R06946003 湯忠憲

11x11	
15x15	

R06946003 湯忠憲

21x21	
31x31