Problem Set 6: Optic Flow

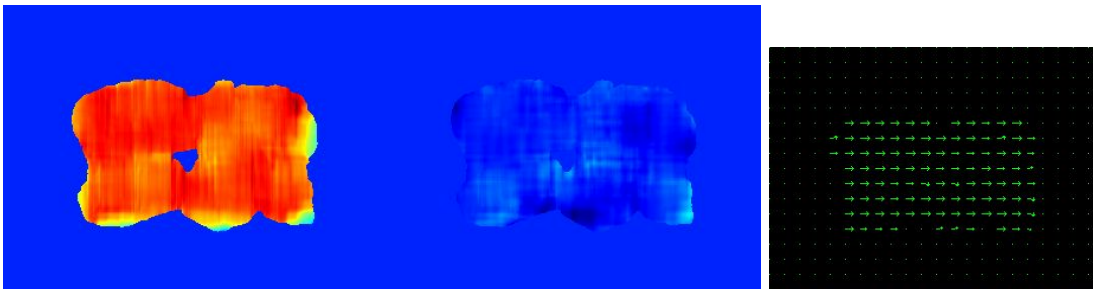
Questions

1. Lucas Kanade Optic Flow

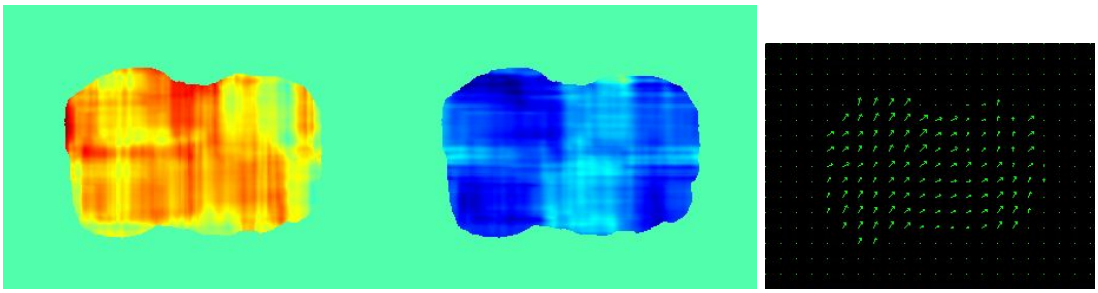
   a. Write a function `optic_flow_LK()` to do the optic flow estimation.

   Function: `optic_flow_LK(A, B) -> U, V`

   Output: Show U and V (the X and Y displacements) either as side-by-side false-color image, or as a quiver plot, when computing motion between:
   - the base `Shift0` and `ShiftR2` as `ps6-1-a-1.png:`



   - the base `Shift0` and `ShiftR5U5` as `ps6-1-a-2.png`

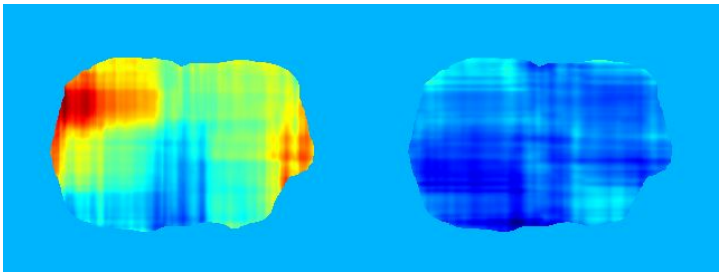


   Output (textual response):
   - If you blur (smooth) the images say how much you did

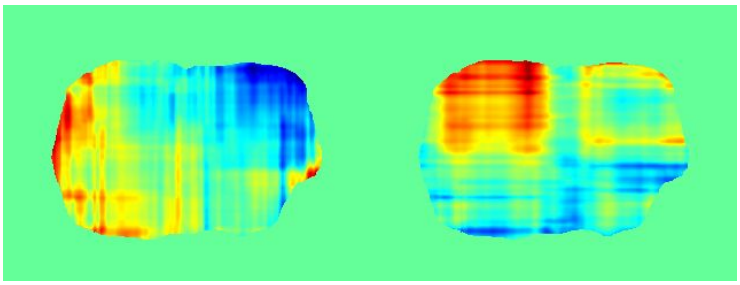
   For these, I used a gaussian blur of sigma = 5, and a summation window of 11 and 61 respectively

   b. Now try the code comparing the base image `Shift0` with the remaining images of `ShiftR10`, `ShiftR20` and `ShiftR40`. Use the same amount of blurring as you did in the previous section. Does it still work? Does it fall apart on any of the pairs?

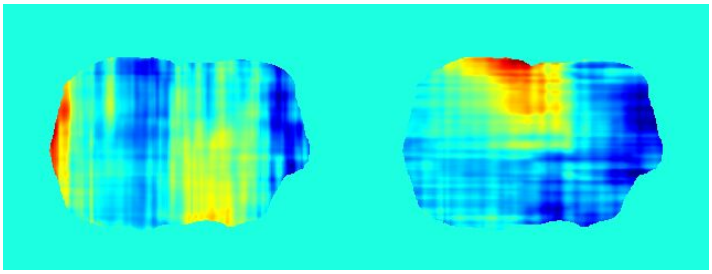
   Output: Show U and V (the X and Y displacements) either as side-by-side false-color image, or as a quiver plot, when computing motion between:
   - the `base Shift0` and `ShiftR10` as `ps6-1-b-1.png`



- the `base Shift0` and `ShiftR20` as `ps6-1-b-2.png`



- the `base Shift0` and `ShiftR40` as `ps6-1-b-3.png`



Output (textual response):

- Describe your results.

R10 came out pretty solidly. This required a decent amount of blurring to get to that point, and the summation window as pretty large (91!). After that, 40 and 60 were pretty much falling apart. I tried a number of different parameters across the board, but none of them lined up nicely to the right (got a bunch of noise included in all of them). I tried blurring more and more, and tried manipulating the window size. But nothing seemed to help very much.

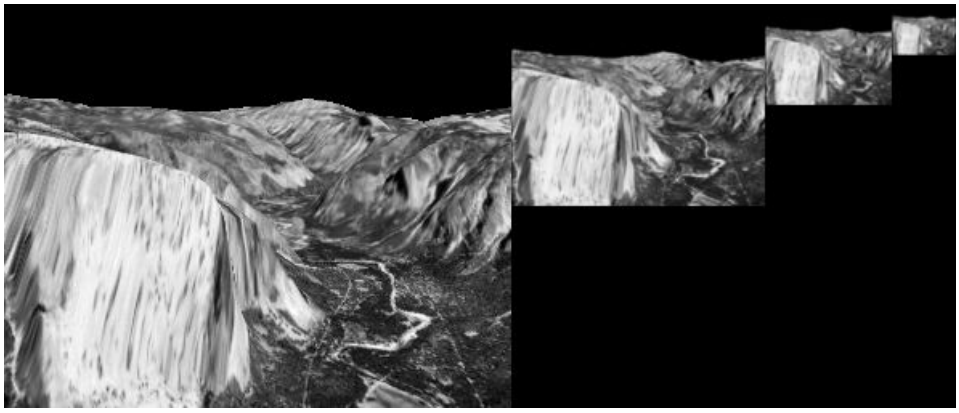
2. Gaussian and Laplacian Pyramids

Recall how a Gaussian pyramid is constructed using the REDUCE operator. Here is the original paper that defines the REDUCE and EXPAND operators:
Burt, P. J., and Adelson, E. H. (1983). The Laplacian Pyramid as a Compact Image Code

a. Write a function to implement REDUCE, and one that uses it to create a Gaussian pyramid. Use this to produce a pyramid of 4 levels (0-3), applying it to the first frame of `DataSeq1` sequence.

Functions:
- `reduce(image) -> reduced_image`
- `gaussian_pyramid(image, levels) -> g_pyr`

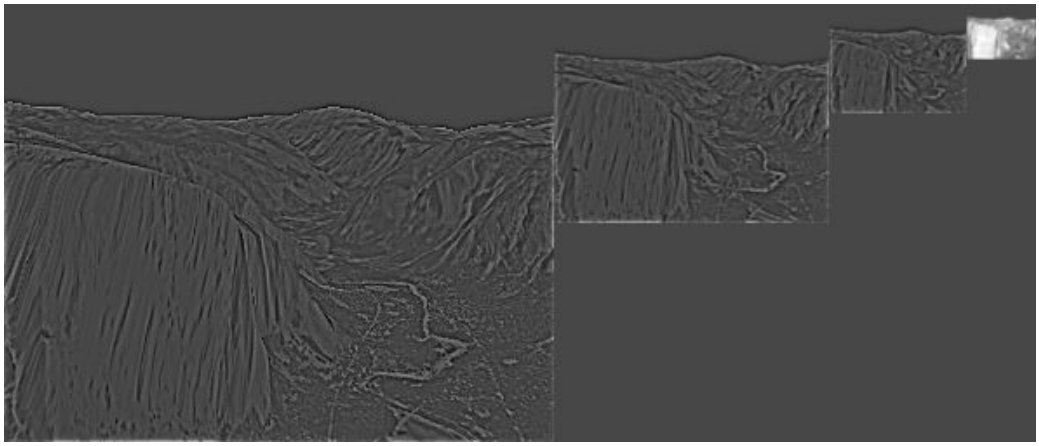
Output:
- the 4 images that make up the Gaussian pyramid, side-by-side, large to small as `ps6-2-a-1.png`; the combined image should look like:



b.

Functions:
- `expand(image) -> expanded_image`
- `laplacian_pyramid(g_pyr) -> l_pyr`

Output:
- the Laplacian pyramid images, side-by-side, large to small (*3 Laplacian images and 1 Gaussian image*), created from the first image of `DataSeq1` as `ps6-2-b-1.png`

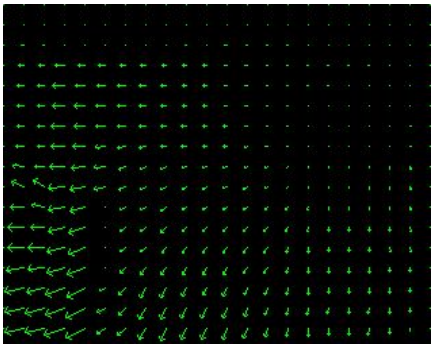


3. Warping by flow

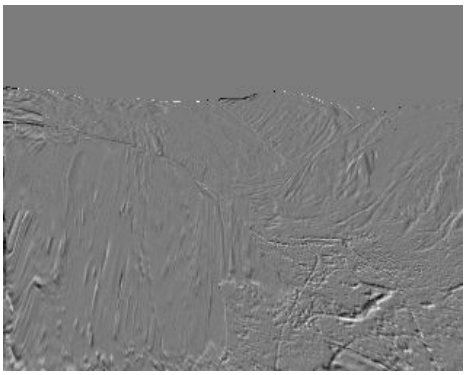
a.

Output:

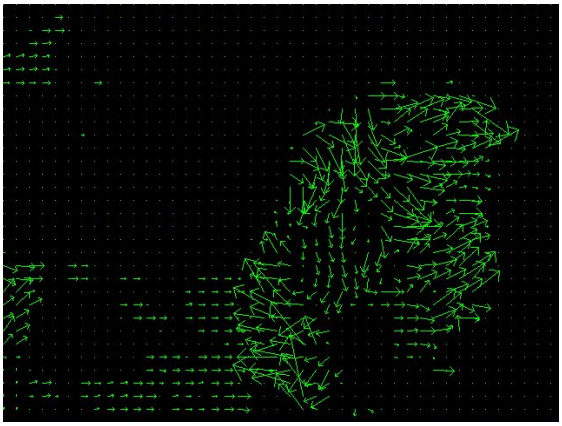
- the images showing the x and y displacements for `DataSeq1` *(either as images or as arrows)* `as ps6-3-a-1.png:`



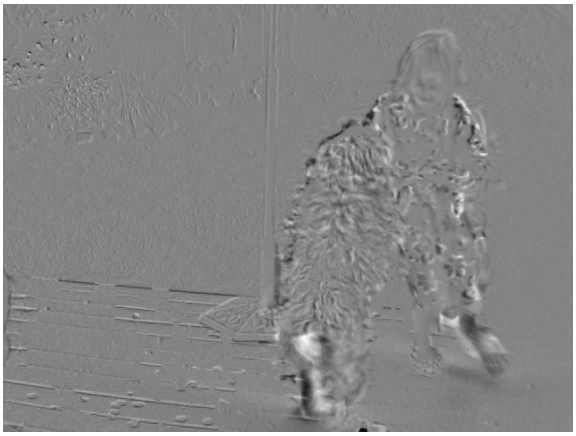
- show the *difference image* between warped image 2 and original image 1 for DataSeq1 `as ps6-3-a-2.png` (zero difference should map to neutral gray, max -ve to black, max +ve to white):



- the images showing the x and y displacements for `DataSeq2` *(either as images or as arrows)* `as ps6-3-a-3.png`



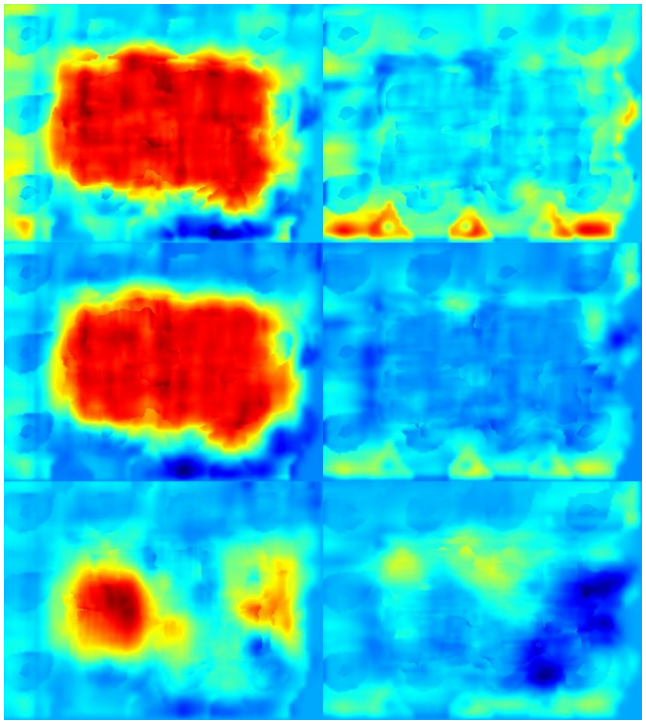
- show the *difference image* between warped image 2 and original image 1 for DataSeq2 `as ps6-3-a-4.png` (zero difference should map to neutral gray, max -ve to black, max +ve to white):



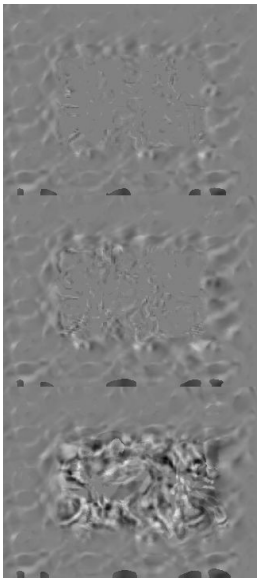
4. Hierarchical LK optic flow

Function: `hierarchical_LK(A, B) -> U, V`

a. Write the function `hierarchical_LK()` to compute the hierarchical LK optic flow.
Output:
- the displacement images between *B* and the original *A* for each of the cases; create a stacked side-by-side false-color image showing these results together as `ps6-4-a-1.png:`



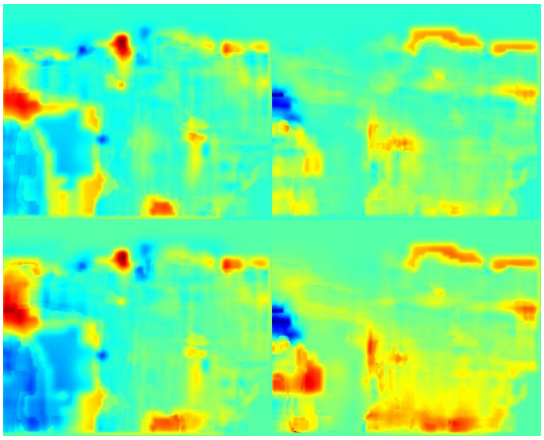
- the difference images between the warped *B* and the original *A* for each of the cases; create a stacked image showing these results together as `ps6-4-a-2.png:`



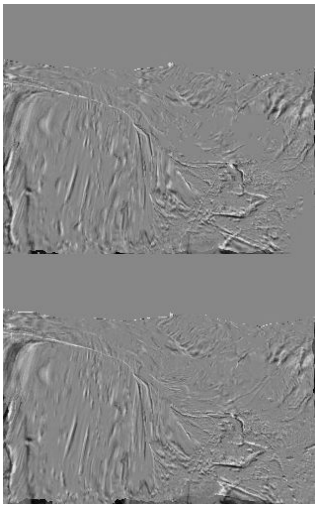
b. Apply your function to `DataSeq1` for the images `yos_img_01.jpg, yos_img_02.jpg` and `yos_img_03.jpg` .

Output:

- the displacement images between *B* and the original *A* for each of the cases; create a stacked side-by-side false-color image showing these results together as `ps6-4-b-1.png:`



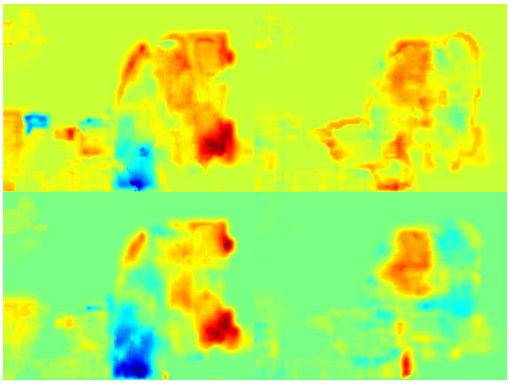
- the difference images between the warped *B* and the original *A* for each of the cases; create a stacked image showing these results together as `ps6-4-b-2.png:`



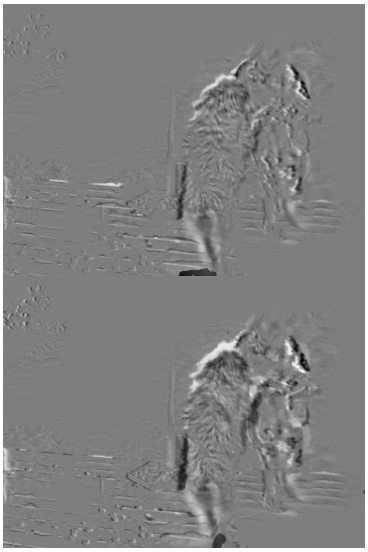
c. Apply your function to `DataSeq2` for the images `0.png, 1.png and 2.png`.

Output:

- the displacement images between *B* and the original *A* for each of the cases; create a stacked side-by-side false-color image showing these results together as `ps6-4-c-1.png:`



- the difference images between the warped *B* and the original *A* for each of the cases; create a stacked image showing these results together as `ps6-4-c-2.png:`

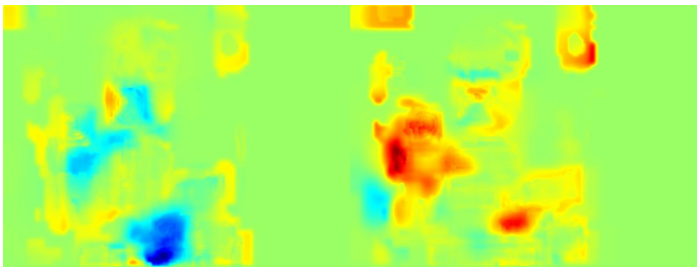


5. The sequence `Juggle` has significant displacement between frames — the juggled balls move significantly. Try your hierarchical LK on that sequence and see if you can warp frame 2 back to frame 1.

    a. Apply your hierarchical LK to the `Juggle` sequence.

    Output:

    - the displacement images between *B* and the original *A* for each of the cases
    create a side-by-side false-color image showing these results in one image as `ps6-5-a-1.png:`



    - the difference images between the warped *B* and the original *A* for each of the cases
    create a side-by-side false-color image showing these results in one image as `ps6-5-a-2.png`