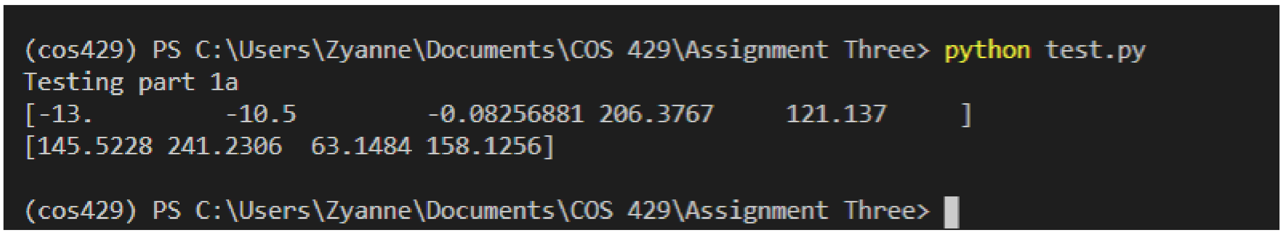
Cos 429: Assignment 3

Matthew Hetrick and Zyanne Clay-Hubbard

November 21, 2019

1 Preliminaries

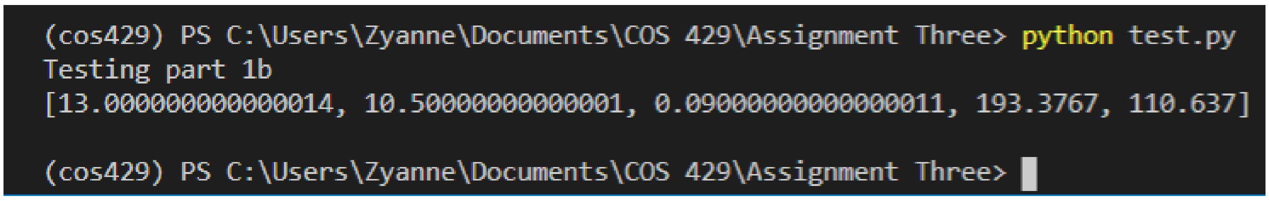
A Motion



```
(cos429) PS C:\Users\Zyanne\Documents\COS 429\Assignment Three> python test.py
Testing part 1a
[-13.          -10.5          -0.08256881 206.3767     121.137     ]
[145.5228 241.2306  63.1484 158.1256]

(cos429) PS C:\Users\Zyanne\Documents\COS 429\Assignment Three>
```

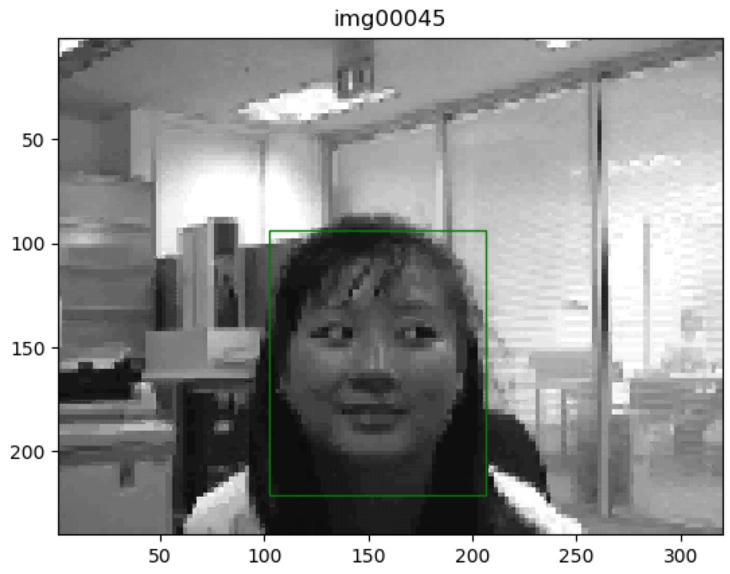
B Rectangles



```
(cos429) PS C:\Users\Zyanne\Documents\COS 429\Assignment Three> python test.py
Testing part 1b
[13.000000000000014, 10.50000000000001, 0.09000000000000011, 193.3767, 110.637]

(cos429) PS C:\Users\Zyanne\Documents\COS 429\Assignment Three>
```

C Image Sequence



2 LK at a Single Level

A (u,v) Motion

The initial error in the first iteration was 0.054466 and the final error after 11 iterations is 0.010983, so the error decreased by 0.010983.

B (u,v,s) Motion

Figure 1: With 'do_scale' = false:

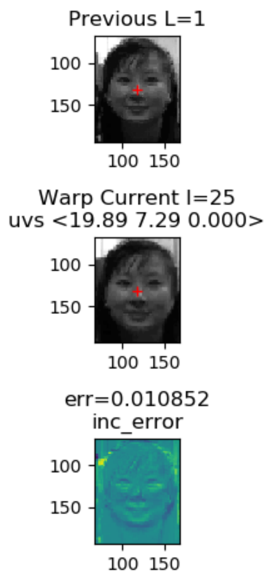
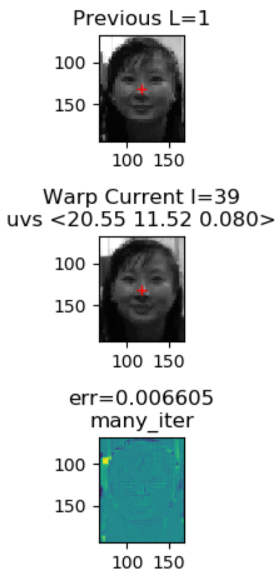


Figure 2: With 'do_scale' = true:



1) How far from the correct motion (in u, v, and s) can a single level of LK typically recover?

Generally, LK recovers much better from u,v changes than from scaling values. Even if increasing u or v all the way to 100, for example, it would move in generally the right direction, and likely would have been able to find the correct answer if given more iterations. However if the scale was changed to anything greater than ≈ 1, LK would time out very quickly and nowhere near the center of the face.

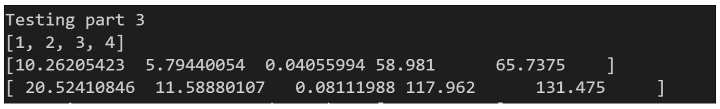
2) How consistent is the result (when it can converge) given different starting motions? Can you make it more consistent by changing the exit conditions - the parameters: max_iter, uvs_min_significant, and err_change_thr. Explain what these do and whether/how changing them changes the accuracy.

The results are fairly consistent when given different values of u and v. Different values of s make things more inconsistent. [max_iters] is the number of times the loop runs, calculating the motion from one frame to the next. Increasing [max_iters] gives the algorithm more time to find the correct solution. A greater [max_iters] would provide more accurate results. [uvs_min_significant] is the minimum amount that the motion has to change from one iteration to the next in order to be considered significant. If [uvs_min_significant] decreases, than the algorithm is forced to keep going until the motion vectors are similar enough from one iteration to the next. This would create more accurate results. [err_change_thr] denotes the minimum change in error acceptable. If the change in error is greater than this threshold, that means that the error has increased from one iteration to the next, and if the error is increasing, the loop should exit. Decreasing [err_change_thr] will make the loop exit sooner, but would leave a narrower error margin, and make the results more accurate that way.

3 Multi-Level LK

A Implementing defineActiveLevels and uvsChangeLevel:

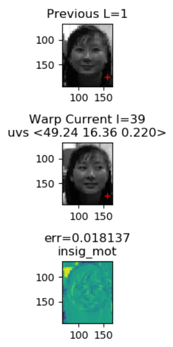
Figure 3: Output for defineActiveLevels and uvsChangeLevel:



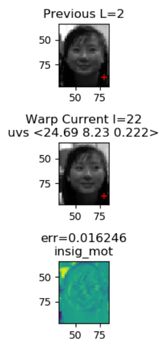
```
Testing part 3
[1, 2, 3, 4]
[10.26205423  5.79440054  0.04055994 58.981      65.7375    ]
[ 20.52410846  11.58880107   0.08111988 117.962      131.475     ]
```

The first line in the clip above shows the output of defineActiveLevels, and it shows that the available levels are 1-4. The line below that is the output of uvsChangeLevel for the provided motion in the testing file when we go from level 1-2. This is smaller than the initial motion, which makes sense since we are going up a level. The third line in this clip is the uvsChangeMotion output from level 2-1. This is larger than the motion from level two, which makes sense since we are going down a level, which takes us back to our original motion!

B Figures generated from LKonPyramid:

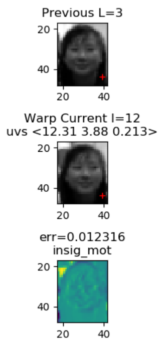


(a) Level 1:

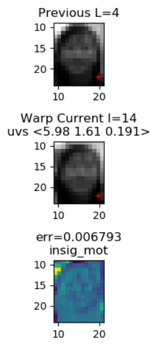


(b) Level 2:

Figure 4: Levels 1 and 2



(a) Level 3:



(b) Level 4:

Figure 5: Levels 3 and 4

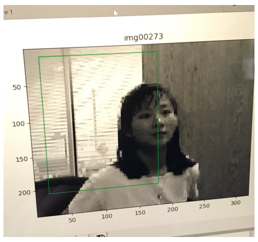
C Initial motion question:

1) How far can init_motion be from the true motion? How does this compare with your results from part 2b on a single level? Describe your findings in your writeup.

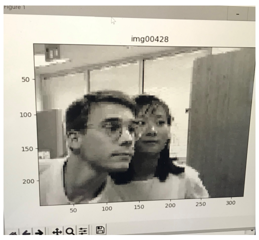
As in part 2b, u can be quite far from the true motion and LK recovers relatively well, because even with increases up to 100 it still converges to a reasonable marker. Moreover, just like in 2b, LK does not recover well with increase in scaling, though it performs slightly better than the single level, for it has relatively good performance (converges to a reasonable marker estimate) up until around 4.
However, the main difference between this implementation and that of 2b is in how well LK recovers if v changes. In 2b, one could change v up to around 100 with seemingly good recovery with more iterations. Now, if v is increased to much more than 25, it ceases to move in the right direction and converges on around 2 iterations. At this point, the markers are not close to each other and there is an error of $\approx 5-9\%$, specifically for $v=35$.

4 Incremental Tracking for a Sequence:

On the girl sequence, tracking becomes inaccurate after the first spin of the subject around frame 114, and becomes incredibly inaccurate completely during the second spin, around frame 224, and fails completely around frame 428 when another face is introduced. Therefore, the longest sequence before it legitimately fails is around 428, or specifically looking at the motion output, it decreases significantly around frame 428 and becomes zero at frame 445. If we define failure as terrible inaccuracy, this begins around frame 224. Attached below are frames 273 and 428.



(a) Gross inaccuracy for frame 273:



(b) Failure at frame 445 with no rect:

Figure 6: Frames 273 and 445

We believe this failure happens for a few reasons:

- The first instance of inaccuracy could be due to girl spinning around and the motion of her face becomes harder to track.
- The second instance increases this error, especially when the girl feigns left and ends up spinning to the right. This becomes inaccurate because the model is predicting the future movement based on the previous movement, so in this situation our tracker predicts a leftward movement that does not actually happen. This is an example of drift, when more errors accumulate over time. Additionally, her movement towards the end of this spin is not small, so it is not in line with our assumption of small movement.
- Lastly, the reason it entirely fails when another face is introduced is because now there is no way to predict the motion when two images are within our rectangle boundary. Moreover, with two overlapping faces in our search rectangle, the neighboring pixels on the intersection of the faces do not move together, causing the algorithm to fail.

On the david sequence, the algorithm fails around frame 394, when the subject puts back on his or her glasses. Therefore, the longest sequence before LK fails is around 394 That frame is attached below.
We believe this is due to change in brightness on the subject's face during the motion of putting the glasses back on, or as coined in lecture, we have poor brightness constancy here. It makes it such that the motion of the pixels on the right-hand side of the subject's face is not the same as on the left-hand side, resulting the inaccurate tracking.

Figure 7: David fail:

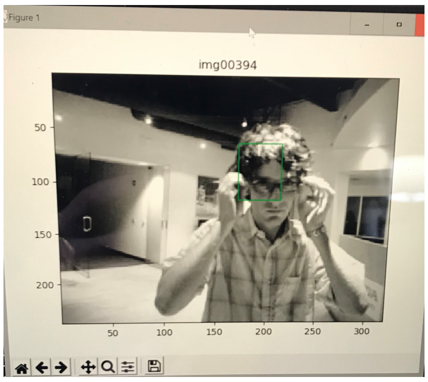


Figure 1
img00394
50
100
150
200
50
100
150
200
250
300