Jonathan Pearce

260672004

Comp 558

Assignment 3

a)

Figure 1

Basketball Frame 7: m = 51, Image smooth σ = 0.25, Gaussian weighting window σ = 10

Figure 2

Basketball Frame 11: m = 51, Image smooth σ = 0.25, Gaussian weighting window σ = 10

Figure 3

Basketball Frame 11: m = 101, Image smooth σ = 0.25, Gaussian weighting window σ = 10

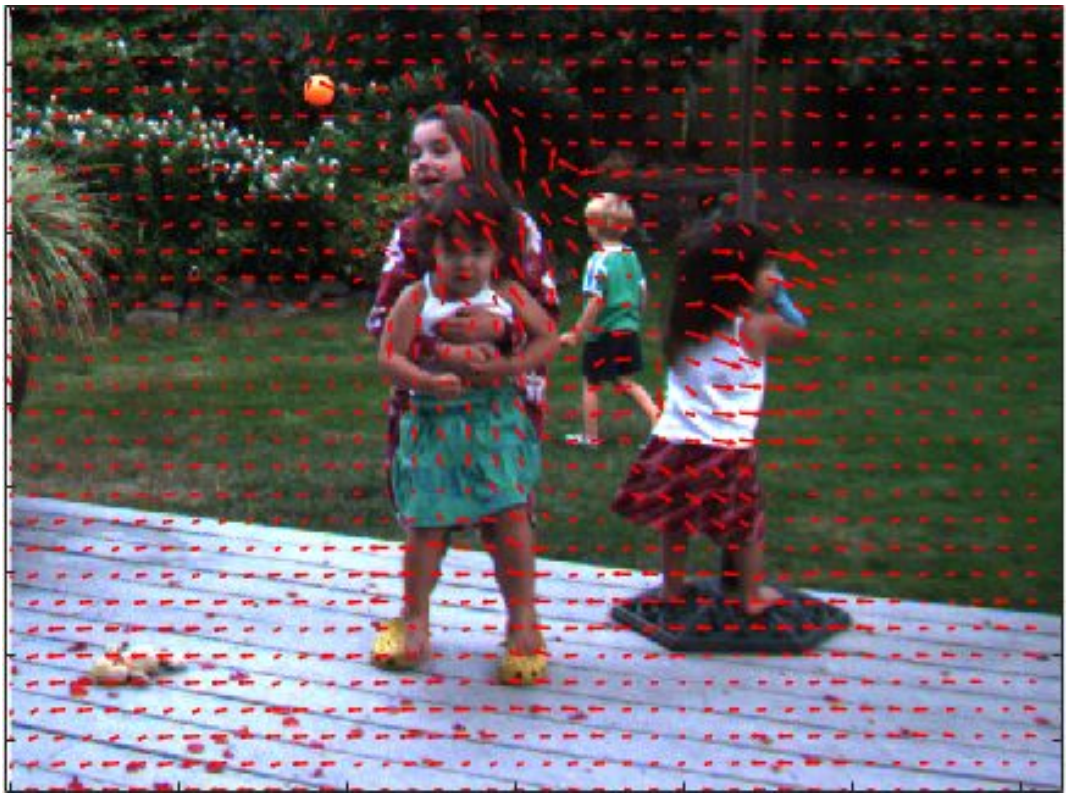

Figure 4

Backyard Frame 7: m = 31, Image smooth σ = 0.25, Gaussian weighting window σ = 10

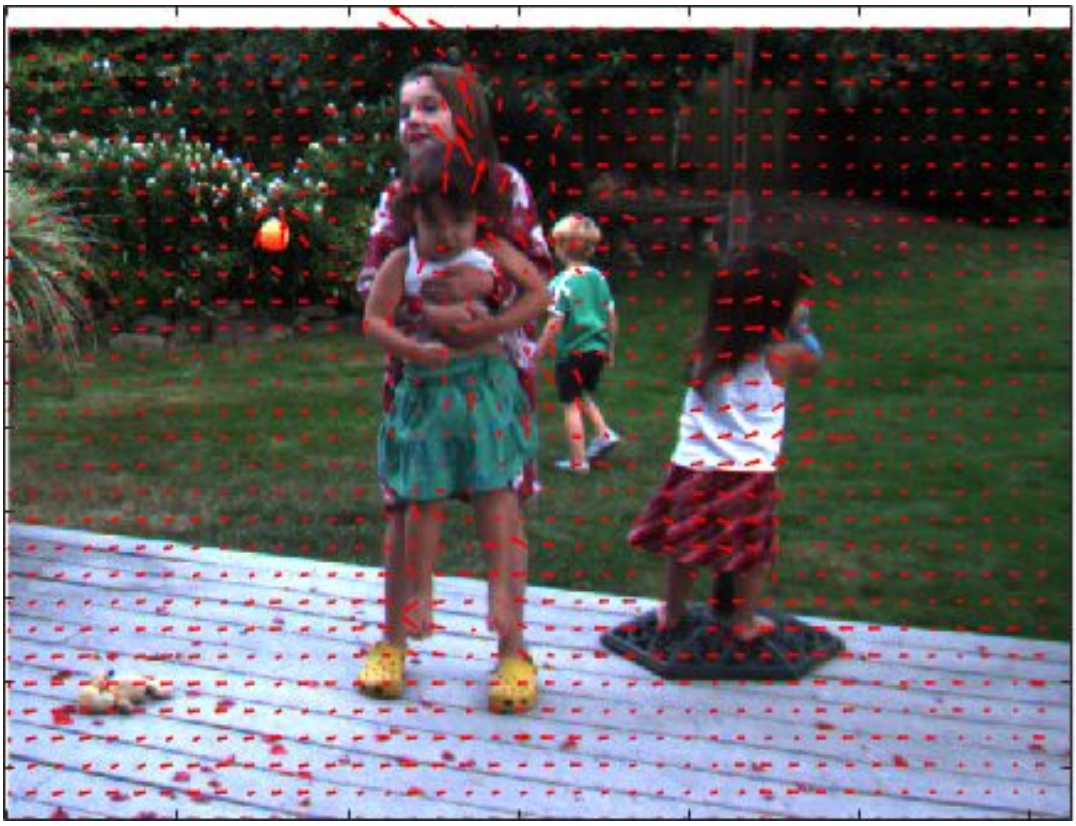

Figure 5

Backyard Frame 11: m = 31, Image smooth σ = 0.25, Gaussian weighting window σ = 10

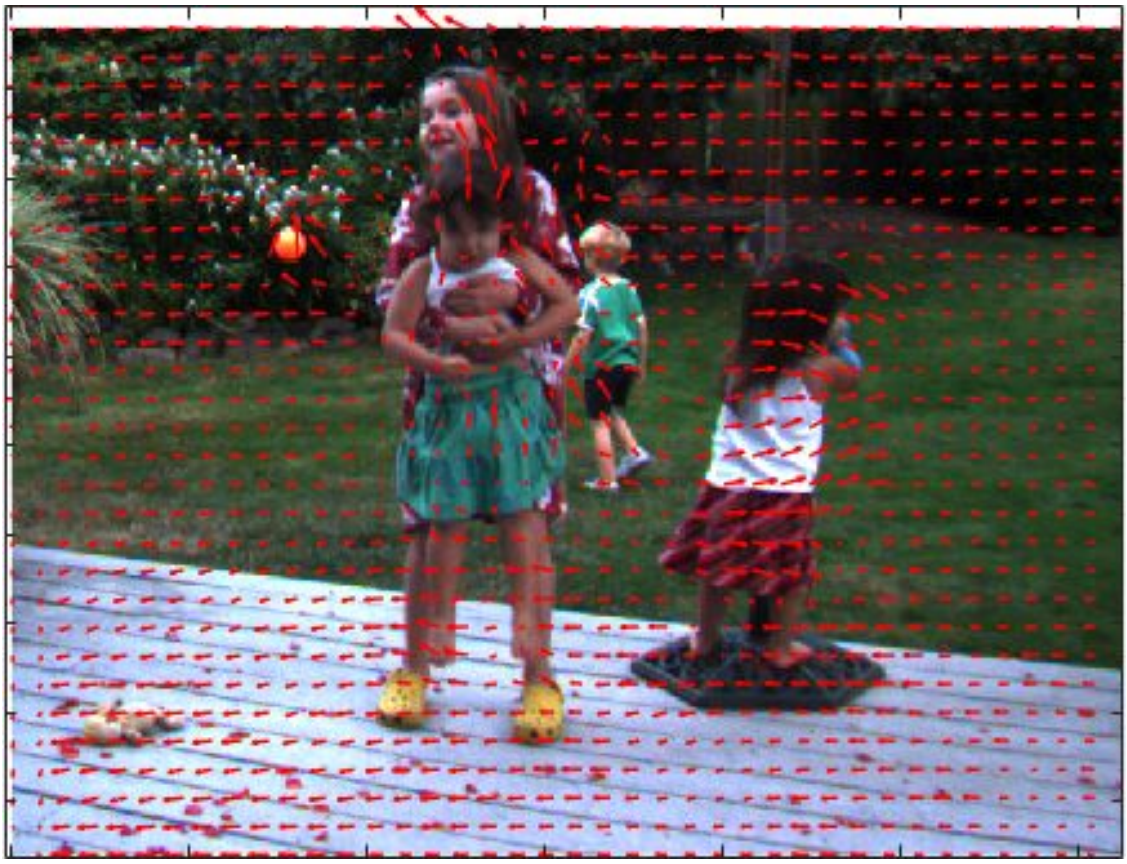

Figure 6

Backyard Frame 11: m = 71, Image smooth σ = 0.25, Gaussian weighting window σ = 10

In general the basic Lucas-Kanade algorithm does well at solving for the motion field between two image frames. In the basketball images, for Figure 1, we capture (i.e. find large motion vectors) the left person's motion upwards and to the right of the image as he begins to throw the basketball. Further, for the person on the right we capture their hands moving upwards in preparation to catch the ball, at the same time this person moves their head back as he becomes more upright in further preparation for the ball, which is also captured by the algorithm. Overall figure 1's motion field successfully captures most of the motion between frames 7 and 8. We find similar success in Figure 2 (and Figure 3) with the exception of the basketball motion not being detected accurately, In Figure 3 we increased m significantly to show that this issue is independent of window size m. Another observation from Figure 2 is we see that Lucas-Kanade is actually able to track the left person's shadow on the wall (the area on the wall with large motion vectors), this was quite interesting because even after seeing the results from the algorithm I still have trouble seeing the shadow of the person move unless I zoom in on the photo significantly. Capturing the shadow was quite impressive as it shows Lucas-Kanade is precise enough to capture differences in motion that are so subtle that humans may miss them depending on the photo conditions.

The backyard images found similar success as the basketball images. Figure 4 (and 5, and 6) shows that Lucas Kanade finds the vertical motion of the two children in the middle of the image frame, the horizontal movement of the girl on the right and even picks up bits of the blonde child's walking motion in the background. What is different about the backyard scene is that the camera was panning to the right ever so slightly and this leads to the background of the image being filled with horizontal motion vectors moving to the left as demonstrated in figures 4,5 and 6. One issue with the backyard scene is the orange ball falling is not detected in any of the motion fields. I tried to increase the window size to ensure we were capturing the ball's movement but figure 6 shows that even the larger window does not help with detecting the ball's motion.

In general the basic Lucas-Kanade does a fine job capturing most of the smaller motions between two frames. The algorithm sometimes overestimates and underestimates certain vectors magnitudes or orientations depending on their neighbourhood environment, but it is a good quick approximation if one wants to see the general motion field within two frames.

I'm not confident why the basketball and smaller orange ball's motion was not being detected properly in Figures 2 to 6. My best idea was that since those objects moved so much between frames (i.e. their velocity was greater than other objects in the frames) this means that these balls are near the edge of the Gaussian window, and since the weighting scheme of the window decreases as we get farther from the candidate pixel, the image difference at the ball's new location would receive very little weight which may have contributed to their motion going undetected. This idea explains why the basketball's motion was found in Figure 1, since it's velocity between frame 7 and 8 was less than that it's velocity between frame 11 and 12.

b)

For both the basketball and backyard scenes the motion field solution in part a from frame 7 to 8 seemed to be the most accurate, at least from a qualitative perspective. Therefore I choose to use these two example for part b to study how an iterative approach improves the results from the basic method.

Figure 7

Basketball Frame 7: m = 51, Image smooth σ = 0.25, Gaussian weighting window σ = 10

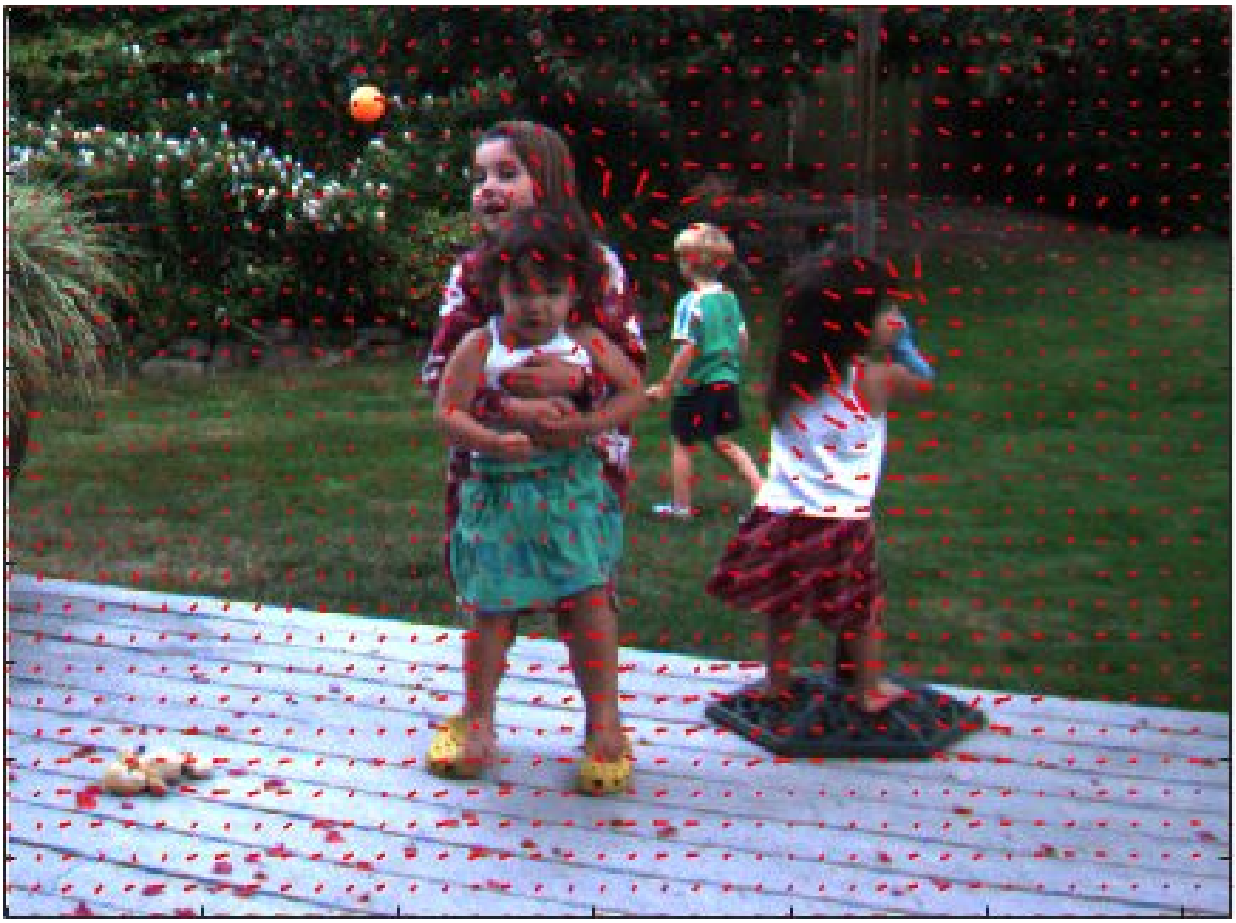

Figure 8

Backyard Frame 7: m = 31, Image smooth σ = 0.25, Gaussian weighting window σ = 10

The iterative version of Lucas-Kanade appears to have the same success as the basic version at finding and identifying general motion between two image frames. The iterative version still does not find the motion displayed by the orange ball in Figure 8, this makes sense because the window size and window weighting did not change between the iterative version and the basic version. The iterative method is able to identifying all other elements of motion in Figure 7 and 8.

What the iterative method appears to improve upon is calculating more accurate vector fields. In particular, smoothing the vectors, both in terms of direction and magnitude. In the basic version of Lucas-Kanade there are many examples of vectors beside each other in the same object that differ significantly in terms of angle or size (e.g. there might be a small horizontal motion vector and then directly to the right there is a large vertical motion vector), this difference in motion vectors that are so close to each other in the same object is difficult to explain. It appears that the iterative method solves this issue by making the change in motion field vectors more gradual in terms of direction and magnitude, as the type of motion changes in the images. This helps make the motion from two image frames easier to interpret. Comparing Figure 1 and Figure 7 as well as Figure 4 and Figure 8 there are numerous examples of this smoothing process, although it does appear to be more prevalent in the Figure 4 & 8 comparison. This smoothing process appears to solve the overestimate and underestimate issue that was discussed in part-a, and makes the direction of the field motion vectors slightly more accurate.

NOTE: My implementation had trouble converging in reasonable time. After a qualitative analysis it appeared that 5 iterations was enough for some of the scale and orientation errors from part-a to be corrected. Since my convergence was taking more than 5 minutes and was unreliable as I experimented with model parameters I decided to cap the iterative algorithm at 5 iterations (~20-30 seconds - ~5 seconds per iteration) to allow me to experiment with parameters properly in a reasonable time frame. I'm really not sure where this source of error is stemming from but for

this part of the assignment the results with 5 iterations seemed reasonable. This protocol of 5 iterations was also used in part c.

An option to try and solve the convergence problem would have been to scale down the images and then scale Vx and Vy back up to the original size at the end of the procedure, similar to what we do in part c.

c)

Figure 9

Backyard Frame 7: m = 31, Image smooth σ = 0.25, Gaussian weighting window σ = 10

Figure 10

Basketball Frame 11: m = 51, Image smooth σ = 0.25, Gaussian weighting window σ = 10

Figure 11

Schleffera Frame 10: m = 31, Image smooth σ = 0.25, Gaussian weighting window σ = 10

Sadly, judging from how Figure 9, 10 and 11 do not display the same accurate and smooth motion field vectors as shown in part-b I feel as though my implementation of this 3rd method is not correct. I believe my procedure for moving through the image pyramids and warping the initial frames using estimates from the pyramid level below is correctly done, therefore I suspect my issue is arising from the problem I talked about in question b, where the convergence of my iterative algorithm is very slow, which required me to cap the number of iterations at a fixed constant.

Regardless, the direction of motion field vectors in these 3 figures appear to be more or less correct, although there scale seems quite off especially in figure 10.

Although my results to do not demonstrate this property, I feel as though the pyramid approach to Lucas-Kanade would allow for the detection of motions with greater velocities such as the basketball in the later frames of the basketball sequence and the orange ball falling in the backyard images (i.e. the motions that were missed in part a and b). This would be possible because at the coarsest/smallest level of the pyramid these features that move a significant amount would be able to be contained within a reasonable size of window frame and remain close to the candidate pixel in the second frame therefore receiving significant weight from the Gaussian weighting window, this was the issue discussed in part-a. Once these faster motions are found and their corresponding motion field vectors are solved for, these vectors would be propagated up the pyramid and refined at each level, leaving the finest image (top level) to have motion field vectors that represent these fast movements.

Because of my issue with iterative convergence it is difficult to assess the efficiency of this pyramid approach. I would imagine that each level is faster than the standard iterative algorithm since we estimate the motion field vectors by scaling the final motion field vectors from the previous level of the pyramid and use these to warp the initial frame at that particular level. In other words at each level, when we run the iterative algorithm we already have a decent approximation.

Another possible source of error is that I did not maintain a constant ratio between the σ used to blur the image and the σ used to weight the Gaussian window. This approach is discussed here, http://www.cim.mcgill.ca/~langer/558/2009/lecture12.pdf on page 1. More generally it is possible that my parameter values were the cause of this method failure.

d)

In order for our method to capture rotations and shearing we would need to create a 2x2 matrix T and include that with V = (Vx,Vy), where T would account for these more general affine motions.

In particular, our flow around a center c would be,

$$flow = T\,(x - c) + V$$

Where x are the pixels in the neighbourhood of c

We can then define a sum of squares problem similar to that seen in lecture,

$$\sum_{x \in Ngd(c)} (I\,(x + \ T\,(x - c) + V\,) - J\,(x))^2$$

Utilizing a Taylor expansion and taking derivatives with respect to 0, we could create a system of equations. Using this system and the method from part b we could iteratively solve for these 6 values (2 – translation values, 4 affine motion values) that best explain the motion experienced by pixel c and its local neighbourhood of pixels. The iteration process would go until the 6 values converge.

Implementation notes:

Added gaussian weighting matrix W to right hand side of Lucas Kanade equation (the same we use on the left hand side) to help localize results further

Multiplied right hand side Lucas Kanade equation by -1 because my motion vectors were going the wrong way