CV Assignment 2 Report

Pratyay Suvarnapathaki, 2020111016

1.1 Thought Experiments

1.1.1 Slow Motion

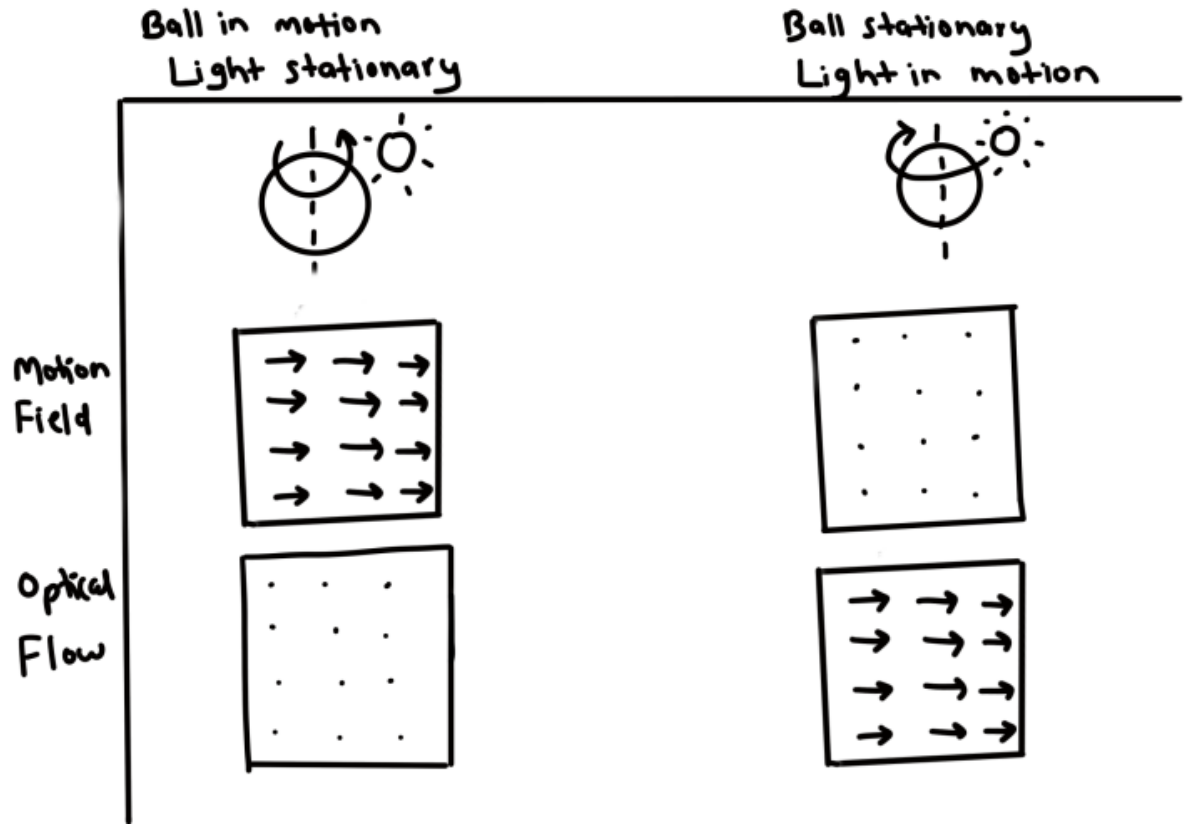
Slow motion → Same motion in a longer duration.
Were we to keep the framerate of the video constant and just ‘make it longer’ by spacing out the frames, the motion would be choppy and slow. Here, we may use dense optical flow to ‘blend in’ consecutive frames, to produce an interpolated frame between the two. The resultant video would be sufficiently long, with smooth motion too, thereby achieving a ‘slow motion’ effect.

1.1.2 The Matrix

Just like slow-mo, we could consider the ‘bullet-time’ scene as one with an added component of ‘flow-mo’, i.e. in addition to increasing the temporal resolution of footage as in slow mo, the interpolation is now also in the spatial domain.
The rooftop scene famously used a rig of 120 cameras surrounding the actor firing at very high speeds, capturing the actor’s flailing motion from all directions. Subsequently, the footage was slowed down, and a smooth rotation around the actor was obtained by using optical flow to interpolate between the images captured by adjacent cameras.

1.1.3 Lambertian Sphere

The very name ‘optical flow’ suggests that what can be captured are mere appearances, which may not necessarily reflect the ground truth of the actual motion occurring in an image. Thus, in a rotating Lambertian sphere, there is an obvious motion field, but that cannot be captured in its appearance owing to the constant state of illumination and smooth surface of the sphere. On the other hand, when the light moves, appearances change in spite of there being no motion actually taking place. Therefore, here, the optical field captures pseudo-movement in the opposite direction to the movement of the light.



1.2 Concept Review

1.2.1 Objective Function

The objective function itself doesn't make any assumptions regarding the noise distribution. Moreover, one of the assumptions of the objective function is brightness constancy.

However, in practice, pixel intensities may be affected due to some noise prevalent in the image. Herein, the objective function being used on a window of pixels can be operated on using a weighted least squares approach to estimate the motion. In this approach, the weights are determined by the noise characteristics, which can be modelled as Gaussian (with known standard deviation and zero mean). Thus, the objective function can still be used as is to reliably retrieve motion from noisy images, using a weighted least squares approach.

1.2.2 Taylor Series Approximation

The first-order Taylor series approximation is used to linearize the objective function and therefore make it more efficient to solve computationally / iteratively. This is because gradients can be calculated easily in this formulation. The mathematical formulation used here is:

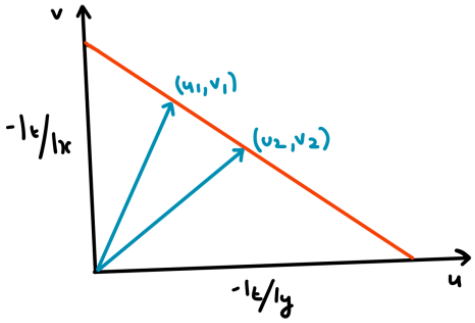
$$E(u+\delta u, v+\delta v) \approx E(u,v) + (\partial E/\partial u)\delta u + (\partial E/\partial v)\delta v$$

1.2.3 Ill-Posed Problem

Post taking the Taylor series approximation of the objective function, we arrive at the equation:

$$I_x u + U_y v + I_t = 0$$

Here, there are clearly two unknowns (u and v), but we only have this single equation. Therefore, the problem of recovering velocities in x and y for a single pixel location is an ill-posed problem. Geometrically, this can be illustrated as:



The ‘solution space’ here lies on the red line, any solution on it is equally valid. This is also termed as the ‘aperture problem’, and can intuitively be extended to explain the barberpole illusion too. To overcome this problem in practice, optical flow is typically computed for all pixels within a windowed neighbourhood of the pixel we actually want to track. This does result in an overdetermined system, but the solution obtained using the least squares method herein tends to be unique and mostly precise.

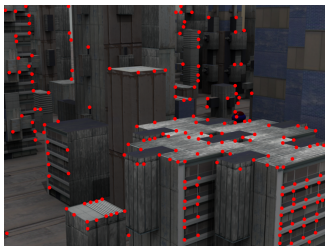
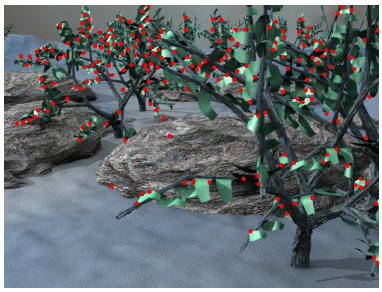
---

2. Implementation of Single-Scale Lucas-Kanade Method

> Note: All the results described/pictured herein have been stored in the apropriate subdirectory under results/.
> Red arrows = my implementation ; Blue = OpenCV implementation

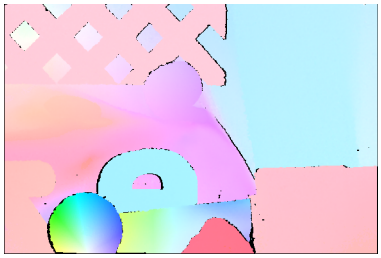
2.1 Harris Corners

A maximum of 200 Harris keypoints are chosen using `cv2.goodFeaturesToTrack()`

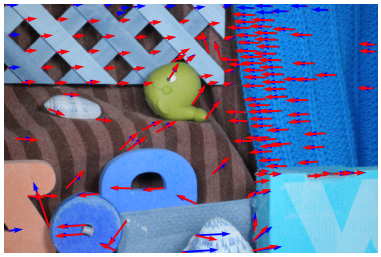
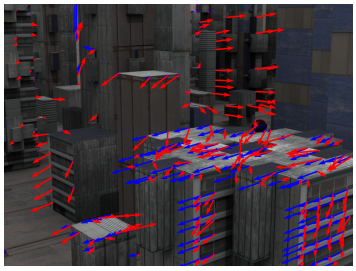
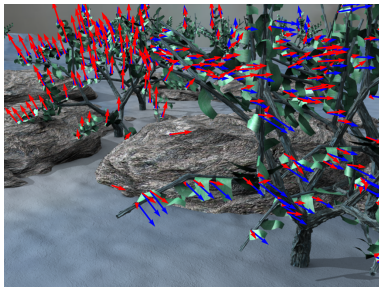




2.2. Forward-Additive Sparse Optical Flow

1. Visualising dense flow using the given .flo files



2. Sparse LK implementation parameters:
   `tau=0.01 ; window=(32,32)`
3. All the frame-wise results + a video depicting all the flows in sequence have been included in /results. Nevertheless, here's the comparison for optical flow between frames 10 and 11.
   Note: Red arrows = my implementation ; Blue = OpenCV implementation

4. Videos included in /results
5. EPE obtained:

Data	RubberWhale	Urban2	Grove3
My EPE	0.992	10.72	3.80
OpenCV EPE	0.182	1.56	1.27

2.3 Analysis of Single-Scale Lucas-Kanade Method

2.3.1 Local Tensor + Threshold

If the structure tensor $A^T A$ has rank 1, then it is rank deficient, which implies it is a singular matrix and therefore isn't invertible. Hence, the optical flow simply cannot be calculated in any case other than if the rank is 2.

We threshold the eigenvalues of the structure tensor, ensuring both of them are significant in terms of value. If λ_1, λ_2 are both small, the gradients are weak (just as in Harris corner detection), and the equations for all the pixels in the window would be more or less the same $\Rightarrow$ cannot reliably compute optical flow.

Additionally, we should also ensure that $\lambda_1 >> \lambda_2$ (or vice versa) does not occur, because that would correspond to there being a prominent edge in the window. Yet again, the optical flow computed here won't be reliable owing to the aperture problem as described above.

2.3.2 Modifications and Thresholds

General Observations:

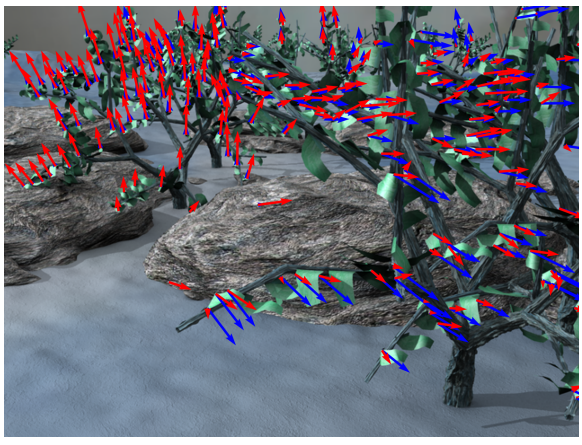
- The single-scale algorithm works extremely well for RubberWhale. This probably has to do with the fact that it has a lot of evenly spaced keypoints, with most of them moving slowly in the same direction.
- When motion gets chaotic in terms of direction, or quick in the temporal frame, the method tends to fall apart. This is probably why the EPE on Grove and Urban is considerably higher than in RubberWhale.

Modifications:

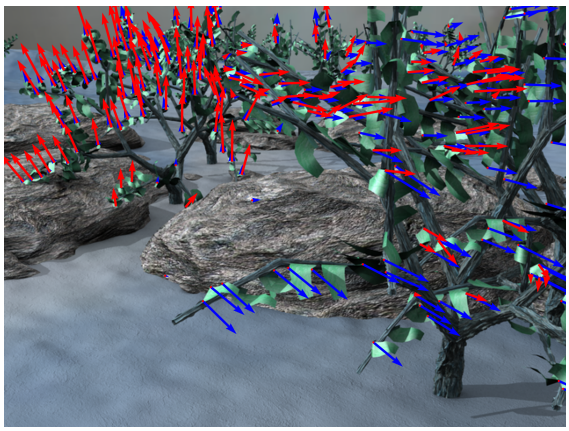
- In `cv2.goodFeaturesToTrack()` I seek features which are at least 10 pixels apart, for ease of visualisation.
- In the visualisation itself, I have added the parameter `angles='xy'` to the quiver plot function in order to represent direction more accurately by plotting in the data space ($(x, y) \rightarrow (x + u, y + v)$ instead of the screen space.
- I employed a 'hack' in order to report accurate EPE values for RubberWhale. Due to some error in reading from the .flo file for RubberWhale, all EPE values (including the one for OpenCV's implementation) had very high values (1 billion+). Hence, I thresholded the EPE calculation to upto a billion, which I deem reasonable.

Thresholding experiment:

- Varying the value of Tau to sub-0.01 levels has little to no effect. This is likely because we are already choosing strong features to track (using Harris detection), resulting in decent eigenvalues for the structure tensor.
- Increasing the Tau threshold to a very large value (say 100) does cut off several points from being displayed. However, these points are not noticeably 'stronger' as the EPE values remain more or less the same.



Tau = 0.01 → EPE = 3.80



Tau = 10 → EPE remains 3.80 (insignificant change)

2.3.3 Window Sizes Experimentation

For reasonably small window sizes (5x5, 12x12), the change in EPE values isn't very significant upon varying window sizes. This is probably because we are able to predict the optical flow reasonably well across the board, as the immediate vicinities of keypoints tend to have motion-wise coherency with the motion of the keypoint itself.

However, upon increasing the window size to 64x64 and 128x128 from my default of 32x32, a significant increase in EPE is found (especially for the latter), due to the loss of motion coherency under such a large area.

The tradeoff here is 'having more equations to work with in the window's span, thereby allowing us to get a better least squares estimate' vs 'having so much data that much of it is plain wrong with respect to the keypoint's motion, thereby misguiding our estimate completely'

Here's a summary of EPE values for single scale LK for frame10-11 of the 'Grove' set with varying window sizes:

Window size	5x5	12x12	32x32	64x64	128x128
EPE	2378	3.79	3.80	3.82	10.7

2.3.4 HSV

Instead of writing out optical flow as plain numbers, utilising the HSV space seems like a convenient way to encode the same. Hue represents the direction of optical flow vectors, while Saturation can be used to encode the magnitude. The V component remains at its maximum value, meaning that the colors are as vivid as possible. The Middlebury Optical Flow dataset probably uses this format because it provides an intuitive representation of flow fields.

---

3. Analysis of Multi-Scale LK Method

I have chosen my default number of levels to be 4.

Here's a summary of EPE values for single scale LK for frame10-11 of the 'Urban' set with varying numLevels values:

Number of Levels	2	4	6
EPE	10.32	8.61	6.96

Since we are refining our estimate of optical flow iteratively for different levels of blurring and scaling, having more levels decreases the error. At each level, we keep adding smaller and smaller refinements to inch closer to the ground truth flow.

3.1 EPE Values

Predictably, the multi-scale EPE values are lower across the board.

Data	RubberWhale	Urban2	Grove3
My SingleScale EPE	0.992	10.72	3.801
My MultiScale EPE	0.787	8.61	2.903

OpenCV EPE	0.182	1.56	1.27

3.2 Images with the Optical Flow

All the frame-wise results + a video depicting all the flows in sequence have been included in /results. Nevertheless, here's the comparison for optical flow between frames 10 and 11.

Note: Red arrows = my implementation ; Blue = OpenCV implementation

Single Scale (LHS) Multi Scale (RHS)