Homework 6: Optical Flow with the Lucas-Kanade Algorithm

Hayleigh Sanders RIN#661195735

1. Introduction

Optical flow is a vector field representing a projection of an image motion field. These vectors are a useful way to estimate the motion of pixels given two images of a moving scene. The Lucas-Kanade algorithm is a simple method using a local approach to estimate the optical flow for each pixel in an image. It uses first order image derivatives and the assumption that optical flow vectors are the same for each pixel in a relative area to solve for the optical flow of each pixel with a least-squares solution. In the following report, this algorithm is applied to two image sequences, each consisting of five frames, to obtain their optical flow.

2. Theory of Motion Analysis

In a grayscale image, each pixel *(c,r)* corresponds to an intensity value *I*. In a sequence of images separated by a time interval *t*, the intensity of each pixel will not change between time *t* and *t+1*, shown in (1).

$$\frac{dI}{dt} = 0 \qquad (1)$$

This constraint is completely satisfied if the motion is translational, or the illumination direction is parallel to the angular velocity for a lambertian surface. The intensity *I* is a function of coordinates *(c,r)*, which are a function of time *t*, shown in equation (2).

$$\frac{\partial I}{\partial t} = \frac{\partial I}{\partial c}\frac{dc}{dt} + \frac{\partial I}{\partial r}\frac{dr}{dt} + \frac{\partial I}{\partial t} = 0 \qquad (2)$$

The spatial intensity gradient is represented by $\partial I/\partial c$ and $\partial I/\partial r$. The temporal intensity gradient is represented by $\partial I/\partial t$.

The image motion field v can be found from the image brightness constancy equation (3), which follows from equation (2), where ∇I is the image intensity gradient:

$$(\nabla I)^{t} v + I_t = 0 \qquad (3)$$

The image brightness constancy equation does not have a constraint on *v* when it is orthogonal to ∇I, which means that the motion flow component can only be found in the direction of the image intensity gradient. Therefore the motion field, known as optical flow, must be parallel to the image gradient.

Let *p* be the pixel at the center of an NxN block of pixels in an image. Assume every point in the block has the same optical flow, *v*. This assumption serves as a smoothness constraint. Therefore the relationship between the spatial image derivatives $I_c(c,r)$, $I_r(c,r)$, temporal image derivative $I_t(c,r)$, and optical flow *v* follows the equation:

$$I_c(c_i, r_i)\, v_c + I_r(c_i, r_i)\, v_r + I_t(c_i, r_i) = 0 \qquad (c_i, r_i) \in R_i \qquad (4)$$

(v_c, v_r) is found by minimizing:

$$\varepsilon^2 = \sum_{(c_i, r_i) \in R_i} \left[I_c(c_i, r_i)\, v_c + I_r(c_i, r_i)\, v_r + I_t(c_i, r_i) \right]^2 \qquad (5)$$

For each pixel in a NxN neighborhood, equation (4) can be used to form a system of equations to solve for the unknown *v(x,y)*. The least-squares solution to *v(x,y)* corresponding to the optical flow at point *p* will be:

$$v(x, y) = (A^t A)^{-1} A^t b \qquad (6)$$

Where *A* and *b* are matrices given by:

$$A = \begin{bmatrix} I_c(c_1, r_1) & I_r(c_1, r_1) \\ I_c(c_2, r_2) & I_r(c_2, r_2) \\ \vdots & \vdots \\ I_c(c_N, r_N) & I_r(c_N, r_N) \end{bmatrix} \qquad (7)$$

$$b = -\left[I_t(c_1, r_1), I_t(c_2, r_2), \ldots, I_t(c_N, r_N) \right]^t \qquad (8)$$

3. Algorithm and Results

The following algorithm was implemented:

1. Select two images at time *t* and *t+1*, beginning with *t=0*
2. Compute the spatial (I_x, I_y) and temporal (I_t) image derivatives using Sobel operators which are convolved with the original images
3. For each NxN window with center pixel *p*, find the *A* and *b* matrices from equations (7) and (8) using the image derivatives from step 2.
4. Find the optical flow vector *v(x,y)* corresponding to the center pixel *p* using equation (6) and the matrices computed from step 3.
5. If the magnitude of *v(x,y)* is very small or if the condition of *A* is large, set *v(x,y)* to zero.
6. Repeat steps 1 to 5 for all pixels in the image

7. Repeat steps 1 to 6 for each image pair at *t* and *t+1*

The algorithm was performed on two image sequences, each with five frames. The optical flow vectors were plotted on the second image from each image pair using the quiver function in Python's matplotlib library. Figures 1, 2, 3 and 4 display the optical flow between each frame of the first sequence. Figures 5, 6, 7 and 8 display the optical flow between each frame of the second sequence.

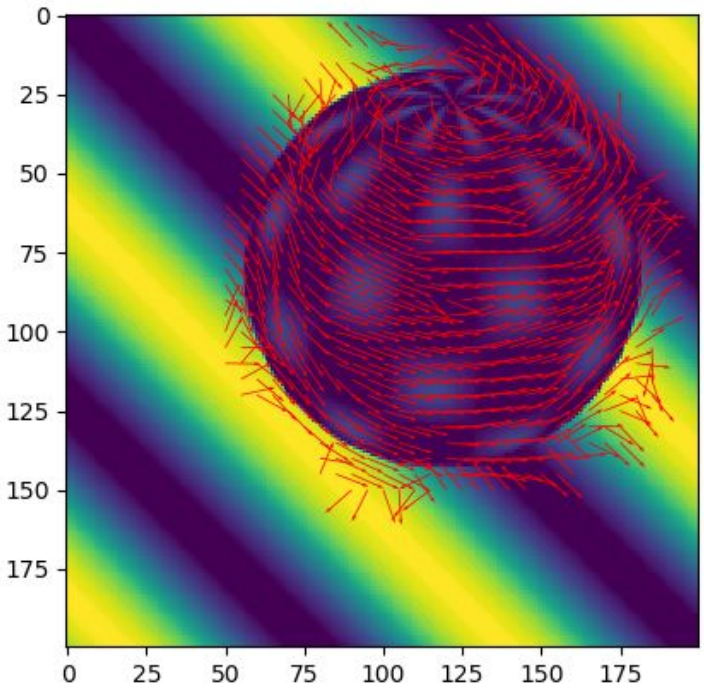

Figure 1: Sequence 1, optical flow from image 1 to 2

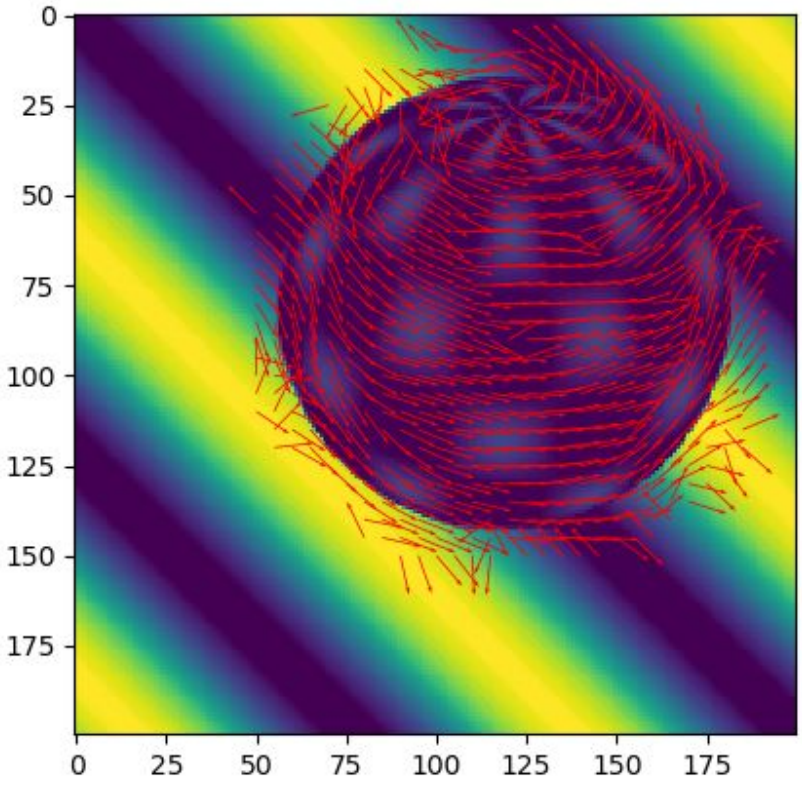

Figure 2: Sequence 1, optical flow from image 2 to 3

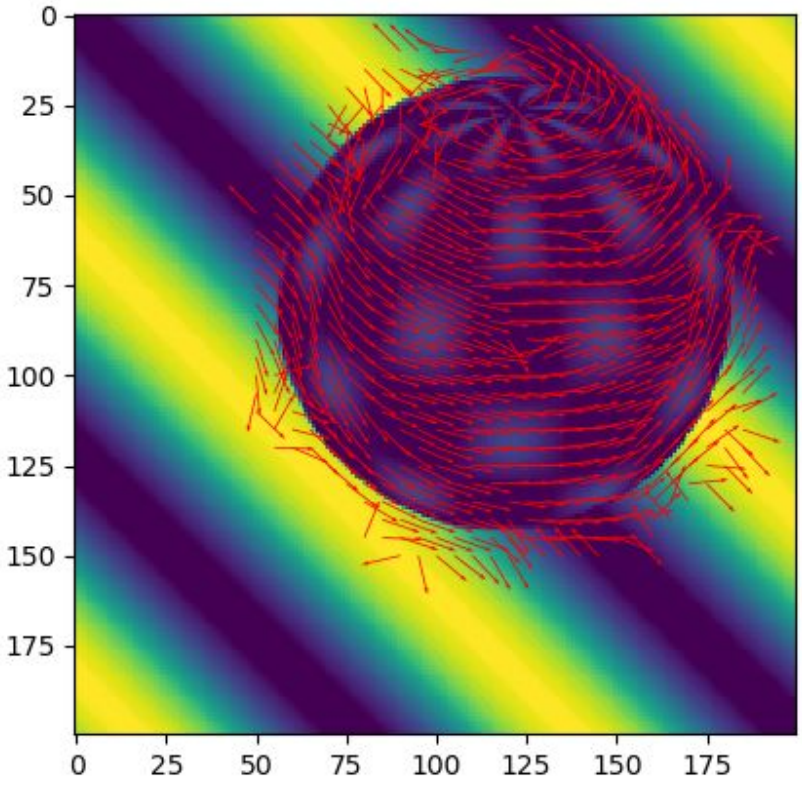

Figure 3: Sequence 1, optical flow from image 3 to 4

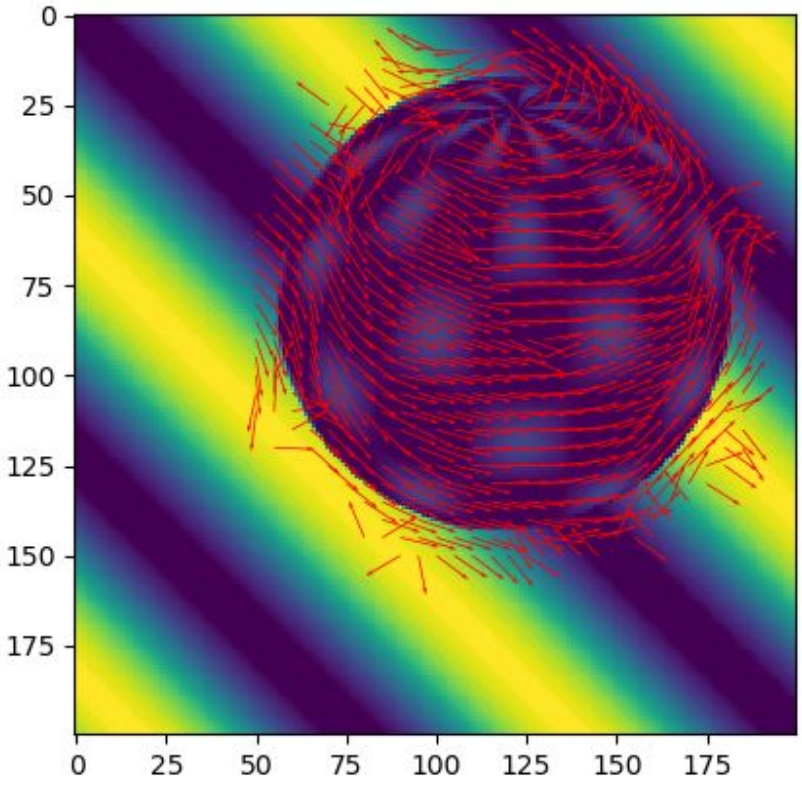

Figure 4: Sequence 1, optical flow from image 4 to 5

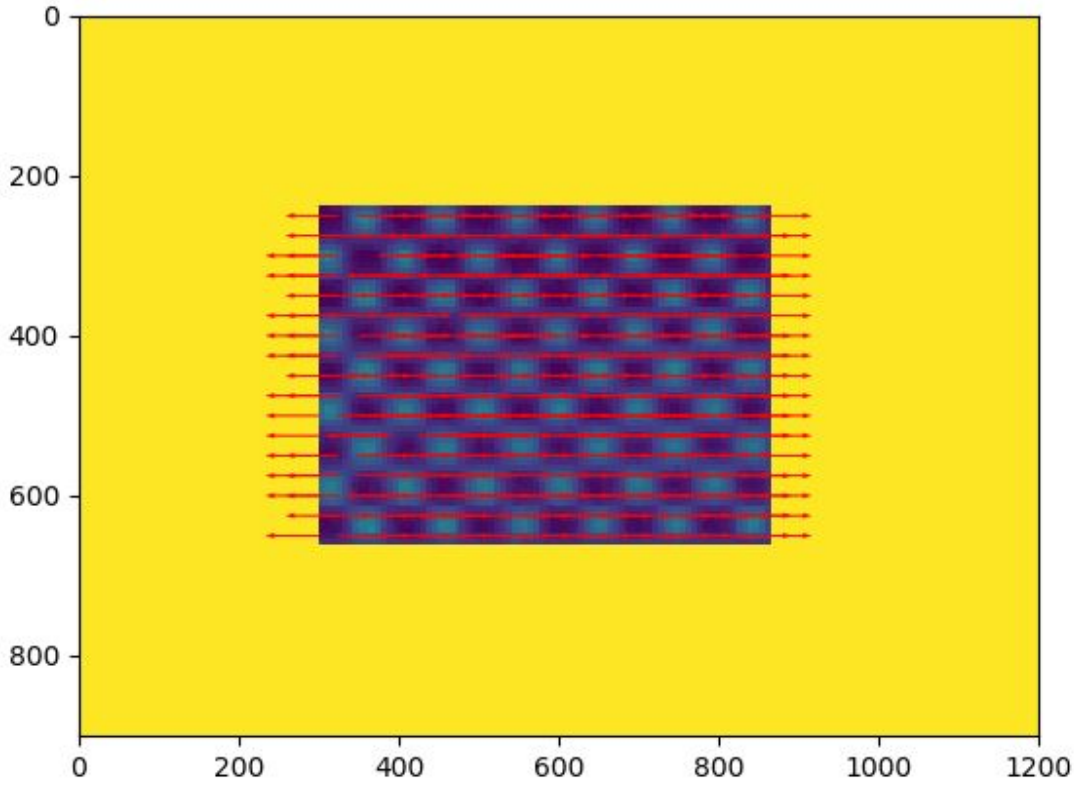

Figure 5: Sequence 2, optical flow from image 1 to 2

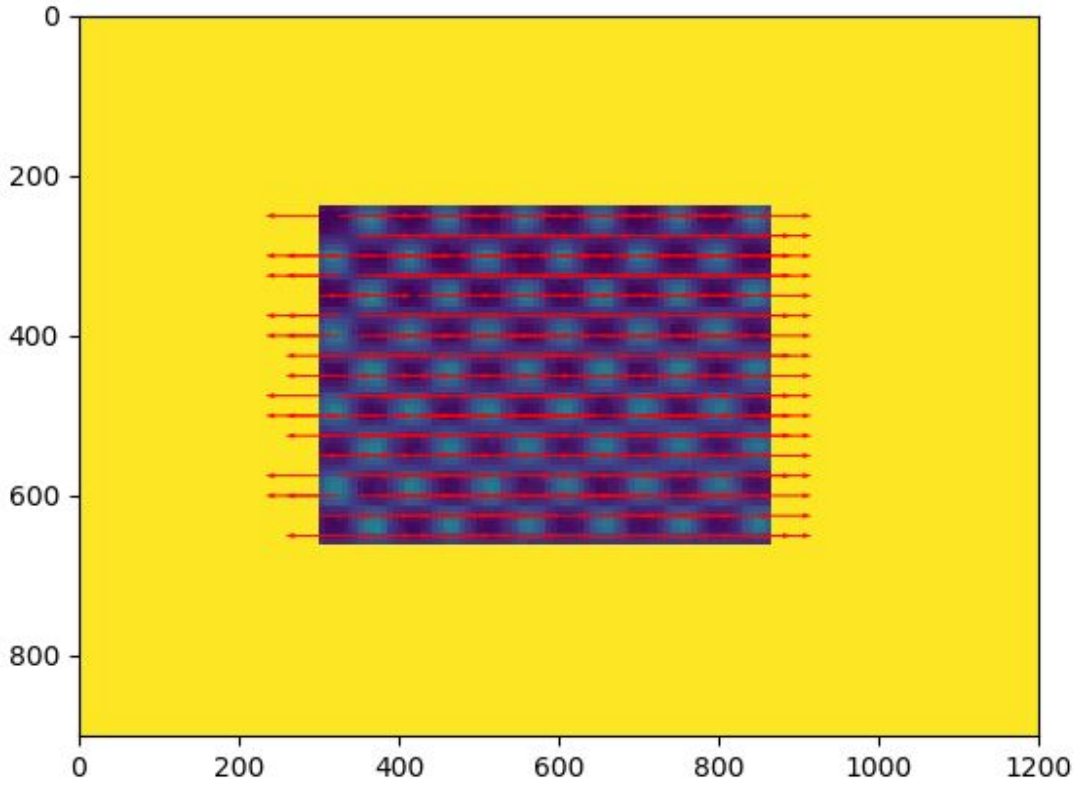

Figure 6: Sequence 2, optical flow from image 2 to 3

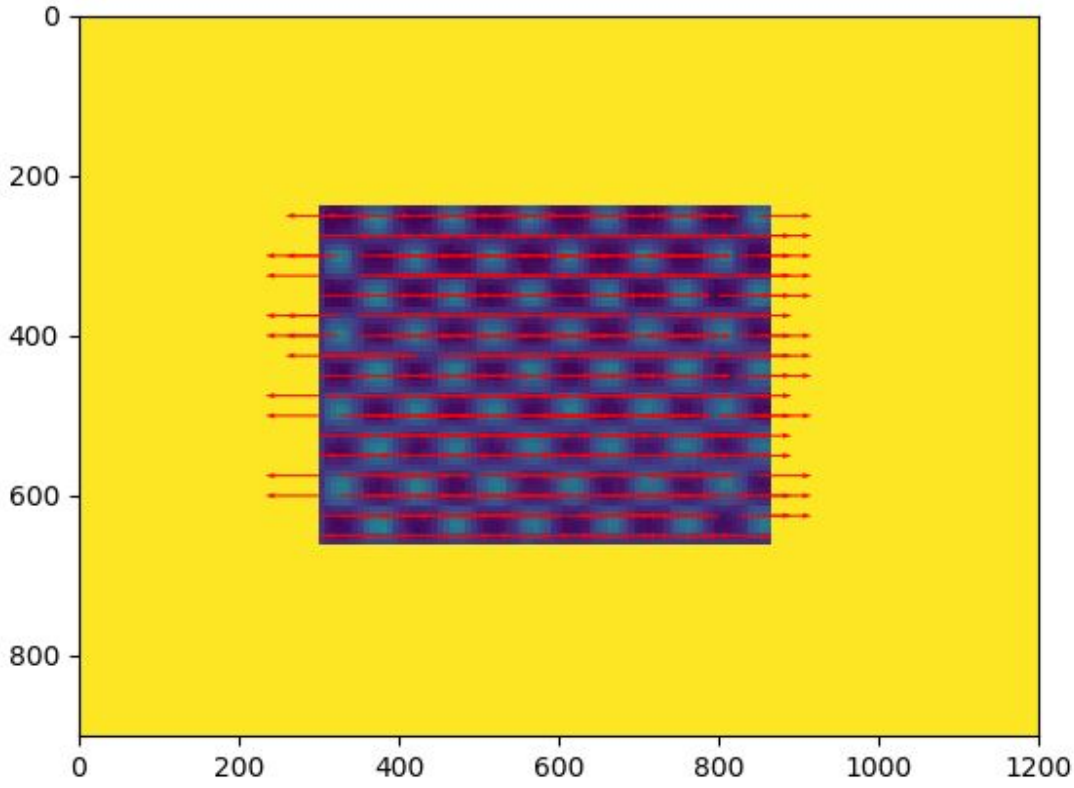

Figure 7: Sequence 2, optical flow from image 3 to 4

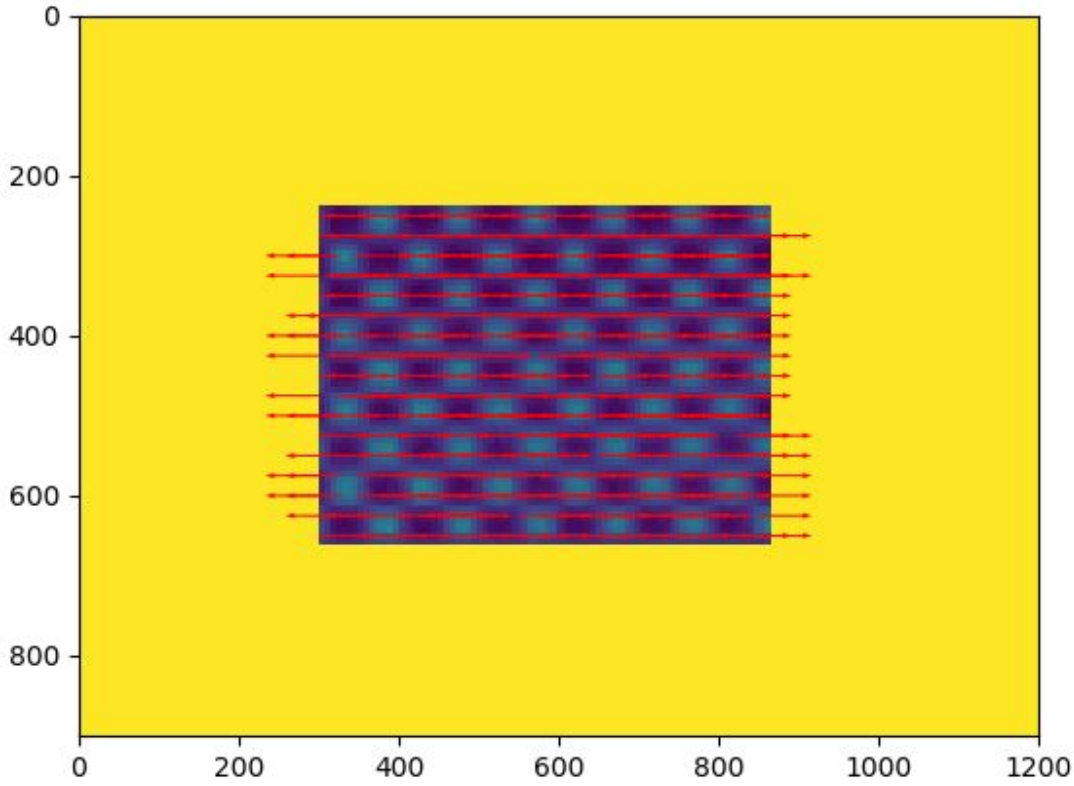

Figure 8: Sequence 2, optical flow from image 4 to 5

For each sequence, optical flow vectors were plotted in intervals because the output was very dense and made it difficult to visualize the total optical flow. For the first sequence, the optical flow vector every five pixels was plotted and for the second sequence, the optical flow vector every 25 pixels was plotted. The window size was manually adjusted for each sequence. It was found that a window size of 20 for the first sequence and 5 for the second sequence worked well to produce an output with enough optical flow vectors. Likewise, the optical flow vector magnitude and A matrix condition threshold from step 5 were manually adjusted until the vector fields did not have large gaps over areas of motion.

From Figures 1 to 4, it is clear from the optical flow vectors that the sequence displays rotational motion and the sphere in the image is rotating counterclockwise. Likewise, from Figures 5 to 8, the sequence displays translational motion and the checkerboard pattern is moving to the right.

The optical flow vector directions are inconsistent around the edges of moving objects in both sequences. For the first sequence, this distortion is noted around the edges of the sphere and in the second sequence it is displayed around the edges of the checkerboard. This is because of the smoothness constraint in equation (4) which assumes every point in a NxN neighborhood

has the same optical flow, *v*. This assumption does not hold around the edges of moving objects in a sequence and is demonstrated in the results.

4. Summary and Conclusion

The optical flow for both sequences was successfully computed and plotted using the Lucas-Kanade algorithm. The optical flow vectors between each image reveal that the sphere in the first sequence is rotating counterclockwise and the checkerboard in the second sequence is moving to the right with translational motion. The optical flow vectors did exhibit distortion around the edges of moving objects in each sequence, which is to be expected from the assumption made by the algorithm that the optical flow for all pixels in a NxN neighborhood is the same.