Assignment 4: Optical Flow

Zebin Xu (zebinxu@nyu.edu)
Department of Computer Science and Engineering
NYU Tandon Schoo of Engineering
May 4, 2017

1. Optical Flow

1.1 Normal Flow

1.1.1 Brief Explanation

Given the assumption of brightness consistency:

$$I_x u + I_y v + I_t = 0 \qquad (1)$$

We need one more constraint to solve the equation. Normal flow is the pixel movement parallel to the image gradient or perpendicular to the boundaries.

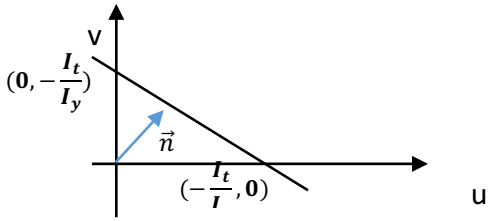

Figure 1. Optical flow equation line on u, v space

The blue arrow perpendicular to the line in Figure 1 is the direction of the normal flow. Point $(-\frac{I_t}{I_x}, 0)$ and $(0, -\frac{I_t}{I_y})$ intersects u and v axis respectively. Let $\vec{n}$ be the normal flow vector, $\vec{V} = [-\frac{I_t}{I_x}, \frac{I_t}{I_y}]^T$,we have:

$$\vec{V} \cdot \vec{n} = 0 \qquad (2)$$

From equation (2) we can calculate the relation between u and v of normal flow:

$$u = \frac{I_x}{I_y} v \qquad (3)$$

Plug (3) into (1) we get the u and v of normal flow.

$$\begin{cases} u = -\frac{I_t I_x}{{I_x}^2 + {I_y}^2} \\ v = -\frac{I_t I_y}{{I_x}^2 + {I_y}^2} \end{cases} \qquad (4)$$

1.1.2 Results and Discussion

We first analyze the following first two consecutive images of a video sequence. In this short video sequence, the left car is moving left while the two right cars are moving right. There is also a small camera motion moving up.

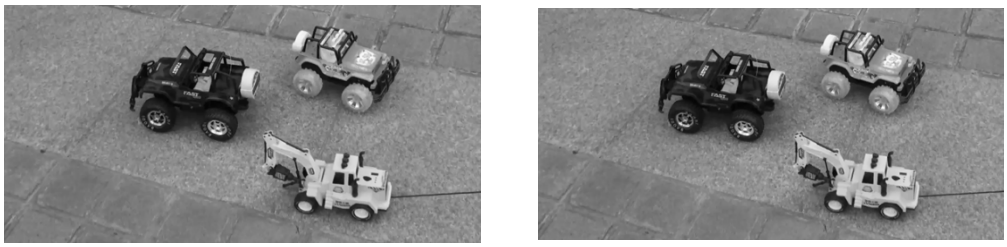

Figure 2. Original Images (toy_formatted2.png and toy_formatted3.png)

We first blur the images using the provided Gaussian filter function with sigma 1. Because we assume the edges in the images are smooth. This smoothing process can avoid sharp edges that may not be suitable for our derivative computation.

We then compute spatial gradients on x and y direction and temporal gradient on the two images. Figure 3 and 4 shows the spatial gradients and the temporal gradient. It can be seen that there are mainly vertical sharp edges in the horizontal (left) spatial gradient image, and horizontal sharp edges in the vertical (right) spatial gradient image. This conforms with our spatial gradient operation. Since the spatial gradient on x is equal to doing convolution using the filter [-1, 0, 1] and the spatial gradient on y is equivalent to using a filter $[-1, 0, 1]^T$. This operation magnifies horizontal and vertical spatial gradient of the images.

The same theory applies to the temporal gradient operation in which we see sharp vertical edges that indicate the horizontal movements.

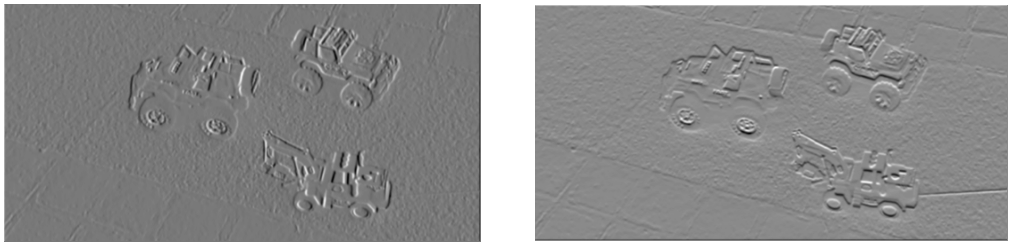

Figure 3. Spatial gradients on x and y (left and right)

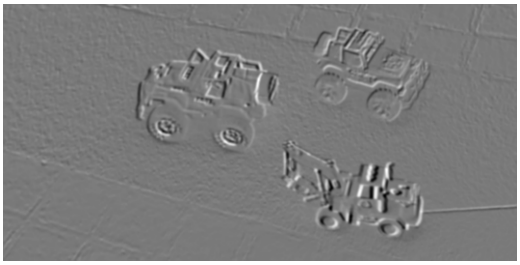

Figure 4. Time Gradient

Then we can compute normal flow of every pixel using equation (4). Since the movement between two images is very small, we magnify the flow by 10 so as to make them visible. The result is shown in Figure 5. The flow in the background can be explained by the small motion of camera.

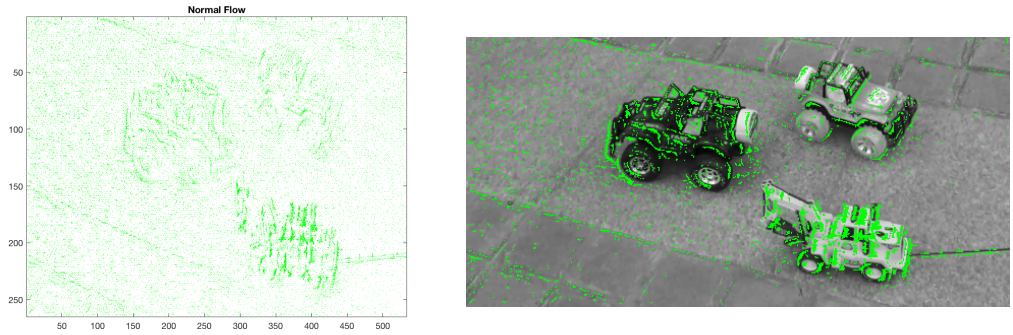

Figure 5. 10x magnified normal flow and flow overlaid on the original image

To better view the flow, we take 3 zoomed patch. It can be seen in Figure 6 that the flow has a lot of noise. But we can largely see horizontal flow on the right direction in the second and third picture and left direction in the first. Since normal flow is estimated on every single pixel and is parallel to the image gradient, it makes sense that flow vectors are not all consistent. The flow noise may also result from the small camera motion and the images' original noise. In the third patch we can see obvious horizontal right flow which mostly appears at the car's edges. These verify normal flow's properties.

Figure 6. Zoomed view of 3 cars with overlaid normal flow

We can also directly display the binary image of the normal flow in x and y direction. It can be seen in the left image the contours of two cars have white color, which tells that their x components of normal flow are 1 after a threshold as a binary image. This verifies that they are moving right. The left car moving left is black because the normal flow around it is negative value and shown 0 after a threshold.

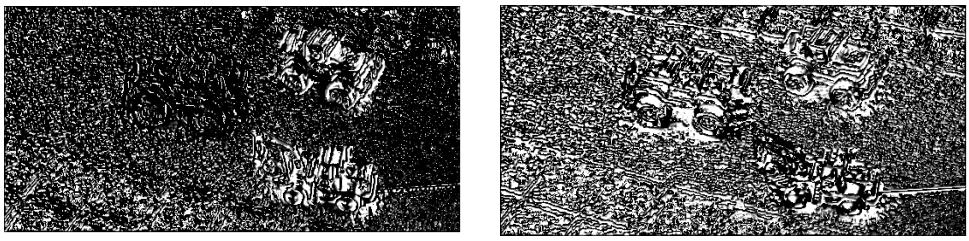

Figure 6. Binary image of normal flow on x and y direction.

We then apply the same procedure for the last two consecutive images (toy_formatted8.png and toy_formatted9.png) in the video sequence. We compare these two results and find that they have almost identical results. This shows that our normal flow algorithm successfully estimates the cars motion although there are noises.

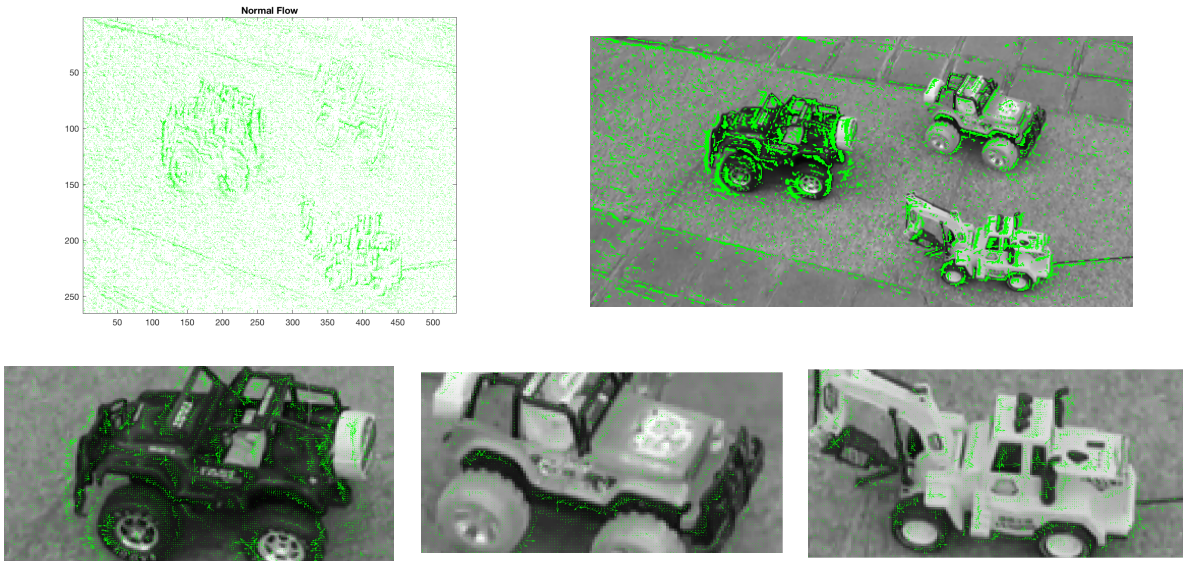

Figure 7. Normal flow, overlaid normal flow, zoomed patches for the last two consecutive images.

1.2 Flow calculated over pixel neighborhood

1.2.1 Brief Explanation

Assuming a 2x2 pixel neighborhood has the same velocity, for each pixel in the 2x2 patch, we can write the equation (5) given the assumption in (1):

$$A\vec{v} = b \qquad (5)$$

where $A = \begin{bmatrix} I_{x0} & I_{y0} \\ I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ I_{x3} & I_{x3} \end{bmatrix}$, $b = -\begin{bmatrix} I_{t0} \\ I_{t1} \\ I_{t2} \\ I_{t3} \end{bmatrix}$. I_{xi}, I_{yi} is the spatial gradient in x and y direction at the **ith** pixel of neighborhood respectively, I_{ti} is the temporal gradient at time **t** at **ith** pixel (i=0,1,2,3).

We can solve $\vec{v} = (u, v)^T$ in this over-constrained system with 4 measurements using the least square method:

$$\vec{v} = (A^T A)^{-1} A^T b \qquad (6)$$

1.2.2 Results and Discussion

After image smoothing, spatial and temporal differentiation same as the steps in normal flow, we choose a 2x2 pixel neighborhoods for every pixel and compute the flow using equation (6) for the first two consecutive images. We compare the flow results in this part in three different ways of flow calculation: 1. Directly to raw images. 2. Images filtered by a Gaussian smoothing of sigma=1, and 3. images filtered by a Gaussian smoothing of sigma=2.

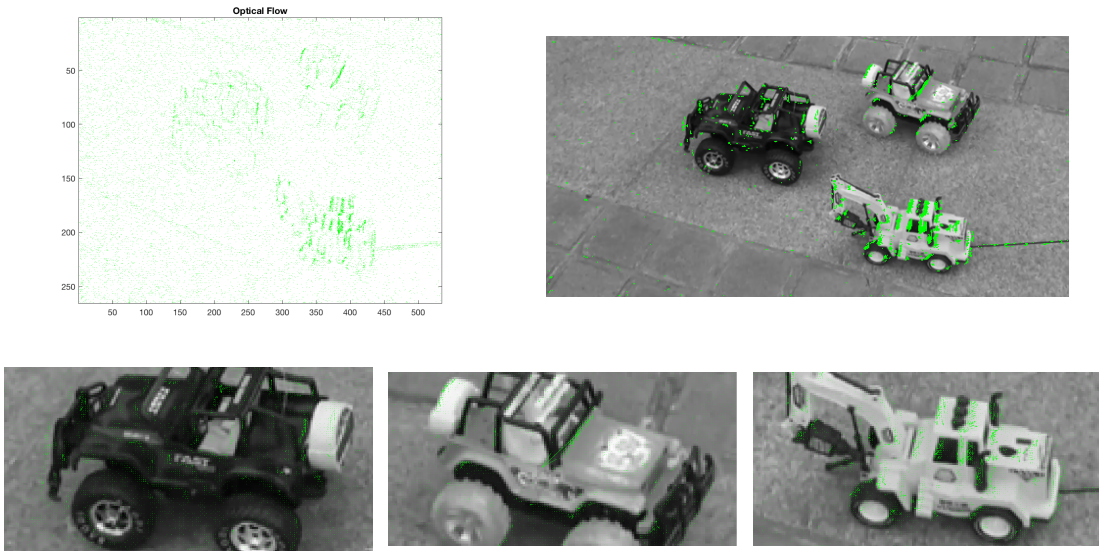

Figure 8. 2x2 pixel neighborhoods calculated flow to raw images

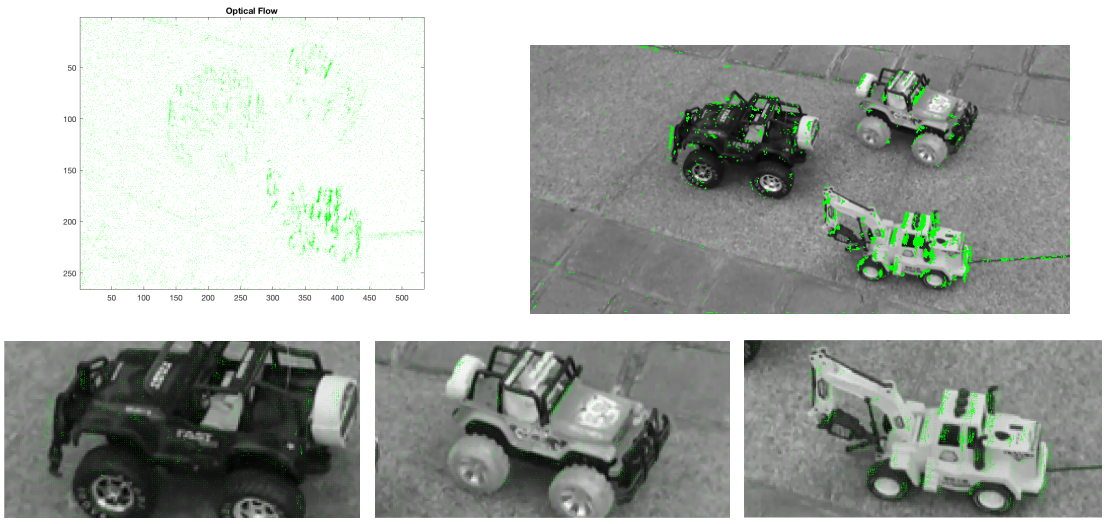

Figure 9. 2x2 pixel neighborhoods calculated flow to images filtered by a Gaussian smoothing of sigma=1

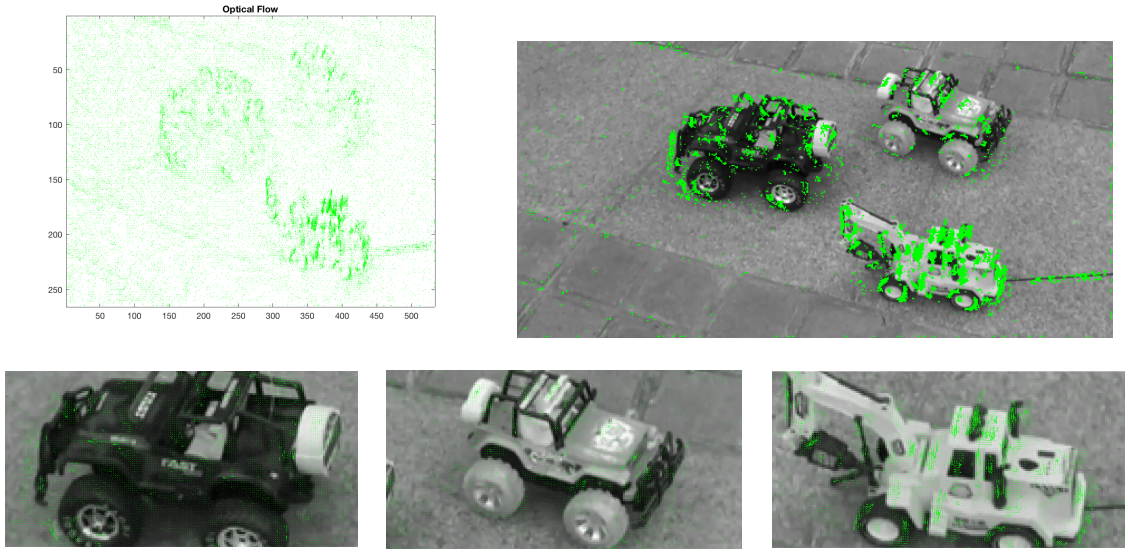

Figure 10. 2x2 pixel neighborhoods calculated flow to images filtered by a Gaussian smoothing of sigma=2

Since the 2x2 calculation of flow is also a locate estimate, it cannot provide flow information in textureless regions or interior uniform regions of edges in the image.

From Figure 8, 9, 10 we found that most flows appear at edges. The most obvious difference that can been clearly seen between normal flow and the flow calculated over 2x2 pixel neighborhoods is that normal flow has more background noises. For results of the calculation to raw images in Figure 8, we notice that the flows are much less than filtered images'. The flows intensity seems to increase as sigma or the smoothness of images increases. But this is not always the case. We tried playing by setting sigma to 10 and finally got a messy result! From the zoomed patch of Figure 8, 9, 10, we can see that when sigma = 2, the flow result is satisfying. The flows around the black car in Figure 10 indicate an obvious left motion.
We tried calculated flow using the first and last images (toy_formatted2.png and toy_formatted9.png) and got a lot of noisy flows. This is because we assume the motion to be small so if the motion is too large the calculation may fail.

Since the 2x2 size is not odd, we tried using a 4 connected neighborhoods of every pixel (size of 5) to compute the flow, which I think may get better result since the pixel we are calculating is in the center of the kernel. Figure 11 shows that the result is better than 2x2 methods and can reflect the motion details like the wheels' motion and cars movement.

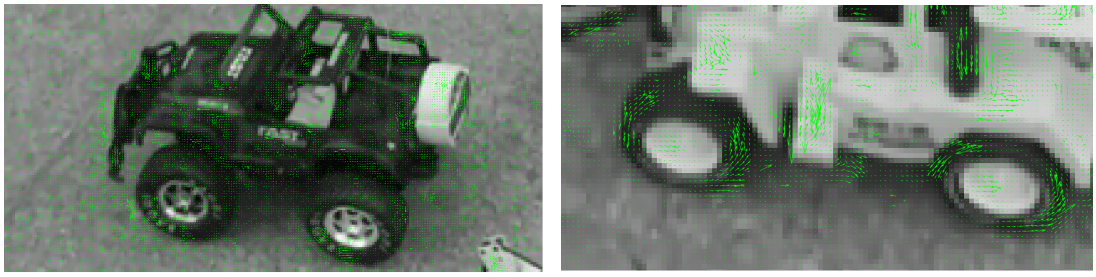

Figure 11. Zoomed patches of 4 connected neighborhoods (size of 5) calculated flow

Factors that may affect the correctness of flow includes the Gaussian filter kernel size, the size of neighborhood used to compute flow, the size and the noises of the images, and the motion between two images. A careful choice of these are needed to obtain a better result of optical flow.

2. Regularization – Horn and Schunk or Lucas and Kanade Method

Apply the Lucas and Kanade method with kernel size of 3, we obtain the following result. It can be seen that the flow is much more consistent and better than previous results.

Figure 12. Zoomed patch of flow calculated using Lucas and Kanade method

3. File Structure

```
.
├── computeNormalFlow.m
├── computeOpticalFlow.m
├── gaussian_filter.m
├── main.m
└── opticalFlowLK.m
```