Assignment 4 Optical Flow

Practical Report

Chuan Long

1. Optical Flow

The objective of the Optical Flow method is to reconstruct the displacement vector field of objects captured with a sequence of images. In fact, based on a set of images capturing the motion of one or multiple objects, we want to be able to reconstruct the displacement field associated to each pixel during the time difference from one frame to another.

1.1 Normal Flow

1.1.1 Equation Summary:

- Calculate the Temporal Gradient Image:

  I(x,t+1)-I(x,t)

- Calculate the Spatial Derivatives:

  E_x = I(x+1,y,t)-I(x,y,t)

  E_y = I(x,y+1,t)-I(x,y,t)

- Calculate the Normal Flow:

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

$$\nabla I \bullet \vec{U} = 0$$

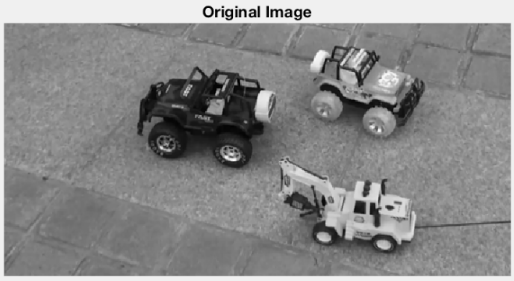
$$u_{\perp} = -\frac{I_t}{|\nabla I|}\frac{\nabla I}{|\nabla I|}$$

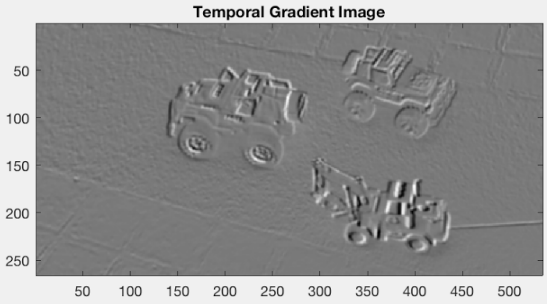
1.1.2 Results

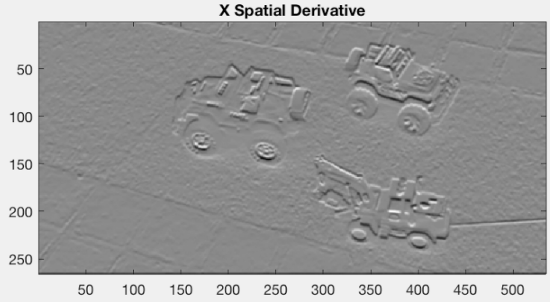
(In this case, the Gaussian Filter is 1.0, and the 2D vector image are shown as different styles: spacing = 20 and spacing = 1)

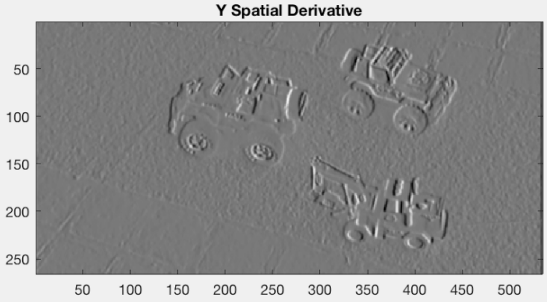
In this case, please notice that we assume the rows of the image is the X axis, the columns of the images is the Y axis, which are follow the spatial derivative equation. However, we will produce the same result if we exchange them. All the image are reproducible and code files are attached. Simply change the objects of the img_1 and img_2 could produce different rows of consecutive images.

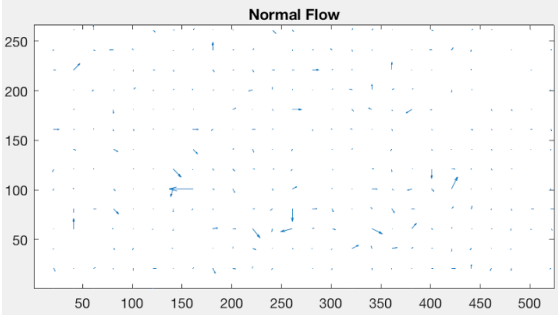
Toy_2 & Toy_3

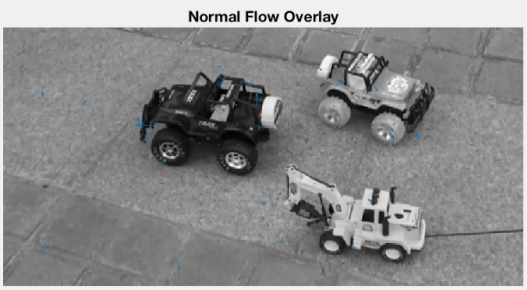

Original Image

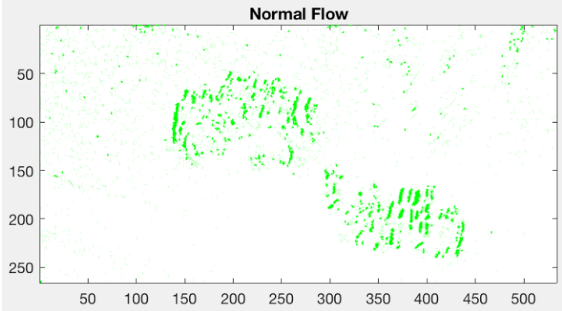

Temporal Gradient Image
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500

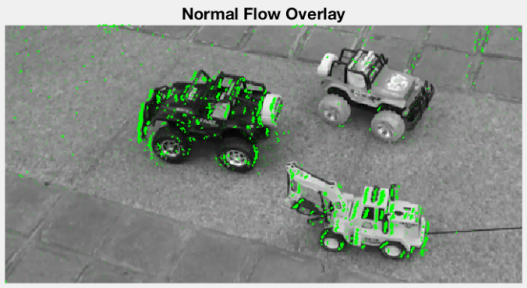

X Spatial Derivative
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500


Y Spatial Derivative
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500


Normal Flow
250
200
150
100
50
50 100 150 200 250 300 350 400 450 500

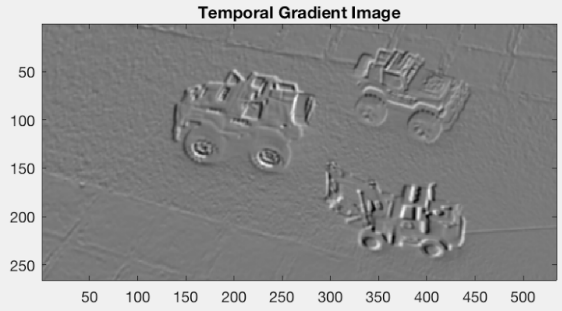

Normal Flow Overlay

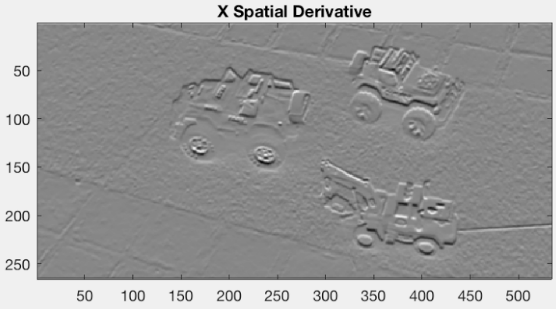

Normal Flow
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500

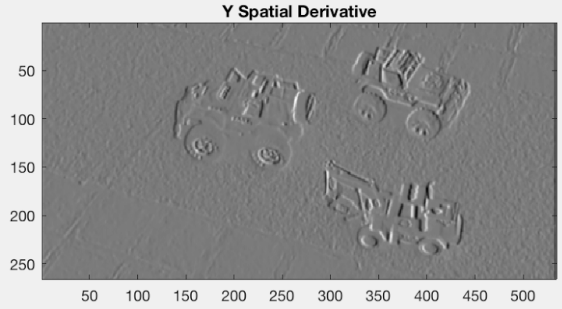

Normal Flow Overlay

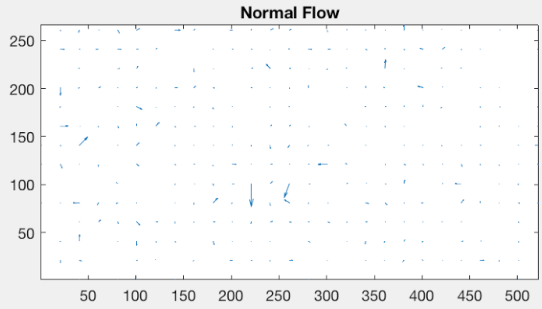
Toy_3 & Toy_4


Original Image

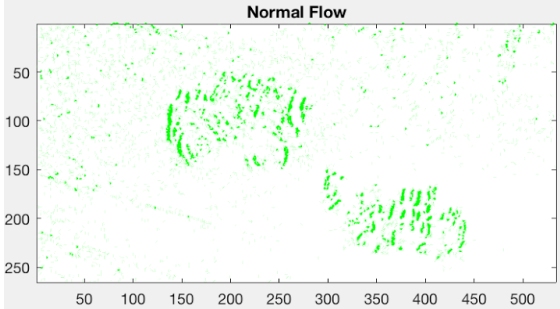

Temporal Gradient Image
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500

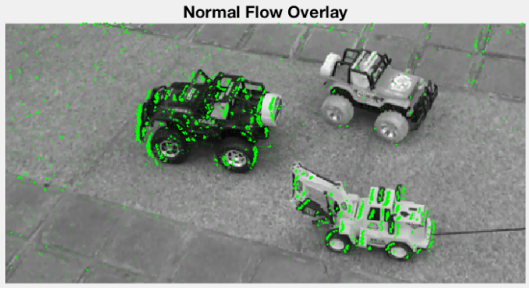

X Spatial Derivative
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500


Y Spatial Derivative
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500


Normal Flow
250
200
150
100
50
50 100 150 200 250 300 350 400 450 500

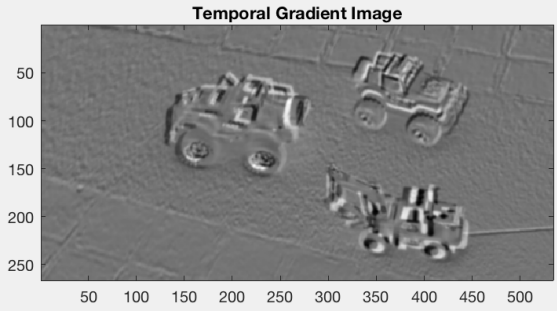

Normal Flow Overlay

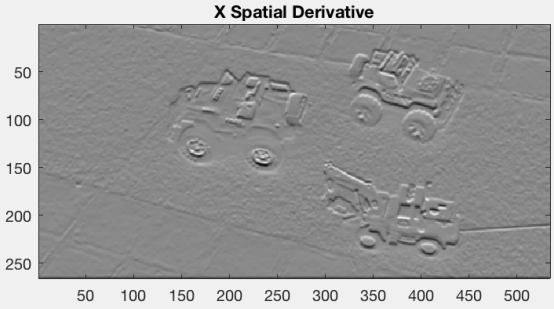

Normal Flow
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500

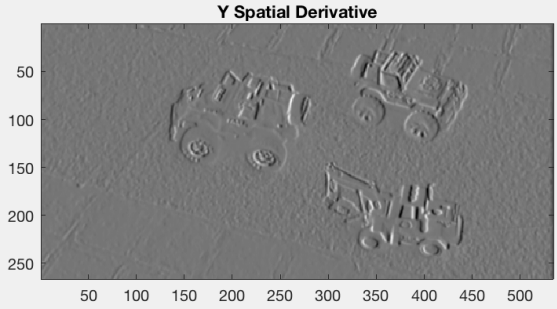

Normal Flow Overlay

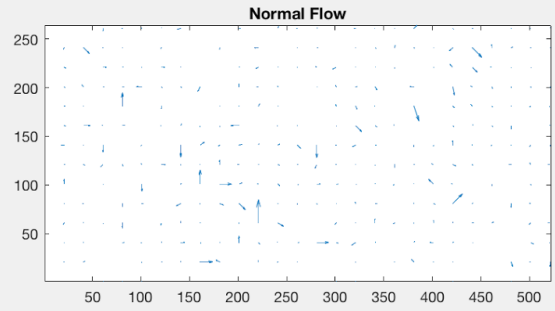
Toy_4 & Toy_5


Original Image

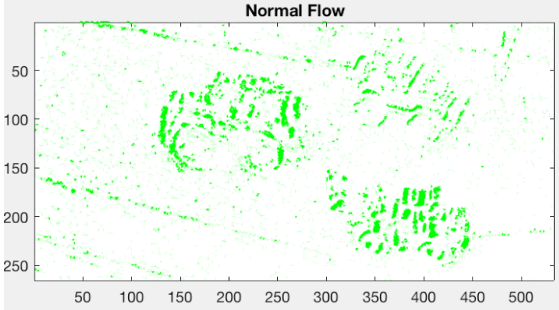

Temporal Gradient Image
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500

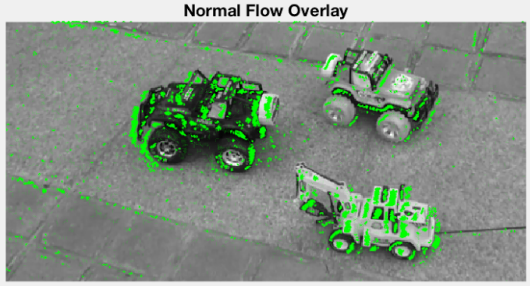

X Spatial Derivative
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500


Y Spatial Derivative
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500


Normal Flow
250
200
150
100
50
50 100 150 200 250 300 350 400 450 500

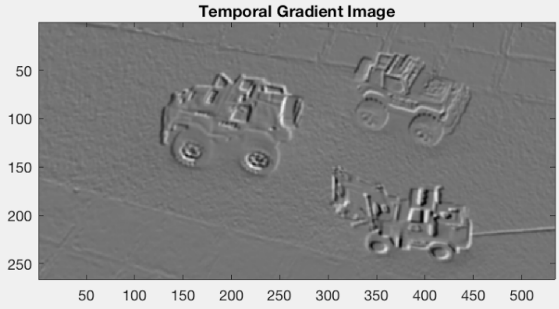

Normal Flow Overlay

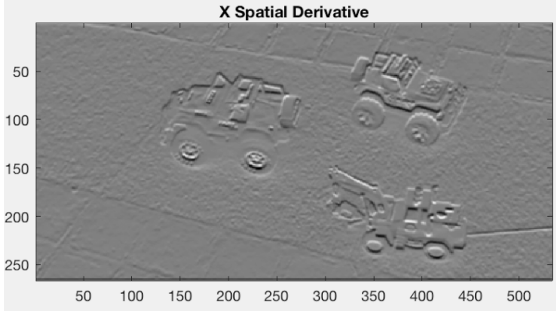

Normal Flow
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500

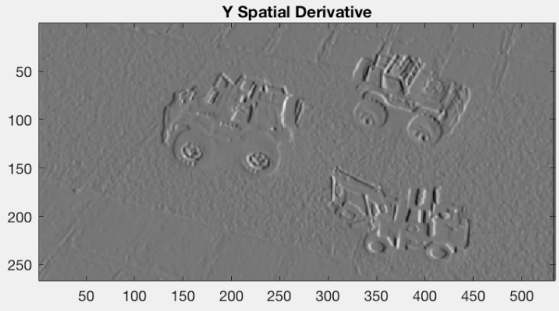

Normal Flow Overlay

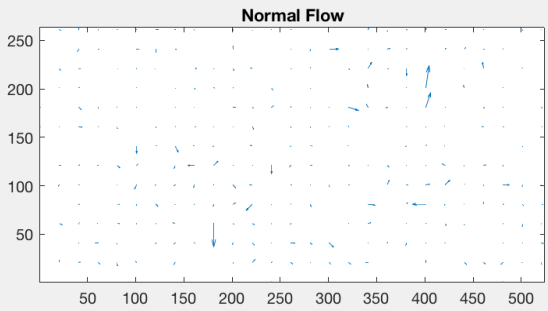
Toy_5 & Toy_6

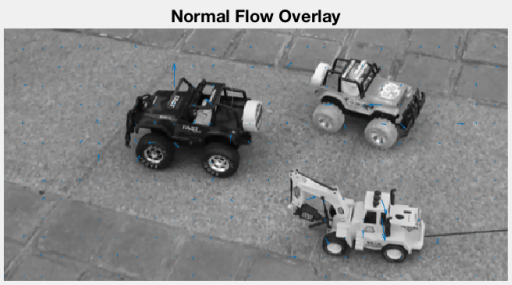

Original Image

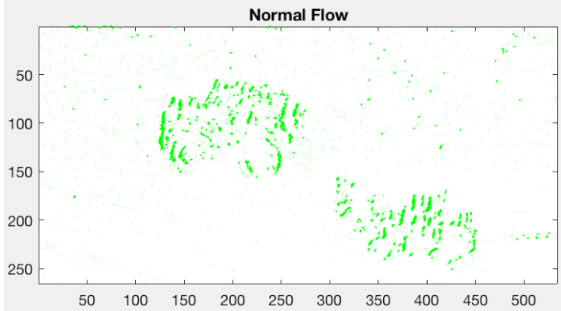

Temporal Gradient Image
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500

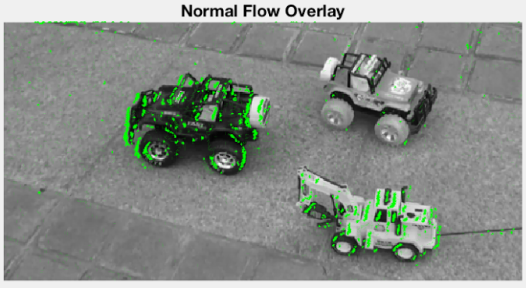

X Spatial Derivative
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500


Y Spatial Derivative
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500


Normal Flow
250
200
150
100
50
50 100 150 200 250 300 350 400 450 500

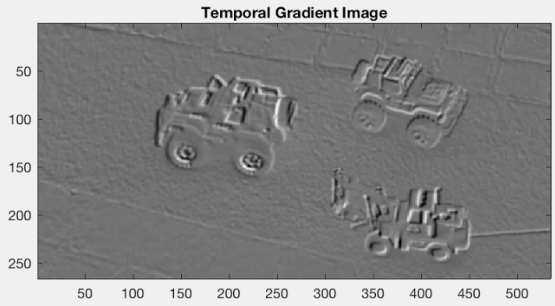

Normal Flow Overlay

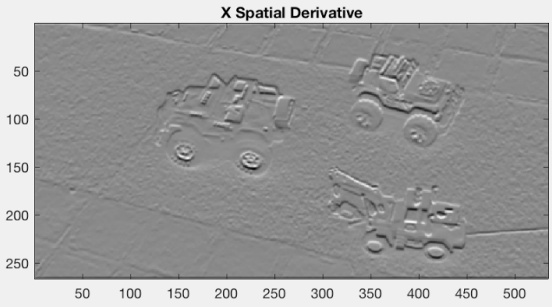

Normal Flow
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500

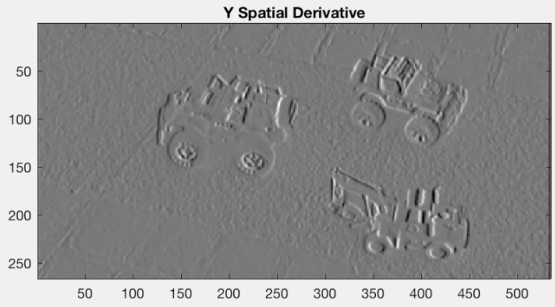

Normal Flow Overlay

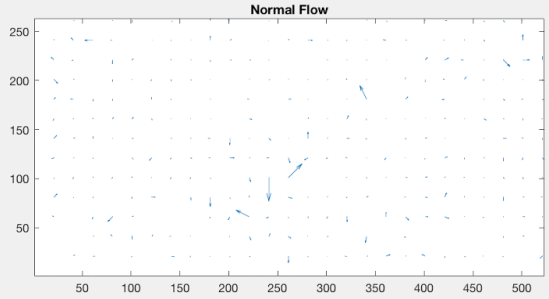
Toy_6 & Toy_7


Original Image

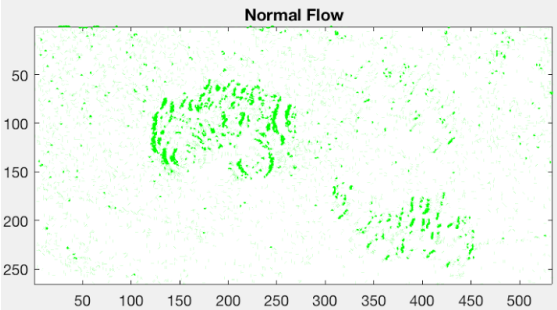

Temporal Gradient Image
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500

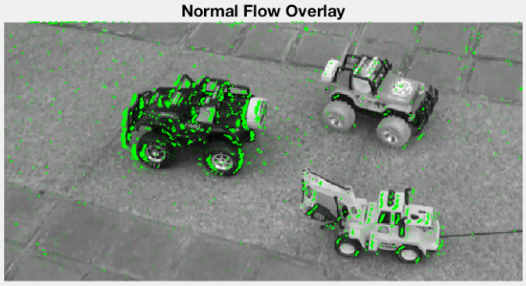

X Spatial Derivative
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500


Y Spatial Derivative
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500


Normal Flow
250
200
150
100
50
50 100 150 200 250 300 350 400 450 500

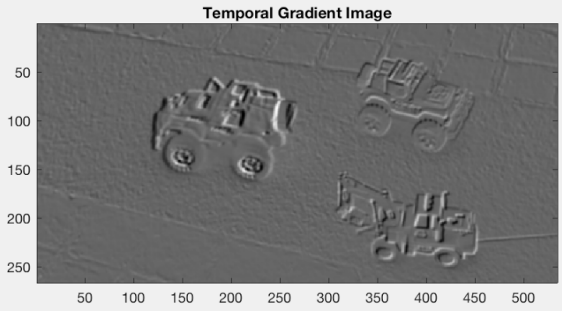

Normal Flow Overlay

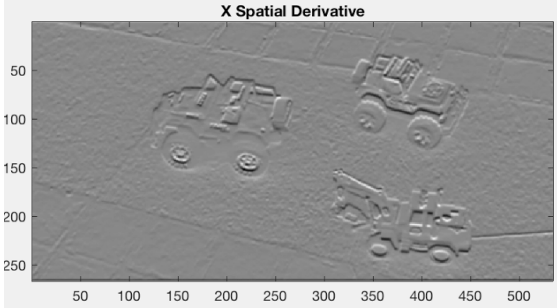

Normal Flow
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500

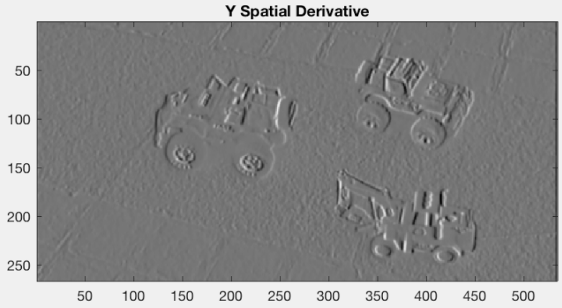

Normal Flow Overlay

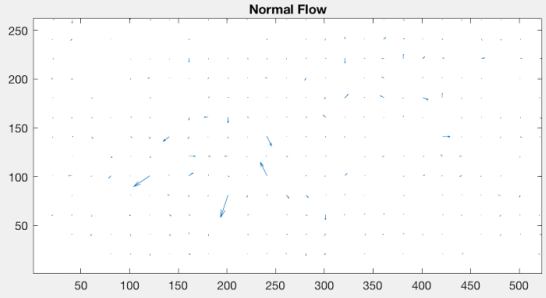
Toy_7 & Toy_8


Original Image

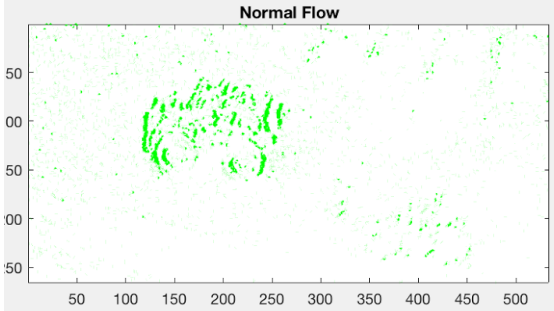

Temporal Gradient Image
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500

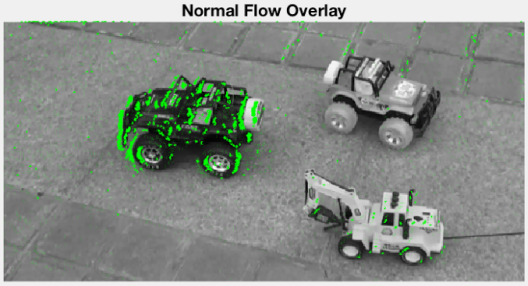

X Spatial Derivative
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500


Y Spatial Derivative
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500


Normal Flow
250
200
150
100
50
50 100 150 200 250 300 350 400 450 500

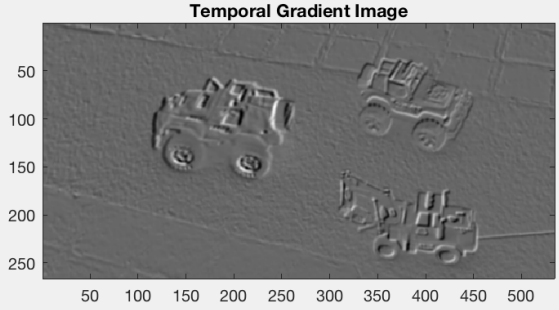

Normal Flow Overlay

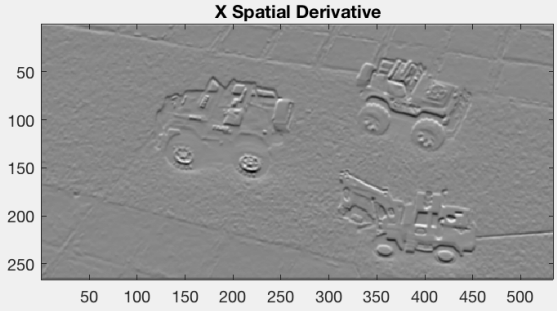

Normal Flow
50 100 150 200 250 300 350 400 450 500

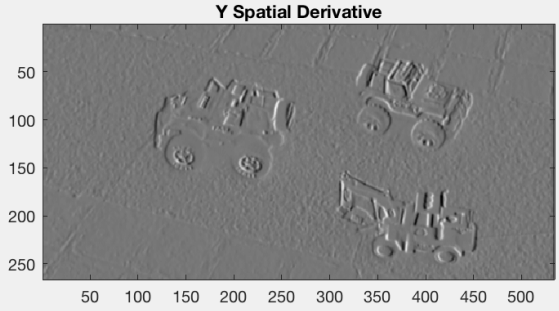

Normal Flow Overlay

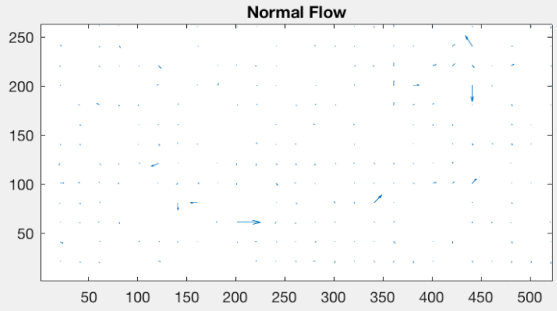
Toy_8 & Toy_9


Original Image

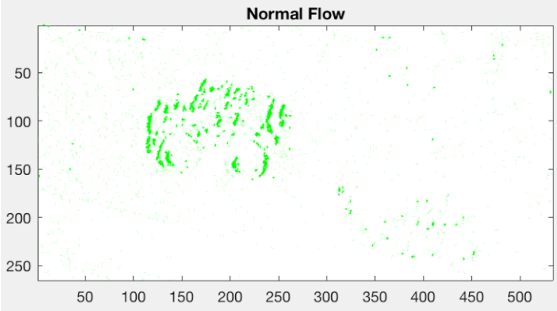

Temporal Gradient Image
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500


X Spatial Derivative
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500


Y Spatial Derivative
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500


Normal Flow
250
200
150
100
50
50 100 150 200 250 300 350 400 450 500


Normal Flow Overlay


Normal Flow
50
100
150
200
250
50 100 150 200 250 300 350 400 450 500


Normal Flow Overlay

1.1.3 Discussion

- As we can see, we have got pretty good results. Large spacing may not produce the intuitive result but the small spacing does. As the green vectors shown, we can see that they are around the skeleton of the toy cars. Because the toy cars are moving relative to the camera. And we can estimate that the ground is moving when there are many green vectors on the ground.
- What is more, with the large spacing, we can also estimate the approximate flow that around the toy cars. We can see that normal flow near some parts of the toy cars. But they are not so consistent, and I will explain later.
- However, the normal flow is not the true optical flow, because we can not get the true flow which is two components from only one equation constraints. But we can calculate the normal flow which is perpendicular to the constraint. But we can also get the approximation of the movement through the normal flows that we have calculated.

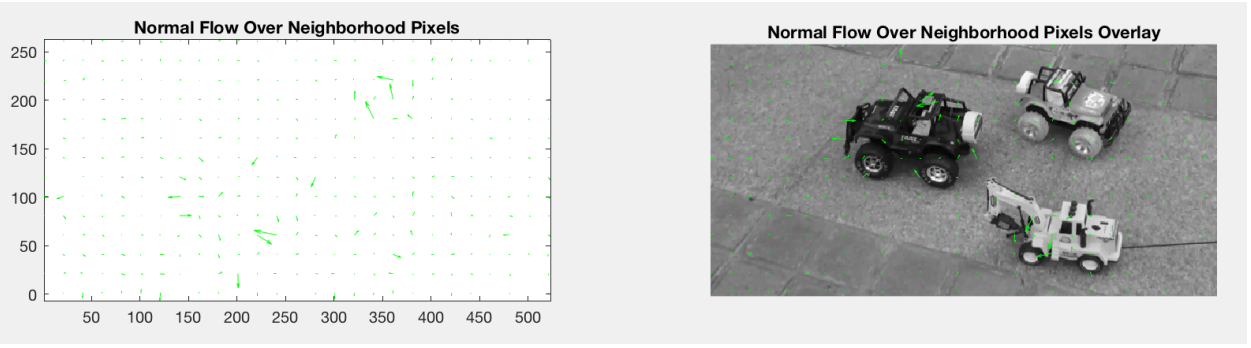
1.2 Optical Flow Over Neighborhood Pixels

1.2.1 Equation Summary

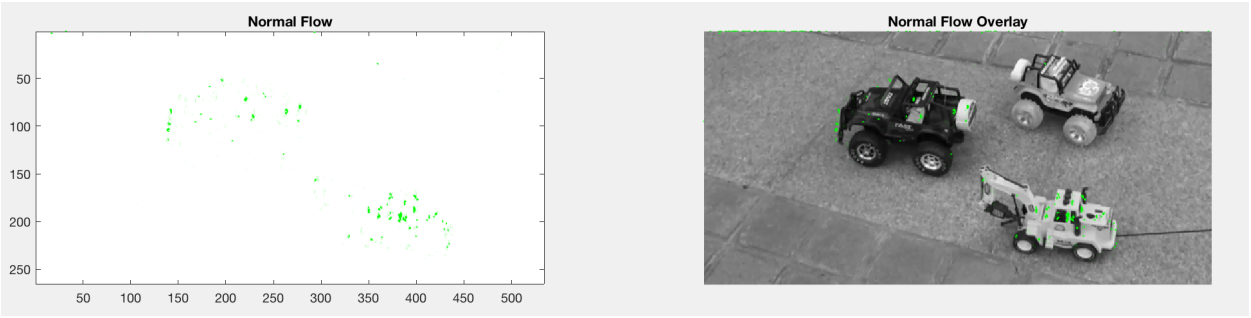
In this case, I use the equation that provided at the Assignment instructions. So I would not replicate them anymore. All I need to do is to stack the appropriate elements into the A and b matrix, using 2 * 2 neighborhood pixels and then solve the matrix based on the SVD method. To avoid the inconsistent rank of matrix, I use pinv() method instead of inv(). What's more, please note that when I use the 2 * 2 neighborhood pixels, I assume that I use 4 equations to estimate the velocity of the top-left pixel.

1.2.2 Results

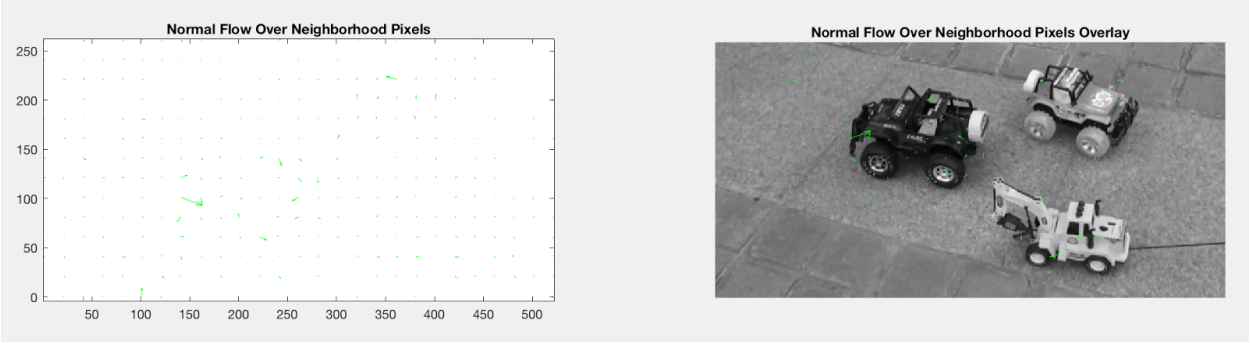
Sigma = 2.0, spacing = 20



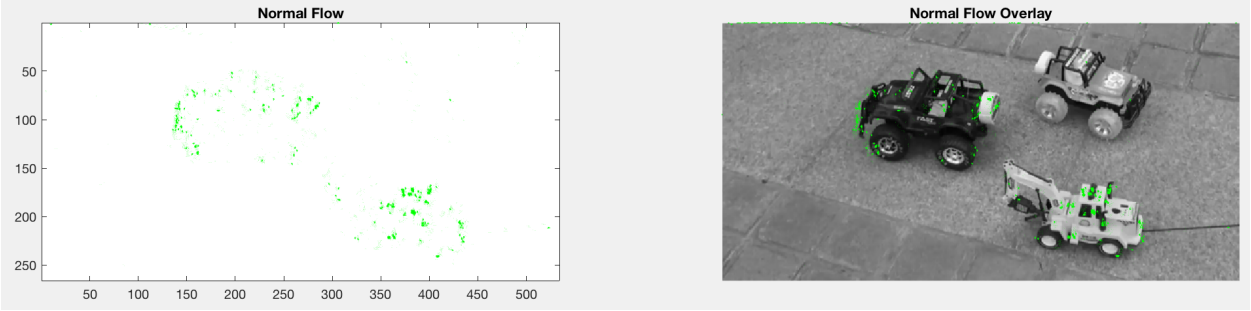
Sigma = 2.0, spacing = 1



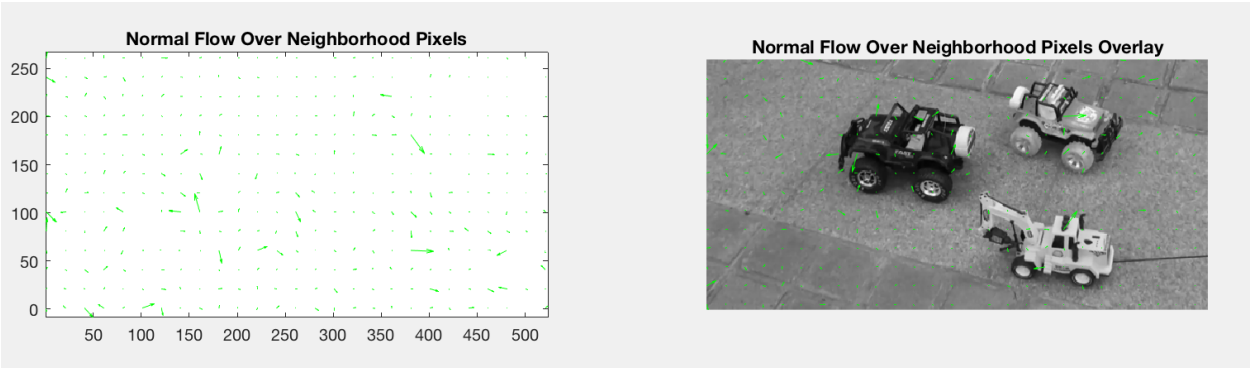
Sigma = 1.0, spacing = 20



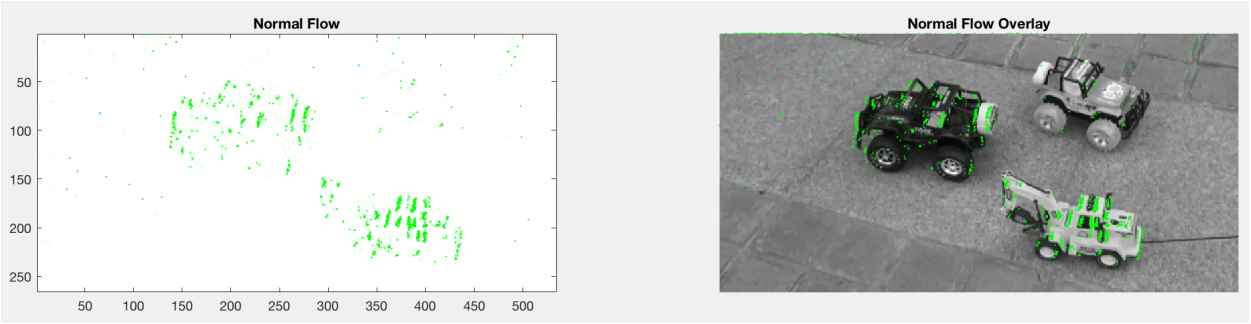
Sigma = 1.0, spacing = 1



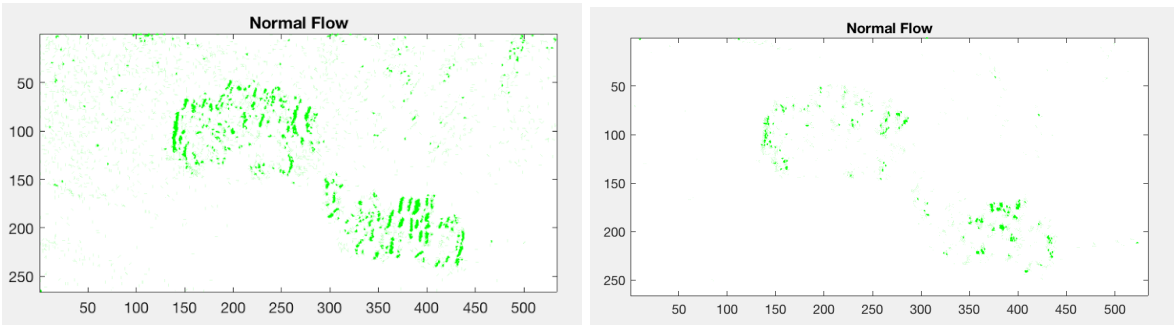
Non Gaussian Filter, spacing = 20



Non Gaussian Filter, spacing = 1



1.2.3 Comparison And Discussion



- To compare the previous method and current method, for the optical flow calculation over neighborhood pixels, because we use four pixels to estimate the velocity of one pixel(the top-left pixel in this case), we can get the true optical flow other than the normal flow that we got in the last section. The above two images are smoothed and plotted through the same style but with different methods. The first one is the normal flow that we got in the last section, the second one is the estimated optical flow that we got now. The first method has much more noise than the second method. So we can conclude that the second method is better than the first method.

- To compare the different results of the over neighborhood pixels method, as we can see, the more smoothed pictures(in this case: sigma 2.0 > sigma 1.0 > No filter), the less optical flows that we can see. Because when the image is more blurred, it lost more details, and the less optical flow would be calculated because the derivative between pixels are much smaller than less smoothed. However, although it would lose some details, but it also filter many noises. Many errors that would be displayed for non-filter images, which would be avoided in the smoothed images. In this case, it is a kind of contradict and complex problem because we could not figure out which one is better to estimate the true optical flow.

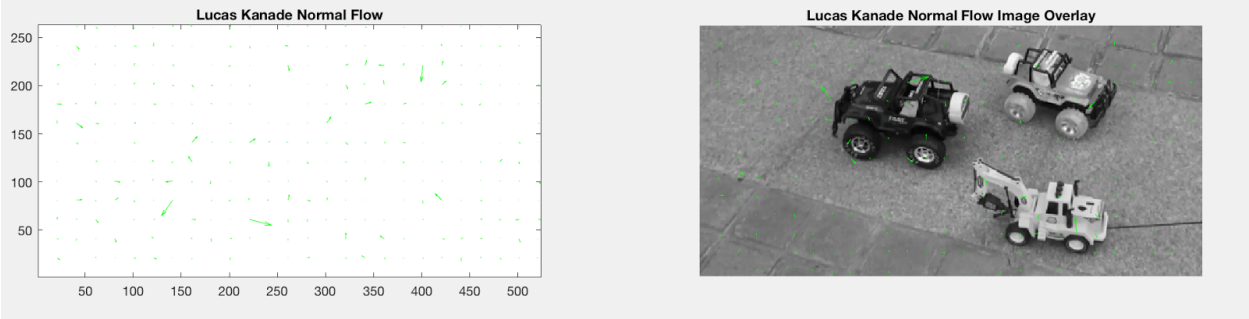
Bonus 1: Horn and Schunck or Lucas and Kanade Method

(Please note that there are some variations of the H&S or L&K method, but here we just implement the standard method)

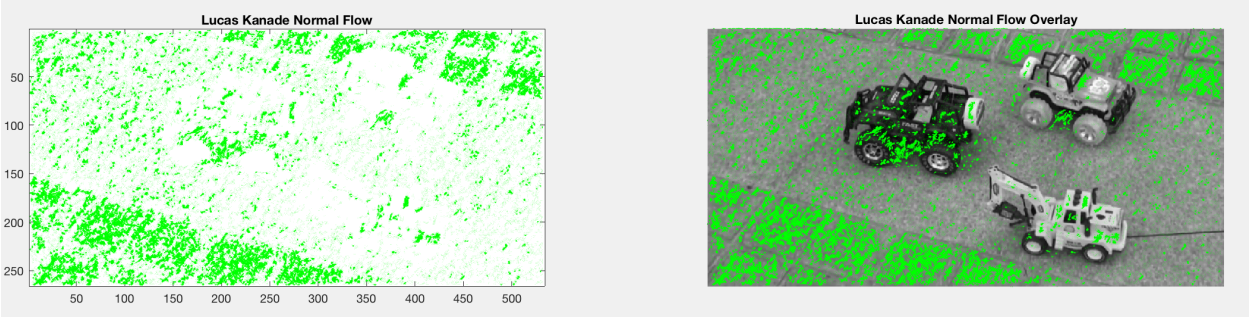
1.Lucas Kanade Method:

Actually, Lucas Kanade method is much like the simplified method that we used before. The most difference between them is the size of the neighborhood that we used. We will not show the pseudocode here because we just need to generalize the block size compared to the simplified over neighborhood pixels method.

Sigma = 1.0, Spacing = 20



Sigma = 1.0, Spacing = 1



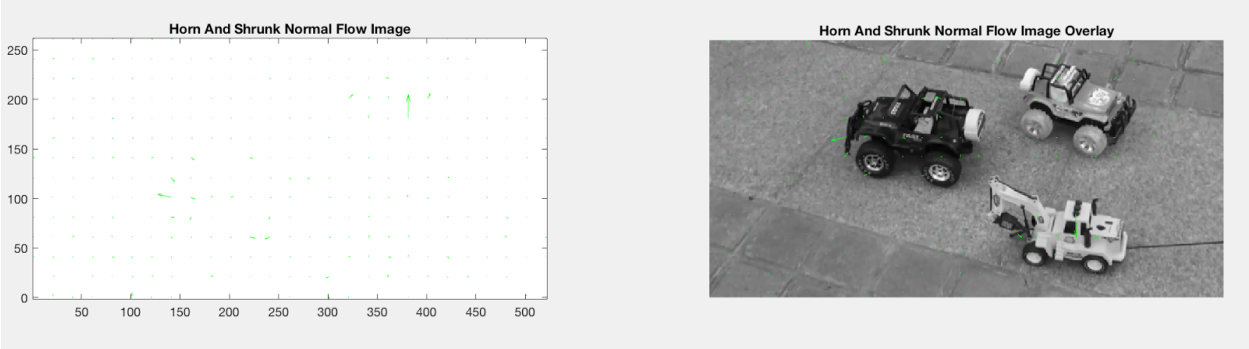
2.Horn And Shrunk

The following is the pseudocode for the Horn and Shrunk method, we use 25 time of iterations to generate the result optical flow vector images.

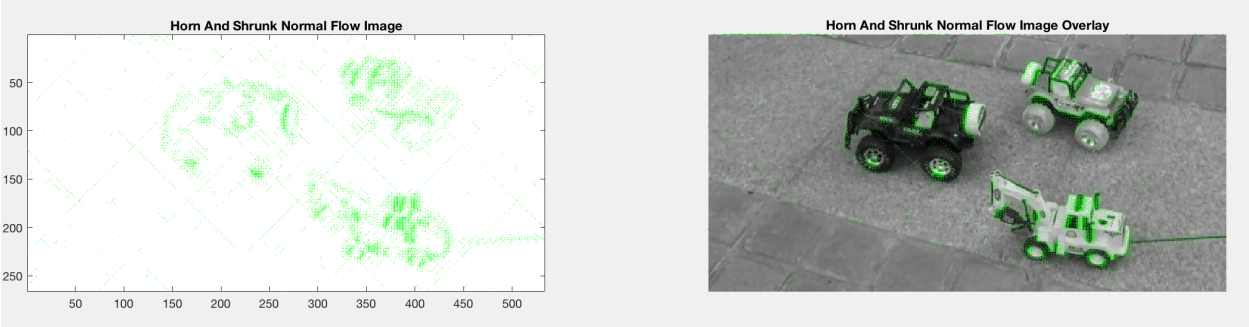
Data: two frames with moving objects
Result: displacement vector field $\vec{u} = (u, v)$
initialization;
compute spatial gradient I_x and I_y and time gradient I_t;
while *iteration* $n \leq n_0$ **do**
  $\bar{u} = \frac{1}{4}$(u(i-1,j)+u(i+1,j)+u(i,j-1)+u(i,j+1)) ;
  $\bar{v} = \frac{1}{4}$(v(i-1,j)+v(i+1,j)+v(i,j-1)+v(i,j+1)) ;
  $\alpha = \lambda \frac{I_x \bar{u}^{(n)} + I_y \bar{v}^{(n)} + I_t}{1+\lambda(I_x^2+I_y^2)}$;
  $u^{(n+1)} = \bar{u}^{(n)} - \alpha I_x$;
  $v^{(n+1)} = \bar{v}^{(n)} - \alpha I_y$;
  $n = n + 1$;
end

Algorithm 1: Horn and Shunck Optical Flow method

Sigma = 1.0, Spacing = 20

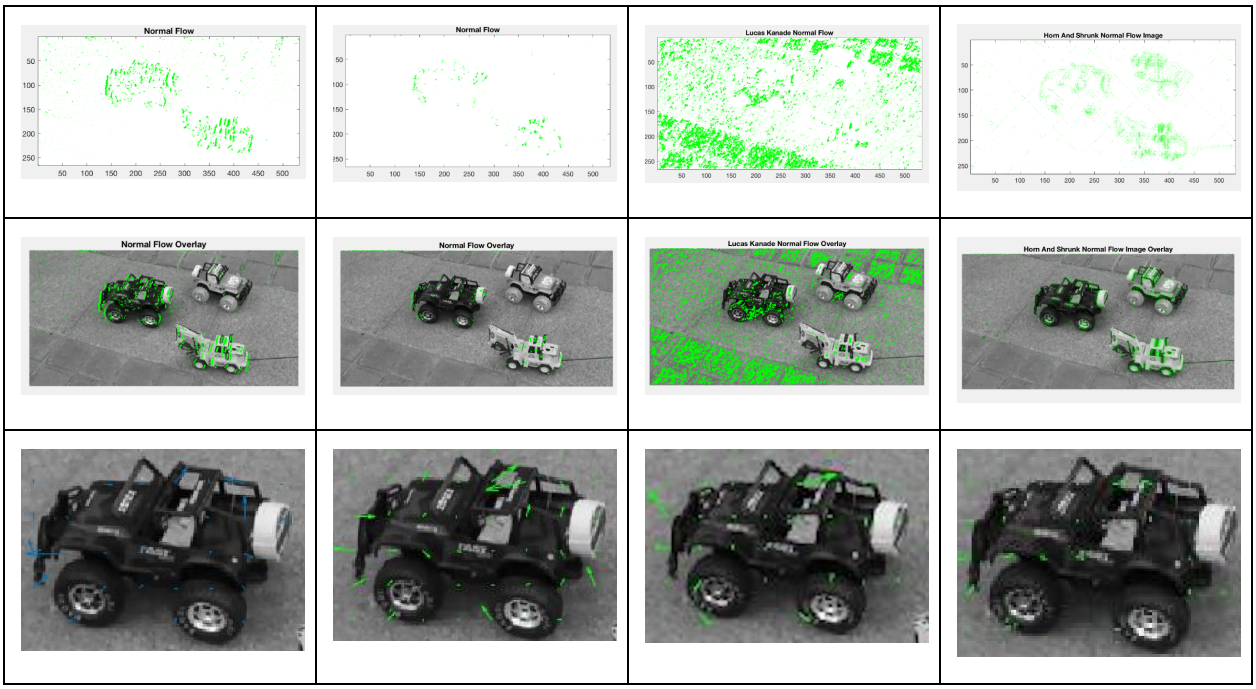


Sigma = 1.0, Spacing = 1



3. Method Comparison (Smoothed - Sigma = 1.0)

Normal Flow	Simplified OF	Lucas and Kanade	Horn and Shrunk



- As we can see, all of them are close to each other. The 2D vectors are almost around the skeleton of toy cars.
- In this case, the result of the Horn and Shrunk method may be the most representative and robust because it is a optimized method based on Lucas and Kanade Method. And we use 25 iterations to optimize the vector over times.
- As we can see, we use 3 * 3 block size to implement the Lucas and Kanade method. Many vectors are displayed on the shape of the ground. Change the block size may have a different result.
- For the zoomed patch of results, we can notice that the more simpler and more coarse method, the flows around one object are not more consistent than the more optimized method such as Horn and Shrunk. And the more robust and excellent algorithm may deal with the noise better.

Bonus 2: Apply optical flow to your own video sequence pair of images

I just recorded a screen video from the youtube. https://www.youtube.com/watch?v=Fg_00IbfZ-c
And transfer it in a very low resolution to calculate the optical flow.

Note: I read in a very small video and display the optical flow on the original image. In this case, I use the method that built in the MATLAB itself. The most difference between what we have done before is that we need to transfer the RGB image to the grey-scaled image first. And just apply the smoothing function and then calculate the optical flow via the MATLAB Lucas Kanade Method. To have a better display, I just scale the optical flow vector. (Because the instruction do not clarify that whether we could use the builtin method or not. To have a comparison, I use the builtin method in this case.)

As we can see, the result of this image is pretty good. The reason might be this function have been optimized some kind compared to the original implementation. It didn't show many noise but the reasonable and representative optical flow. It can show the movement of the runner in a very good way.

That's all!

Thank you!!